How to Train Your Latent Flow Map for Few-Step Text Generation

Anonymous authors
Paper under double-blind review

Abstract

Diffusion language models generate tokens in parallel, but they need many denoising steps, which offsets their speed advantage. Flow maps reduce the number of steps by learning to jump between noise levels in a single network evaluation. Prior flow-map language models build these maps over token embeddings or frozen encoder states, and each makes its own design choices, so it is unclear which choices matter. We study them in a latent space learned jointly with the denoiser: we distill a latent diffusion language model into a flow map within its own latent space and analyze each component of the distillation, namely the regression target, self-conditioning, rollout depth, decoder alignment and the treatment of the latent space. Refining the target with the teacher matters most for open-ended text, while aligning the decoder to the flow map's outputs is essential on GSM8K, where a single wrong token breaks a program. The pre-trained latent space can stay frozen, and adapting it during distillation, together with the teacher, can improve coverage (pass@k at large k) but not single-sample accuracy, which drops on Sudoku. With the resulting recipe, distillation on GSM8K takes 5k iterations, and our model with 8 steps exceeds its teacher with up to 64 steps and matches the best baseline, which needs 64 steps. On OpenWebText, with 8 steps, it has 40% lower generative perplexity than the strongest flow-map baseline at matched entropy.

1 Introduction

Diffusion language models generate text by refining all positions of a sequence in parallel (Li et al., 2022; Sahoo et al., 2024; Meshchaninov et al., 2026), so in principle they need far fewer network evaluations than an autoregressive model, which needs one per token. In practice, however, they need many denoising steps to reach high quality, which cancels much of the expected speedup (Lee et al., 2026). The difficulty is sharpest for discrete diffusion models (Sahoo et al., 2024; 2025), which predict every token independently at each step and therefore struggle to change many tokens at once without losing coherence.

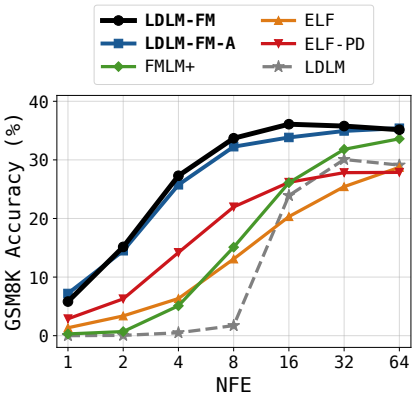


Figure 1: GSM8K accuracy against the number of steps (NFE). With 8 steps, LDLM-FM is as accurate as FMLM+ with 64 and more accurate than its teacher (LDLM) at any budget up to 64. FMLM+ values are from Agarwal et al. (2026), for its variant initialized from a pre-trained masked diffusion model.

Continuous diffusion does not factorize over tokens: it updates the whole sequence jointly, and its probability-flow ODE carries each noise sample to a sequence along a deterministic trajectory. A *flow map* learns this trajectory directly, moving a noisy state from one noise level to any lower one in a single evaluation (Boffi et al., 2025b), so generation needs only a few steps. Flow maps are often distilled from a pre-trained diffusion model, the *teacher*, which supplies the target that the flow map regresses onto. Recent flow-map language models build such maps over one-hot token embeddings, as FMLM (Lee et al., 2026) and FMLM+ (Agarwal et al., 2026) do, or over the frozen hidden states of a pre-trained encoder, as FMLM* (Yoo et al., 2026) does. None of these spaces is learned for denoising.

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.

LDLM (Meshchaninov et al., 2026) learns its latent space jointly with the diffusion model, and in a controlled comparison this space outperforms shallow token embeddings, frozen encoder states and separately trained autoencoders. A flow map in such a space, however, has more components to get right than a flow map over tokens. The decoder, which maps latents back to text, is trained on encoder outputs but receives flow-map outputs at inference. The space is also trainable, so distillation can keep it frozen or adapt it. If the space changes, the teacher will no longer be accurate on it. Every flow map also faces general choices, such as the regression target, and prior flow-map language models make them differently and evaluate them under different protocols, so it is unclear which choices matter.

In this work, we build a latent flow map by distilling LDLM within its own latent space and analyze the components of the flow map one at a time (Sections 4 and 5). The components are the diagonal target that the map regresses onto, self-conditioning on the model's previous estimate of the clean latent (Chen et al., 2023), the rollout depth (how many short jumps a long jump is trained to match), the alignment of the decoder to flow-map outputs, and the treatment of the latent space. We call the resulting model LDLM-FM. By default it keeps the latent space frozen, and the variant LDLM-FM-A adapts it.

As in FMLM*, we refine the target by iterating the self-conditioned teacher. On OpenWebText (Gokaslan & Cohen, 2019), the target matters most: with 8 steps, the iterated-teacher target gives 61% lower generative perplexity than the clean latent at matched entropy. On GSM8K (Cobbe et al., 2021), where one wrong token breaks a program, decoder alignment matters most: with 8 steps, it raises accuracy in our ablations from 14.8% to 39.1%, whereas on OpenWebText it brings no gain. On OpenWebText and GSM8K, adapting the space does not improve perplexity or accuracy. Adapting the space, while updating the teacher, can instead improve coverage, the share of problems that at least one of k samples solves. On GSM8K and on Sudoku puzzles (Agarwal et al., 2026), its pass@k starts at or below that of the frozen space at $k{=}1$, but grows faster with k and overtakes it.

Combining the best setting of each component gives our recipe (Table 6). The flow map has 136M parameters, and distillation takes 5k iterations on GSM8K and 20k on OpenWebText, at batch size 64 on two GPUs (Appendix A). On GSM8K, LDLM-FM with 8 steps is more accurate than its teacher at any budget up to 64 steps and as accurate as the best baseline result, FMLM+ with 64 steps (Figure 1). On OpenWebText, FMLM* is the strongest flow-map baseline. At the entropy of real text, LDLM-FM with 4 steps matches the perplexity and MAUVE of FMLM* with 8 steps, and with 8 steps LDLM-FM has 40% lower perplexity. Our contributions are:

- **A recipe for latent flow maps.** We motivate and ablate each component on OpenWebText and GSM8K, including decoder alignment, which is, to our knowledge, new to flow-map language models and essential on GSM8K (Sections 4 and 5.1).
- **The role of the latent space.** The jointly learned space can stay frozen, and adapting it together with the teacher can improve coverage but not single-sample accuracy, which drops on Sudoku (Sections 4.2 and 5.2).
- **Few-step results.** Our model matches or exceeds all flow-map and latent baselines we compare against: on GSM8K at every step budget, and on OpenWebText at 4 and 8 steps at matched entropy (Section 6).

2 Related work

Diffusion language models. Few-step methods for discrete diffusion (Sahoo et al., 2024) distill a many-step teacher into a factorized student, as in SDTT (Deschenaux & Gulcehre, 2025) and Duo (Sahoo et al., 2025), or into one that models dependencies between tokens, as in Di4C (Hayakawa et al., 2025). LKF (Ahmed et al., 2026) models these dependencies without a teacher. Recent continuous models have closed much of the gap to discrete methods (Lee et al., 2026; Chen et al., 2026; Hu et al., 2026). They differ in their representation space: one-hot token embeddings or the simplex (Lee et al., 2026; Roos et al., 2026), token embeddings learned with the model (Li et al., 2022; Chen et al., 2026), frozen encoder states (Shabalin et al., 2025; Hu et al., 2026; Yoo et al., 2026), or the latent space of a separately trained autoencoder (Lovelace et al., 2023; Meshchaninov et al., 2025; Guo et al., 2026). LDLM (Meshchaninov et al., 2026) instead trains a deep contextual encoder jointly with the diffusion model. We distill LDLM.

Flow maps and few-step distillation. Progressive distillation (Salimans & Ho, 2022), consistency trajectory models (Kim et al., 2024), shortcut models (Frans et al., 2025) and flow maps (Boffi et al., 2025b;a; Geng et al., 2025) reduce the number of sampling steps of continuous diffusion. FMLM (Lee et al., 2026) distills a flow over one-hot token embeddings into a flow map that generates in one step, and Potaptchik et al. (2026) and Roos et al. (2026) build flow maps on the probability simplex. FMLM+ (Agarwal et al., 2026) and MLFM (Azangulov et al., 2026) build masking into flow language models, so that any subset of tokens can serve as clean context, and FMLM+ re-noises low-confidence tokens at inference. FMLM* (Yoo et al., 2026) shows that self-conditioning performs one step of a fixed-point iteration and runs several steps to refine the target. Our diagonal and semigroup losses follow FMLM and FMLM*, and our targets iterate the teacher as FMLM* does. We differ from these works in three ways. We distill in a latent space learned jointly with the teacher and test whether to keep it frozen or adapt it (Section 4.2). We keep the self-conditioning input in the flow map, whereas FMLM* distills the iteration into a map without it. Finally, we align the decoder by training it on flow-map outputs, whereas TEncDM and LDLM train it only on noised encoder outputs (Shabalin et al., 2025; Meshchaninov et al., 2026). Our main baselines are FMLM* and FMLM+: FMLM* outperforms SDTT, Duo and Di4C at one to four steps with data-level entropy, and on GSM8K, FMLM+ trained from scratch with 32 steps matches MDLM and Duo with 1024.

3 Latent flow maps

3.1 Latent diffusion language models

Autoencoder. LDLM (Meshchaninov et al., 2026) encodes a token sequence $y = (y_1, \dots, y_n) \in \mathcal{V}^n$ in two stages. A frozen pre-trained token encoder $\mathcal{E}_h$ produces hidden states h, and a trainable latent encoder $\mathcal{E}_z^\theta$, a perceiver resampler (Alayrac et al., 2022), maps them to m latents $z_0 \in \mathbb{R}^{m \times d}$, none of which corresponds to a token position. We write $\mathcal{E}_z^\theta(y)$ for the two stages composed. Decoding mirrors this: a latent decoder $\mathcal{D}_h^\theta$ maps latents back to hidden states, and a token decoder $\mathcal{D}_y^\theta$ maps hidden states to tokens. Both decoders are trained to reconstruct from a noised latent, $\hat{h} = \mathcal{D}_h^\theta(\mathcal{E}_z^\theta(y) + \varepsilon)$ with $\varepsilon \sim \mathcal{N}(0, \sigma_{\text{dec}}^2 I)$:

$$\mathcal{L}_{\text{feat}} = \big\|\hat{h} - h\big\|^2, \qquad \mathcal{L}_{\text{ce}} = \text{CE}\big(\mathcal{D}_y^\theta(\text{sg}[\hat{h}]), y\big), \tag{1}$$

where $\text{sg}[\cdot]$ is the stop-gradient operator. $\mathcal{L}_{\text{feat}}$ is the only reconstruction loss that reaches the encoder, and $\mathcal{L}_{\text{ce}}$ trains $\mathcal{D}_y^\theta$ alone. The input noise σ_{dec} makes the latent decoder robust to imprecise latents and the latent space smoother (Meshchaninov et al., 2026).

Diffusion in the latent space. Diffusion runs on the encoder output $z_0 = \mathcal{E}_z^\theta(y)$ under a variance-preserving schedule,

$$z_t = \alpha_t z_0 + \sigma_t \epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \tag{2}$$

with α_t decreasing from 1 at $t = 0$ (data) to 0 at $t = 1$ (noise) and $\alpha_t^2 + \sigma_t^2 = 1$. A denoiser $T(z_t, t)$, our teacher, is trained to recover z_0 by minimizing $\mathcal{L}_{\text{diff}} = \mathbb{E}_{z_0,t,\epsilon}\|T(z_t, t) - z_0\|^2$. It is self-conditioned (Chen et al., 2023): it may also receive its previous estimate of z_0 as a self-conditioning input $\tilde{z}$, written $T(z_t, t; \tilde{z})$, and it is trained both with and without this input. We write $T(z_t, t)$ for $T(z_t, t; \mathbf{0})$. Encoder, decoders and denoiser minimize $\mathcal{L}_{\text{diff}} + \mathcal{L}_{\text{feat}} + \mathcal{L}_{\text{ce}}$ *jointly*, so the diffusion loss shapes the latent space. The gradient of $\mathcal{L}_{\text{diff}}$ into the encoder is ramped in only after a reconstruction-only warmup, and diffusion times are drawn from an adaptive distribution under which the denoising loss grows linearly in time (Meshchaninov et al., 2026).

3.2 Flow maps

Definition. Deterministic samplers such as DDIM (Song et al., 2021) integrate the probability-flow ODE associated with equation 2, with a denoiser in place of the posterior mean $\mathbb{E}[z_0 \mid z_t]$. The *two-time flow map* $X_{s,t}$ is the solution operator of this ODE: it maps the state at time t to the state of the same trajectory at any time $s < t$ in one evaluation (Boffi et al., 2025b). Since the trajectory through each state is unique, the map has the semigroup property:

$$X_{s,t} = X_{s,u} \circ X_{u,t}, \qquad s < u < t. \tag{3}$$

A long jump must therefore agree with every decomposition of it into shorter jumps, which makes the map hard to learn.

Parameterization. Following the teacher and Lee et al. (2026), we parameterize the flow map through a two-time denoiser $\psi_\theta(z_t, t, s)$ that outputs a clean latent: at $s = t$ it plays the role of the denoiser, and at $s = 0$ it is the endpoint of the trajectory. We omit its self-conditioning and guidance inputs (Section 4.1) from the notation, and statements about the ODE and the semigroup property hold for fixed values of these inputs. The map is recovered by inferring the noise implied at (z_t, t) and re-noising to time s:

$$X^\theta_{s,t}(z_t) = \frac{\sigma_s}{\sigma_t}\, z_t + \Big(\alpha_s - \frac{\sigma_s}{\sigma_t}\, \alpha_t\Big)\psi_\theta(z_t, t, s). \tag{4}$$

This is the DDIM update with ψ_θ as the clean-latent prediction, and it loses no generality, since the coefficient of ψ_θ is positive for $s < t$. At $s = t$ the bracket vanishes, so $X^\theta_{t,t}(z_t) = z_t$ by construction. The diagonal $\psi_\theta(z_t, t, t)$ nevertheless matters: differentiating equation 4 with respect to s at $s = t$ shows that it sets the velocity field of the ODE, and the semigroup property makes the map the flow of this field.

3.3 Distilling a flow map in the latent space

We distill the teacher into a latent flow map X^θ, parameterized by ψ_θ, inside its own latent space. By default, the encoder and the teacher are frozen, and only the flow map and the decoders are trained. We adapt the space in Section 4.2.

Diagonal and semigroup losses. On the diagonal, the map regresses onto a diagonal target $\hat{z}(z_t, t)$ derived from the teacher:

$$\mathcal{L}_{\text{diag}} = \big\|\psi_\theta(z_t, t, t) - \text{sg}[\hat{z}]\big\|^2. \tag{5}$$

With $\hat{z} = T(z_t, t)$, the map integrates the teacher's ODE. We instead refine the target (Section 4.1), so the map is not bound to this ODE. Off the diagonal, for $s < t$, let $R_r(z_t, t, s)$ denote r chained jumps of X^θ from t to s, with intermediate times placed at equal probability mass under the teacher's distribution of diffusion times. The map is trained to agree with it:

$$\mathcal{L}_{\text{semi}} = \big\|X^\theta_{s,t}(z_t) - \text{sg}\big[R_r(z_t, t, s)\big]\big\|^2, \tag{6}$$

where the rollout uses the current network, gradients flow only through the single jump, and the depth r is set in Section 4.1.

Full objective. The decoders keep training on the reconstruction losses of equation 1, and the full objective is

$$\mathcal{L} = \mathcal{L}_{\text{diag}} + \mathcal{L}_{\text{semi}} + \lambda_f\, \mathcal{L}_{\text{feat}} + \lambda_c\, \mathcal{L}_{\text{ce}}, \tag{7}$$

to which we add a decoder alignment term in Section 4.1. Half of each batch trains the diagonal loss and half the semigroup loss. On GSM8K, $1/64$ of the semigroup examples are pinned to $(s, t) = (0, 1)$ so that the one-step jump is trained explicitly. Loss weights and σ_{dec} are listed in Appendix A.

Conditional generation. Unlike token-aligned models such as FMLM+ (Agarwal et al., 2026), we cannot condition by masking answer positions. Instead, we split the latents into a question block of m_q latents and an answer block of m_a latents, $z = [z^q, z^a]$, and a block attention mask in the resampler lets the question block attend only to the question tokens y^q. The question block is encoded once, $z^q_0 = \mathcal{E}^\theta_z(y^q)$, and held fixed while the map generates the answer block.

Sampling. Generation chains N jumps of equation 4 on a time grid built as for the rollouts. In the stochastic variant, each step re-noises with a mixture of the implied noise $\hat{\epsilon} = (z_t - \alpha_t \psi_\theta(z_t, t, s))/\sigma_t$ and fresh noise ϵ', $z_s = \alpha_s \psi_\theta(z_t, t, s) + \sigma_s\big(\sqrt{1-\eta^2}\,\hat{\epsilon} + \eta\,\epsilon'\big)$, analogous to the γ-sampling of Kim et al. (2024). We call η the *churn*: $\eta = 0$ gives the deterministic update, $\eta = 1$ redraws the noise entirely, and the two coincide at $N = 1$. Sampler settings are listed in Appendix A.

4 A recipe for latent flow maps

We now describe the components of the recipe. Section 5 tests them in the same order.

4.1 COMPONENTS OF THE RECIPE

Refine the diagonal target. The diagonal target $\hat{z}$ in equation 5 sets the velocity field that the map integrates (Section 3.2), so errors in $\hat{z}$ carry over to every jump. The clean latent z_0 is an unbiased but noisy target for the posterior mean $\mathbb{E}[z_0 \mid z_t]$, which the teacher predicts directly, and the teacher's prediction is itself imperfect. Following FMLM* (Yoo et al., 2026), we build the target from the teacher itself. Iterating the self-conditioned teacher corrects its errors toward a fixed point, and guiding the iteration moves the target beyond the posterior mean, which at high noise averages over many plausible texts. The target uses an exponential moving average $\overline{T}$ of the teacher, which equals the teacher unless the teacher trains (Section 4.2).

Unconditional generation. We iterate $\overline{T}$, feeding each prediction back as its self-conditioning input. As in classifier-free guidance (Ho & Salimans, 2022; Hu et al., 2026), each iteration extrapolates, with weight W, from the prediction without this input toward the prediction with it:

$$\hat{z}^{(j+1)} = \overline{T}(z_t, t; \mathbf{0}) + W\big(\overline{T}(z_t, t; \hat{z}^{(j)}) - \overline{T}(z_t, t; \mathbf{0})\big), \qquad \hat{z} = \hat{z}^{(J)}, \quad \hat{z}^{(0)} = \mathbf{0}. \tag{8}$$

A single iteration ($J{=}1$) therefore returns the plain teacher prediction $\overline{T}(z_t, t)$, and each further iteration costs one teacher evaluation per diagonal example. As in FMLM*, we draw W per example log-uniformly, here from $[1, 2]$, and pass it to the flow map as an input. Multi-step sampling uses $W{=}4$, outside the training range, in every reported multi-step result (Appendix A).

Conditional target. We guide each iteration by the question instead, with a weight W_c, drawn per example from $\log \mathcal{U}[1, 3]$ and passed to the flow map as an input, as W is in the unconditional case, so that the target depends more strongly on the question. After each iteration, we reset the question block to its clean encoding. Each iteration then costs two teacher evaluations (equation 11, Appendix A), and $J{=}1$ is the question-guided teacher without refinement.

Keep the self-conditioning input. In an ideal flow map, the self-conditioning input adds no information, and FMLM* distills the iteration into a map without it (Yoo et al., 2026). A learned map is imperfect, and the self-conditioning input passes on the estimate of the clean latent from the previous step, so each step can refine this estimate instead of computing it again from the noisy state alone. We therefore expect the benefit to grow with the number of steps that use the input. At $N{=}1$, no step receives it. We keep the input, and the rollouts in equation 6 pass it as sampling does: the first step receives none, and each later step receives the previous prediction (Appendix A).

Deepen rollouts for long jumps only. The semigroup loss in equation 6 compares one jump against an r-step rollout of the flow map. For a short jump, a 2-step rollout is already accurate, but for a long jump it consists of long and inaccurate jumps. A deeper rollout follows the trajectory more closely, but it compounds the errors of the flow map and costs more evaluations. We therefore use 2-step rollouts for short jumps and 4-step rollouts for long ones, $[r_1, r_2] = [2, 4]$, with the threshold in Table 6. FMLM* compares every jump against a single midpoint, the $[2, 2]$ schedule in Table 1. Since one network serves all jumps, a better fit on long jumps can also improve the shorter jumps that multi-step sampling uses. In our experiments, the deeper rollout helps with 2 steps and does not hurt with more (Section 5.1).

Align the decoder when exact tokens matter. The decoders are trained on noised encoder outputs, but at inference they decode flow-map outputs, whose errors are structured, unlike the isotropic noise the decoders were trained on. For open-ended text a few decoding errors are tolerable, but a single wrong token can break an executable solution. We therefore also train the latent decoder on the detached one-step prediction of the flow map at low noise levels:

$$\mathcal{L}_{\text{align}} = \big\| \mathcal{D}_h^\theta\big(\text{sg}[\psi_\theta(z_t, t, 0)]\big) - h \big\|^2, \tag{9}$$

where h are the hidden states of the original text. We add $\mathcal{L}_{\text{align}}$ to equation 7 on GSM8K and Sudoku but not on OpenWebText. Since the prediction is detached, the term sends no gradient to the flow map. Section 5.1 also tests a cross-entropy term for the token decoder.

4.2 THE LATENT SPACE

Keep the space frozen unless coverage matters. The latent space determines what the map must learn: how far apart the two ends of a jump are and how curved the path between them is. LDLM

shapes its space with the denoising objective, and we test whether the flow map benefits from training the encoder further during distillation, through the reconstruction and distillation losses and, on OpenWebText, an anchor. We compare two treatments of the space in Section 5.2.

Frozen. LDLM-FM, the default of Section 3.3, keeps the encoder and the teacher frozen, so the teacher stays accurate on the space it was fitted to.

Adapted during distillation. In LDLM-FM-A, the encoder keeps training, and we make two changes. First, a frozen teacher would be evaluated on a drifting space, so we keep training it (K updates per step, with and without the self-conditioning input). Second, the distillation losses regress onto teacher targets rather than onto the encoder output, so, unlike in the joint training of LDLM, no loss ties the encoder directly to the predictions of the map, and its latents can become harder to denoise. On OpenWebText, we therefore add an anchor that pulls the encoder toward the detached diagonal prediction of the flow map:

$$\lambda_a \left\| \mathcal{E}_z^\theta(y) - \mathrm{sg}\big[\psi_\theta(z_t, t, t)\big] \right\|^2. \tag{10}$$

The encoder learning rate is annealed to zero early in training, so that the teacher and the flow map finish training on a fixed space.

5 Empirical analysis

Benchmarks. On OpenWebText, we distill the released LDLM checkpoint (Meshchaninov et al., 2026) and generate 1000 sequences of 1024 tokens for evaluation (Appendix A). On GSM8K, we train our own conditional teacher on TinyGSM (Liu et al., 2023), following Agarwal et al. (2026), and grade the generated Python solutions on the test split by execution. On Sudoku, we use data generated as in Agarwal et al. (2026) and train with our GSM8K recipe unchanged on 96k puzzles. A grid counts as solved when it is valid and matches the clues. Appendix B compares against FMLM+ retrained on the same data.

Metrics. On OpenWebText, we report generative perplexity (gPPL) under GPT-2 Large (Radford et al., 2019), per-sample unigram entropy, corpus-level diversity from repeated n-grams (Su et al., 2022), and MAUVE (Pillutla et al., 2021), since perplexity alone can be gamed even at data-level entropy (Franca & Tong, 2026). We report each model at its operating point: among the Pareto-optimal points of its sampler sweep in gPPL and entropy, the one whose entropy is closest to that of real text. Across four training seeds of our models, gPPL at this point varies by 1.9–4.9 points (Appendix A), so for all models we treat differences below 1.9 as ties and differences up to 4.9 as comparable.

On GSM8K, we report pass@1 over five evaluation rounds, and on GSM8K and Sudoku, pass@k with the estimator of Chen et al. (2021) over 128 rounds. Each GSM8K configuration is a single training run, and four runs of one configuration vary with a standard deviation of about 1.2 points (1.45 at $N{=}4$), so for all models we treat differences below 3 points as run-to-run variation. The GSM8K ablations are evaluated on a random subset of 1000 problems from the GSM8K training split. Our models train only on TinyGSM, so these problems are unseen, and the test split is reserved for the comparison in Section 6.

5.1 Ablations of the recipe

Each row of Tables 1 and 2 changes one choice of the recipe in the frozen space.

Diagonal target. The target has the largest effect on OpenWebText. With 8 steps, the refined target ($J{=}5$) gives a gPPL of 44.3, against 112.5 for the clean latent and 89.5 for a single teacher prediction ($J{=}1$). This suggests that refinement moves the target away from the posterior mean. At $N{=}8$, $J{=}3$ and $J{=}7$ reach lower gPPL (35.1 and 40.5) only at a lower diversity (25.8 and 27.5, against 29.6 for $J{=}5$ and 33.0 for real text). On GSM8K, the clean-latent target costs 14 points of pass@1 at $N{=}8$, while $J{=}1$, which is already guided by the question, matches $J{=}5$.

Self-conditioning. Keeping the self-conditioning input lowers gPPL on OpenWebText at both budgets, and a model trained without it is also worse (72.5 against 44.3 at $N{=}8$), so the gain is not

Table 1: **Ablations on OpenWebText (frozen space).** gPPL and diversity ($\times 100$) at matched entropy. Grey marks the recipe.

	N=2		N=8	
Configuration	gPPL	Div.	gPPL	Div.
Real text	15.0	33.0	15.0	33.0
Diagonal target				
clean z_0	154.6	35.9	112.5	39.3
J=1	200.8	40.6	89.5	37.9
J=3	126.8	36.4	35.1	25.8
J=5	106.5	36.0	44.3	29.6
J=7	100.9	34.8	40.5	27.5
Self-conditioning				
retained	106.5	36.0	44.3	29.6
off at eval	115.2	37.0	70.0	34.1
never trained	110.3	34.9	72.5	33.7
Rollout depth				
$[2, 2]$	115.1	35.4	46.9	29.8
$[2, 4]$	106.5	36.0	44.3	29.6
$[4, 4]$	94.0	32.4	38.8	24.1
Decoder alignment				
none	106.5	36.0	44.3	29.6
with CE	117.2	32.8	46.2	29.8
no CE	129.2	36.1	44.5	29.7
$t \leq 0.25$	121.6	35.6	41.3	28.8
$t \leq 0.5$	126.6	34.8	43.6	29.7

Table 2: **Ablations on GSM8K (frozen space).** Pass@1 (%) over five evaluation rounds on 1000 problems from the GSM8K training split. Grey marks the recipe. Removing decoder alignment costs about 25 points at N=8 and N=32.

Configuration	N=2	N=8	N=32
Diagonal target			
clean z_0	7.2	25.0	31.8
J=1	14.5	38.6	38.7
J=3	17.2	38.0	39.5
J=5	17.0	39.1	40.6
J=7	18.5	38.9	40.4
Self-conditioning			
retained	17.0	39.1	40.6
off at eval	17.9	35.3	37.9
never trained	18.2	34.5	36.7
Rollout depth			
$[2, 2]$	16.1	38.7	40.6
$[2, 4]$	17.0	39.1	40.6
$[4, 4]$	15.4	20.9	11.6
Decoder alignment			
none	6.4	14.8	15.4
with CE	10.8	32.8	35.3
$t \leq 0.25$	17.0	38.4	40.4
$t \leq 0.5$	17.0	39.1	40.6
$t \leq 1$	15.6	37.9	40.4

an artifact of a mismatch between training and sampling. On GSM8K, dropping the input lowers pass@1 by 2.7 to 4.6 points at N=8 and N=32, at the edge of run-to-run variation, while at N=2 all three settings are within 2 points. As predicted in Section 4.1, the benefit is larger with 8 steps than with 2.

Rollout depth. Deepening the rollout for long jumps lowers gPPL on OpenWebText by 9 points at N=2, even though no two-step sampling jump is long enough to use it: the better fit on long jumps carries over to shorter ones (Section 4.1). At N=8 the difference (2.6) is comparable to the seed deviation, and on GSM8K $[2, 2]$ and $[2, 4]$ are within run-to-run variation. Deepening the rollout for all jumps, $[4, 4]$, lowers gPPL on OpenWebText but collapses diversity (to 24.1 against 33.0 for real text at N=8), and on GSM8K its accuracy drops from 20.9 at N=8 to 11.6 at N=32, which suggests compounding errors.

Decoder alignment. As Section 4.1 predicts, alignment matters only where exact tokens do. On OpenWebText, it raises gPPL at N=2 (from 106.5 to 117.2–129.2) and changes it at N=8 by at most 3.0 points, comparable to the seed deviation, so we do not use it there. On GSM8K, it is essential: without it, pass@1 at N=8 falls from 39.1 to 14.8. The range of noise levels matters little, while adding a cross-entropy term costs 6 points.

5.2 The latent space

Frozen versus adapted. Adapting the space changes gPPL by at most 1.6 points at every budget in Table 3, within the seed deviation, and GSM8K pass@1 by at most 2.3 points (Figure 1). For single-sample quality, the frozen space is enough, and it is our default.

Where adaptation helps. Adapting the space can raise coverage without raising single-sample accuracy (Figure 2). On GSM8K, it raises pass@128 by 3.5 to 6 points at $N \in \{1, 2, 32, 64\}$, and at $N \in \{4, 8, 16\}$ the gain is within run-to-run variation. On Sudoku, it lowers pass@1 by 7 to 13

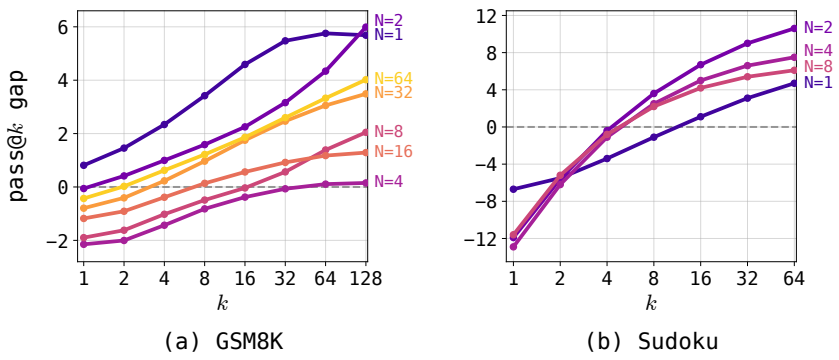


Figure 2: Pass@k of LDLM-FM-A minus LDLM-FM (points) on GSM8K (a) and Sudoku (b), one line per step budget N. As k grows, the adapted space closes its gap to the frozen one and overtakes it.

points but raises pass@64 by 5 to 11 points. The teacher also trains in LDLM-FM-A, so we cannot attribute the gain to the space alone. On OpenWebText, LDLM-FM-A has a higher MAUVE at N=8 (0.92 against 0.86). Appendix B ablates the adapted configuration.

6 Comparison with few-step language models

6.1 OpenWebText

We compare against FMLM (Lee et al., 2026), FMLM* (Yoo et al., 2026) and ELF with progressive distillation (ELF-PD; Hu et al., 2026), all evaluated from their released checkpoints and swept under our protocol (Appendix A). Figure 3 shows the front of each method, and Table 3 reports its operating point.

FMLM* is the strongest baseline: FMLM and ELF-PD reach their perplexity only at a diversity well below that of real text. With 4 and 8 steps, LDLM-FM and LDLM-FM-A outperform FMLM*. With 4 steps, LDLM-FM matches FMLM* with 8 steps in gPPL (68.9 against 73.6) and MAUVE (0.83 against 0.82), at a higher diversity. With 8 steps, its gPPL is 40% lower than that of FMLM* (44.3 against 73.6) at the same entropy. It also matches its teacher with 128 steps in gPPL (44.3 against 43.2) and MAUVE (0.86 against 0.89), at an entropy closer to real text (5.45 against 5.53). With 8 or fewer steps, the teacher's samples collapse. With 2 or fewer steps, every flow map stays far from real text (gPPL above 90, against 15), and FMLM* has the lowest gPPL at N=2.

6.2 GSM8K

We compare against the variant of FMLM+ (Agarwal et al., 2026) initialized from a pre-trained masked diffusion model, using its reported numbers, and against ELF and ELF-PD, which we train for this task. We could not obtain a working GSM8K model by adapting the released training pipeline of FMLM*, so it is not included. LDLM-FM is at least as accurate as every latent and flow-map baseline at every budget, and more accurate by more than run-to-run variation up to 32 steps (Figure 1). With 8 steps, it reaches 33.7%, as much as FMLM+ with 64 steps (33.6%) and more than its teacher with up to 64 steps, which peaks at about 30%. The teacher keeps its original decoder, and removing decoder alignment costs about 24 points at 8 steps in our ablations (Table 2), so the aligned decoder is part of this result. With 16 steps, LDLM-FM exceeds FMLM+ with 64.

7 Conclusion

We introduced LDLM-FM, a latent flow map distilled from LDLM, and analyzed its components one at a time. Refining the diagonal target matters most for open-ended text, decoder alignment is essential on GSM8K, self-conditioning helps more with 8 steps than with 2, and deepening the rollouts for all jumps hurts with 8 or more steps. Freezing the jointly learned space costs no single-sample

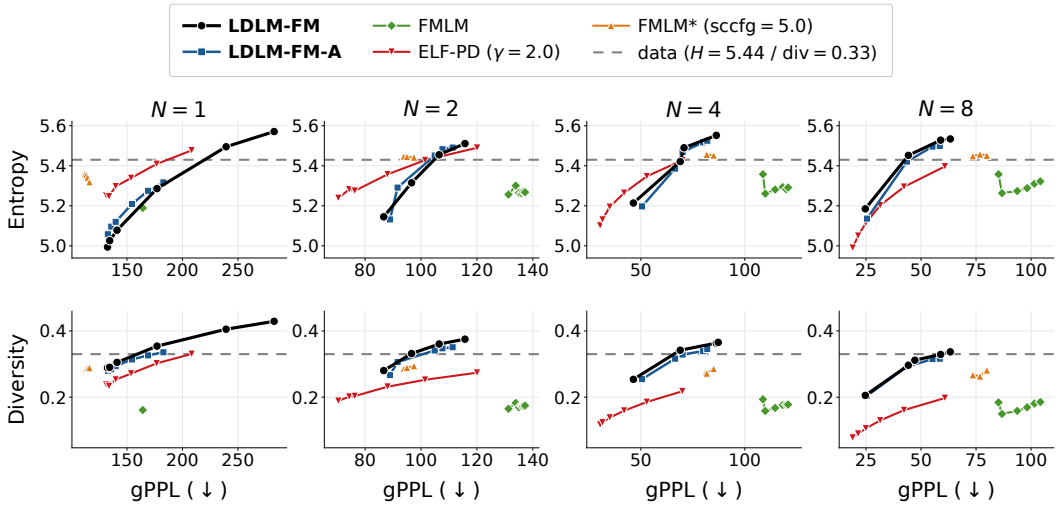


Figure 3: **Quality–diversity fronts on OpenWebText** with N=1, 2, 4 and 8 steps (columns): generative perplexity against per-sample entropy (top) and corpus-level diversity (bottom) along each method's sampler sweep. Dashed lines mark real text. With 4 and 8 steps, LDLM-FM has lower gPPL than FMLM* and higher diversity than ELF-PD.

Table 3: **Few-step generation on OpenWebText.** Each entry is the method's operating point in Figure 3. LDLM, the teacher, is shown with 128 steps in every column as a many-step reference and does not appear in the figure. Red flags entropy below 5.40 or diversity below 0.25. Bold marks the best gPPL and MAUVE per column among unflagged few-step entries, together with ties: gPPL within 1.9, the smallest seed deviation, and MAUVE less than 0.05 below the best. †ELF-PD uses a different tokenizer and yields fewer than 1024 GPT-2 tokens per sample, which collapses its MAUVE. Grey values truncate generations and references to 900 tokens.

	N=2				N=4				N=8			
Method	gPPL	Ent.	Div.	MAUVE	gPPL	Ent.	Div.	MAUVE	gPPL	Ent.	Div.	MAUVE
Real text	15.0	5.44	0.33	1.00	15.0	5.44	0.33	1.00	15.0	5.44	0.33	1.00
LDLM, N=128	43.2	5.53	0.31	0.89	43.2	5.53	0.31	0.89	43.2	5.53	0.31	0.89
FMLM	133.9	5.30	0.18	0.03	108.7	5.36	0.19	0.08	85.2	5.36	0.18	0.23
ELF-PD	101.5	5.43	0.25	0.01 0.39†	69.9	5.43	0.22	0.01 0.64†	60.8	5.40	0.20	0.02 0.69†
FMLM*	**93.9**	5.45	0.29	**0.73**	81.3	5.46	0.28	0.79	73.6	5.45	0.27	0.82
LDLM-FM	106.5	5.46	0.36	**0.71**	**68.9**	5.42	0.34	**0.83**	**44.3**	5.45	0.30	0.86
LDLM-FM-A	104.9	5.45	0.34	0.68	**70.1**	5.47	0.33	**0.87**	**43.6**	5.42	0.29	**0.92**

quality, and adapting it together with the teacher can raise coverage but lowers pass@1 on Sudoku. On GSM8K, distillation takes 5k iterations, and LDLM-FM with 8 steps is more accurate than its teacher with up to 64 steps and as accurate as FMLM+ with 64. On OpenWebText, LDLM-FM with 8 steps has 40% lower perplexity than FMLM* at matched entropy. These results suggest that a latent diffusion language model can be distilled into a strong few-step generator without retraining its latent space.

Limitations. We report network evaluations rather than wall-clock latency, and the GSM8K comparison with FMLM+ uses its reported numbers. Our evidence covers three benchmarks at a single model scale, with one teacher family and one latent space, and each GSM8K configuration is a single training run. With one or two steps, every method we evaluate remains far from real text on OpenWebText.

AI USE STATEMENT

We used large language models to help write and edit code for training, evaluation and baselines, to summarize and compare related work and check our descriptions of it against the cited papers, to write plotting scripts and lay out tables and figures, and to draft, restructure and rephrase the text. The authors designed the method and experiments, ran all experiments, reviewed all generated code and text, and take full responsibility for the content of the paper.

ETHICS STATEMENT

This work uses only publicly available datasets (OpenWebText, TinyGSM, GSM8K and generated Sudoku puzzles) and involves no human subjects. Like other language models, the models we train can generate incorrect or harmful text, and faster generation lowers the cost of producing it.

REPRODUCIBILITY STATEMENT

Sections 3.3 and 4 describe the full training objective and every component of the recipe. Appendix A gives the optimizer settings, schedules, batch sizes, training lengths and hardware, and Table 6 lists all hyperparameters for both datasets. On OpenWebText, we start from the publicly released LDLM checkpoint and evaluate baselines from their released checkpoints. The evaluation protocol, including the selection of operating points, the MAUVE setup and the number of evaluation rounds, is described in Section 5 and Appendix A. We will release our code upon publication.

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. *arXiv preprint arXiv:2606.24773*, 2026.

Mansoor Ahmed, Yue-Tsz Fan, Hemanth Venkateswara, and Murray Patterson. Latent-kernel discrete flow maps for few-step generation. *arXiv preprint arXiv:2607.27529*, 2026.

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. In *Advances in Neural Information Processing Systems*, volume 35, pp. 23716–23736, 2022.

Iskander Azangulov, Kianoosh Ashouritaklimi, Leo Zhang, Simon Vary, and Patrick Rebeschini. Masked language flow models. *arXiv preprint arXiv:2606.27617*, 2026.

Nicholas Boffi, Michael Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, pp. 33346–33382. Curran Associates, Inc., 2025a. doi: 10.52202/085713-1121. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/2ff3605c9e442d55156d99fe4f5f228d-Paper-Conference.pdf.

Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Trans. Mach. Learn. Res.*, 2025, 2025b. URL https://openreview.net/forum?id=cqDH0e6ak2.

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.

Ting Chen, Ruixiang Zhang, and Geoffrey E. Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=3itjR9QxFw.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. Langflow: Continuous diffusion rivals discrete in language modeling. *arXiv preprint arXiv:2604.11748*, 2026.

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. *arXiv preprint arXiv:2110.14168*, 2021.

Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast LLMs via self-distillation through time. In *International Conference on Learning Representations*, 2025.

Antonio Franca and Alexander Tong. Hacking generative perplexity: Why unconditional text evaluation needs distributional metrics. In *ICML Workshop on Structured Probabilistic Inference and Generative Modeling (SPIGM)*, 2026. URL https://arxiv.org/abs/2606.08417.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. In *International Conference on Learning Representations*, 2025.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *Advances in Neural Information Processing Systems*, volume 38, pp. 75460–75482, 2025. doi: 10.52202/085713-2534. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/6d13e085b79d454da5910e4ca82a3d9d-Paper-Conference.pdf.

Aaron Gokaslan and Vanya Cohen. Openwebtext corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Hongcan Guo, Qinyu Zhao, Yian Zhao, Shen Nie, Rui Zhu, Qiushan Guo, Feng Wang, Tao Yang, Hengshuang Zhao, Guoqiang Wei, and Yan Zeng. Continuous latent diffusion language model. *arXiv preprint arXiv:2605.06548*, 2026.

Satoshi Hayakawa, Yuhta Takida, Masaaki Imaizumi, Hiromi Wakaki, and Yuki Mitsufuji. Distillation of discrete diffusion through dimensional correlations. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 22259–22297. PMLR, 2025. URL https://proceedings.mlr.press/v267/hayakawa25a.html.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. Elf: Embedded language flows. *arXiv preprint arXiv:2605.10938*, 2026.

Dongjun Kim, Chieh-Hsin Lai, Wei-Hsiang Liao, Naoki Murata, Yuhta Takida, Toshimitsu Uesaka, Yutong He, Yuki Mitsufuji, and Stefano Ermon. Consistency trajectory models: Learning probability flow ODE trajectory of diffusion. In *International Conference on Learning Representations*, 2024.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M. Boffi, and Jinwoo Kim. Flow map language models: One-step language modeling via continuous denoising. *arXiv preprint arXiv:2602.16813*, 2026.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation. *Advances in Neural Information Processing Systems*, 35: 4328–4343, 2022.

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. *arXiv preprint arXiv:2312.09241*, 2023.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent diffusion for language generation. In *Advances in Neural Information Processing Systems*, volume 36, pp. 56998–57025, 2023.

Viacheslav Meshchaninov, Egor Chimbulatov, Alexander Shabalin, Aleksandr Abramov, and Dmitry Vetrov. Cosmos: Compressed and smooth latent space for text diffusion modeling. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Viacheslav Meshchaninov, Alexander Shabalin, Egor Chimbulatov, Nikita Gushchin, Ilya Koziev, Alexander Korotin, and Dmitry Vetrov. How to train your latent diffusion language model jointly with the latent space. *arXiv preprint arXiv:2605.07933*, 2026.

Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. MAUVE: Measuring the gap between neural text and human text using divergence frontiers. In *Advances in Neural Information Processing Systems*, volume 34, pp. 4816–4828, 2021.

Peter Potaptchik, Jason Yim, Adhi Saravanan, Peter Holderrieth, Eric Vanden-Eijnden, and Michael S. Albergo. Discrete flow maps. *arXiv preprint arXiv:2604.09784*, 2026.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, İsmail İlkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps. In *International Conference on Machine Learning*, 2026. arXiv:2602.12233.

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*, 37:130136–130184, 2024.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T Chiu, and Volodymyr Kuleshov. The diffusion duality. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 52584–52619. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/sahoo25a.html.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *The Tenth International Conference on Learning Representations, ICLR*. OpenReview.net, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Alexander Shabalin, Viacheslav Meshchaninov, Egor Chimbulatov, Vladislav Lapikov, Roman Kim, Grigory Bartosh, Dmitry Molchanov, Sergey Markov, and Dmitry Vetrov. TEncDM: Understanding the properties of the diffusion model in the space of language model encodings. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pp. 25110–25118, 2025. doi: 10.1609/aaai.v39i23.34696.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021.

Yixuan Su, Tian Lan, Yan Wang, Dani Yogatama, Lingpeng Kong, and Nigel Collier. A contrastive framework for neural text generation. In *Advances in Neural Information Processing Systems*, volume 35, pp. 21548–21561, 2022.

Jaehoon Yoo, Wonjung Kim, Floor Eijkelboom, Chanhyuk Lee, Nicholas M. Boffi, Seunghoon Hong, and Jinwoo Kim. Self-conditioned flow map language models via fixed-point flows. *arXiv preprint arXiv:2607.00714*, 2026.

A Implementation details

Training. On GSM8K, all models are initialized from our conditional teacher, trained for 5000 iterations at batch size 64 on two H100 GPUs. Training times are drawn from a logit-normal distribution following Hu et al. (2026) rather than the adaptive schedule of Meshchaninov et al. (2026), which performed worse on this task. In the adapted configuration, the encoder shares the flow

map's optimizer (base 10^{-4}, linear warmup over 500 steps) through a parameter group whose cosine scale anneals from 1 to 0 over the first 300 steps, so its effective rate peaks near 1.6×10^{-5} and is zero from step 300 of 5000 onward. The teacher takes $K{=}5$ updates per step under its own optimizer at learning rate 10^{-4}, and the diagonal target is formed from an EMA copy with decay 0.999. All GSM8K numbers are reported at the final 5000-iteration checkpoint, with no checkpoint selection.

On OpenWebText we start from the released checkpoint of Meshchaninov et al. (2026) and keep its adaptive time schedule. Models train for 20000 iterations on two H100 GPUs at batch size 64, with AdamW, learning rate 10^{-4} decayed by cosine to 10^{-5} after 5000 warm-up steps, weight decay 0, dropout 0.1. All OpenWebText numbers are reported at the final 20000-iteration checkpoint, and no checkpoint selection is performed. In the adapted configuration, the encoder has no learning rate of its own. It sits in a separate parameter group sharing the flow map's schedule (base 10^{-4}, cosine to 10^{-5}, linear warmup over 5000 steps) with a cosine scale annealed from 1 to 0 over the first 3000 steps. Because the anneal ends before warmup does, the encoder's effective rate peaks near 1.6×10^{-5} around step 1250 and is zero from step 3000 of 20000 onward.

Cost of adaptation. Both OpenWebText configurations of Section 5.2 run at the same batch size. Relative to the frozen configuration, adapting the encoder adds its backward pass, the encoder forward required by the anchor term, and K teacher updates per step.

Evaluation. GSM8K accuracy is graded by executing the generated Python solution on the 1319-problem test split, and ablations on a random 1000-problem subset of the training split. Every reported number is a mean over independent evaluation rounds (5 for single-sample accuracy, 128 for pass@k) under the stochastic sampler, $\eta{=}1$.

OpenWebText metrics are computed on 1000 generated sequences of 1024 tokens: generative perplexity under GPT-2 Large, per-sample unigram entropy, and the corpus-level diversity $\prod_{n=2}^{4}(1 - \mathrm{rep}_n)$, where rep_n is the fraction of repeated n-grams (Su et al., 2022). Each method trades perplexity against entropy through its sampler settings, so we compare methods at a common entropy rather than at a fixed setting. Every OpenWebText method is swept over churn and self-conditioning guidance (FMLM exposes only churn). For each method, we keep the slice of this grid with the best trade-off between perplexity and entropy: for $N \geq 2$, we fix W and sweep η for LDLM-FM and FMLM*, and we fix its own churn parameter and sweep guidance for ELF-PD. Figure 3 shows these slices. The *operating point* of a method is the Pareto-optimal point of its slice, in gPPL and entropy, whose entropy is closest to that of real text. All OpenWebText tables report operating points, and MAUVE plays no part in selecting them.

Run-to-run variability on OpenWebText is measured at the operating point, from four independently trained replicates of the recipe: the deviation of perplexity is 1.9–4.9 points across budgets and that of diversity is 0.007–0.010. MAUVE uses `gpt2-large` features and five disjoint 1000-document reference samples of the training split, scored against each, and we report the mean across references. Generations are decoded without special tokens, as the references are: leaving the end-of-text marker in the text collapses MAUVE for every latent model, ours included. Baseline OpenWebText numbers are computed from the authors' released checkpoints under this protocol.

Table 4: **Pipeline sizes on OpenWebText.** Parameters trained and frozen during training, and active at inference.

	training		inference
Model	trained	frozen	active
FMLM	169.7M	–	169.7M
ELF-PD	104.6M	35M	104.6M
FMLM*	157.3M	774M	157.3M
LDLM-FM	258.4M	308.4M	258.4M
LDLM-FM-A	442.4M	124.4M	258.4M

Table 5: **Components of LDLM-FM.** Parameter counts of each component on OpenWebText.

Component	Params
Encoder (two-stage)	**175.6M**
⊢ frozen GPT-2 text enc.	124.4M
⌞ perceiver-resampler	51.2M
Flow map	**136.4M**
Decoder (two-stage)	**122.0M**
⊢ latent decoder	55.1M
⌞ token decoder	66.9M

Architecture. All components of the pipeline of Section 3.1 are feed-forward. Table 4 compares pipeline sizes on OpenWebText. The trained count of LDLM-FM-A includes the 51.2M resampler and the 132.8M teacher, which receives K optimizer updates per step, and both are frozen in LDLM-FM. A matched evaluation count is not a matched computation: at every step, LDLM-FM runs its 136.4M flow map, against 169.7M parameters for FMLM and 157.3M for FMLM*, and it runs its 122.0M decoder once per sample. We report evaluation counts because they are the axis prior few-step work reports.

At inference only the flow map and the two-stage decoder are used (on GSM8K, the encoder also runs once on the question). On both datasets, the map is trained conditioned on the guidance weight of its target, drawn per example and supplied as an input: the self-conditioning weight $W \sim \log\mathcal{U}[1, 2]$ on OpenWebText and the question weight $W_c \sim \log\mathcal{U}[1, 3]$ on GSM8K. A single distilled model therefore exposes this weight as a free sampling parameter, and sweeping it costs no additional network evaluations. Both models use the previous step's clean-latent prediction as the self-conditioning input during generation. The rollouts of equation 6 pass this input the same way during training: the first step receives none and every later step receives the previous step's prediction. The single jump and the diagonal branch receive a self-conditioning input with probability p_{sc}, drawn once per example: a first pass without the input supplies a detached prediction that serves as the input for the second. Parameter counts are given in Table 5.

Conditional latent layout. The resampler maps a token sequence to a fixed number of latents by cross-attention, which removes any positional correspondence between latents and tokens: every latent may attend to every position. The question/answer split is therefore enforced by a block attention mask in the resampler. We use m_q=192 and m_a=448 within a 640-latent budget. The budget constrains the *concatenation*, so the two marginals never reach their maxima together – a 206-token question does not accompany a 592-token program – and a fixed split cannot allocate the full budget to both sides. At $192/448$, truncation affects 0.001% of training questions and 0.334% of training solutions, and no row of the GSM8K test split.

Conditional target. For conditional generation (Section 4.1), each iteration is guided by the question with the weight W_c drawn for that example. We write $[\varnothing, z^a]$ for z with its question block set to zero. With the conditional branch $A^{(j)}$ and the unconditional branch $B^{(j)}$, each iteration is then

$$\hat{z}^{(j+1)} = B^{(j)} + W_c\left(A^{(j)} - B^{(j)}\right), \qquad \begin{aligned} A^{(j)} &= \overline{T}\left(z_t,\, t;\, \hat{z}^{(j)}\right), \\ B^{(j)} &= \overline{T}\left([\varnothing, z_t^a],\, t;\, [\varnothing, \hat{z}^a]^{(j)}\right), \end{aligned} \tag{11}$$

with $\hat{z}^{(0)}$ and $\hat{z}$ as in equation 8. Both branches receive the previous iterate as their self-conditioning input, and the unconditional branch has the question block zeroed in this input as well, so no information about the question leaks into it. After each iteration, the question block is reset to its clean encoding, so guidance acts only on the answer.

B Additional ablations

Adapted space. Tables 8 and 9 ablate the adapted configuration of Section 4.2. Reducing the teacher updates from K=5 to K=2 changes little on either benchmark. The anchor is essential on OpenWebText: without it, gPPL is about five times higher, and no point of the front comes near the entropy of real text (the closest lie at 5.76 and 5.64). On GSM8K, the anchor does not help, so we omit it there. On OpenWebText, shorter anneals of 500 and 1000 steps raise gPPL at N=8 to 51.6 and 48.0, against 43.6. Longer anneals lower gPPL at N=2 but reduce diversity, to 27.9 at N=8 for 10000 steps, and 20000 steps raises gPPL at N=8 to 47.0. On GSM8K, extending the anneal from 300 to 1000 steps costs 3.8 points of pass@1 at N=8, and extending it to 3000 steps costs 9.5. The best anneal thus depends on the setting, which also differs in the anchor and the training length. Accuracy is pass@1 over five evaluation rounds under the stochastic sampler (η=1), and each configuration is a single training run.

Sudoku. The Sudoku training set of Agarwal et al. (2026) is small, so we generate more puzzles and train on 96k in total: 48k for the teacher and 48k for distillation. For a fair comparison, we

Table 6: Hyperparameters. The upper block is the frozen configuration of Section 3.3, and the lower block applies only when the encoder is trained during distillation (Section 4.2). Sampler sweeps hold W at 4 and vary $\eta \in [0, 1]$ for $N \geq 2$, and vary W at $\eta = 0$ for $N{=}1$, where churn is inert. The rollout depth $[r_1, r_2]$ gives the number of steps for jumps whose span of the diffusion-time distribution is below and above 0.7.

Symbol	Meaning	OpenWebText	GSM8K
Objective			
λ_f	feature-reconstruction weight, equation 7	0.5	1.0
λ_c	cross-entropy weight, equation 7	1.0	1.0
σ_{dec}	decoder input noise scale, equation 1	3.0	3.0
λ_{al}	decoder alignment weight, equation 9	0	1.0
–	alignment time range	–	$t < \frac{1}{2}$
Branch composition			
–	diagonal fraction of each batch	0.5	0.5
–	semigroup share at $(s, t) = (0, 1)$	0	$1/64$
$r(s, t)$	rollout depth by jump span	$[2, 4]$	$[2, 4]$
Target construction			
J	iterations of equation 8, equation 11	5	5
$[w_{\min}, w_{\max}]$	self-conditioning weight range	$[1, 2]$	–
W	self-conditioning weight at inference	swept over $[0.5, 4]$	–
The training range was chosen by scoring the teacher over candidate weights.At inference we sweep more widely, and multi-step operating points use $W{=}4$, outside the range the flow map was trained on.			
$[w_{c,\min}, w_{c,\max}]$	question guidance weight range	–	$[1, 3]$
W_c	question guidance weight at inference	–	2.5
p_{sc}	self-conditioning probability	0.5	0.5
–	teacher EMA decay	0.999	0.999
Conditional latents			
m_q	question latents	–	192
m_a	answer latents	–	448
Training and sampling			
η	churn	swept	1
–	training time schedule	adaptive	logit-normal
–	training iterations / checkpoint	20000 / final	5000 / final
–	batch size	64	64
Adapted space only			
K	teacher updates per step	5	5
–	teacher learning rate	10^{-4}	10^{-4}
–	encoder anneal (steps)	3000	300
λ_a	encoder anchor weight, equation 10	0.7	0

retrain FMLM+ with its released recipe on the same 96k puzzles in a single stage. Table 7 reports single-sample accuracy on the hard test split. LDLM-FM solves more puzzles than FMLM+ at every budget, and the gap is largest with one and two steps. LDLM-FM-A solves fewer puzzles than LDLM-FM at every budget, in line with Figure 2.

Table 7: **Sudoku.** Puzzles solved (%) on the hard test split of Agarwal et al. (2026), single sample.

Model	$N{=}1$	$N{=}2$	$N{=}4$	$N{=}8$
FMLM+, retrained	4.8	28.7	72.4	79.5
LDLM-FM	27.1	67.0	79.8	83.6
LDLM-FM-A	20.3	55.1	66.9	72.0

Table 8: **Adapted space, GSM8K.** Pass@1 (%) over five rounds on the 1000-problem training subset. Grey marks the adapted configuration.

Configuration	N=2	N=8	N=32
Teacher updates			
K=5	16.3	34.8	39.3
K=2	16.8	35.8	39.2
Encoder anchor			
none	16.3	34.8	39.3
flow map ψ_θ	15.6	35.4	39.6
teacher target	15.5	34.2	37.7
Encoder anneal			
100	16.4	35.2	38.8
300	16.3	34.8	39.3
1000	16.2	31.0	33.1
3000	12.9	25.3	27.0
5000	12.9	24.4	25.0

Table 9: **Adapted space, OpenWebText.** gPPL and diversity ($\times$100). Grey marks the adapted configuration.

	N=2		N=8	
Configuration	gPPL	Div.	gPPL	Div.
Teacher updates				
K=5	104.9	34.1	43.6	29.2
K=2	106.1	33.6	46.2	29.3
Encoder anchor				
none	555.4	48.8	249.9	40.4
flow map ψ_θ	104.9	34.1	43.6	29.2
Encoder anneal				
500	103.7	32.1	51.6	30.7
1000	112.1	34.0	48.0	29.2
3000	104.9	34.1	43.6	29.2
10000	97.6	32.5	40.9	27.9
20000	97.2	32.6	47.0	29.7

C SAMPLES

LDLM-FM, Sampling Steps: 1 Gen.PPL: **318.3** | Entropy: **5.63**

<|endoftext|>ing with infected animals corpses, in in Mexico to benze in Konankal Kud.
Other companies did the same partnerships as the Herschemical- Company.. well,, with a and powerful, feeling of aweness. Share your stories with our photos.
Common sense with American Agriculture
The Monsanto will the leaders for andizing an epidemic- pandaging proportions. Nor the EA might be well, the brave few g be are good examples of compassion. Especially for the citizens where theyhans are facing much direity in the outbreak crisis. News News
Alamia: Reonic C Outbreak Has Been Combined Over 10 Years Ago. Reonic Cide Containment on Alamia Suffing Haiti in 2014 2013 passed 4.5 Rations. Nationwidewide, the WHO of Agricultural Produignty Food outbreaks were sprayed across the world this year by anti-amines in the United States. The government were precautionial precautions for outbreaks at the.
The country began to comply with chemicals and . Biological resistance against the Reemic from the initial attacks on HFC outbreaks and pressure out the U.S. sought against the affected countries.
Following previous reports suggested a reover in Haiti in January and October. As in the Mid East. Once time the were released, a fully controlled flight around the ocean waters, "to their largest extent," including the early Alargal Kargal Kud was on the. .This year, 300+ deaths – and quite a 1,000 people – and required “a total of of deaths – have left Somalia. Gluonic nitoid, lack of food labels, was a HOPE. one of the land wide lists of drugs were called for. The ,000 were notifiable deaths in the country. All 300+ cases in a critical area were were 42 fatal convembered.
We are pond wonder, has never been wrong. They are wearing the inanhaph, and the hand over on their worst possible fears. Sadly for the outbreak has the anti-Sony corporation, an elite-wide narrative of epidemic outbreaks in Europe and the Aletheada calls for massive Alano-chemical shortages, and also love for the herbicide and use. Some with across the world and helped nurse Nancy Gayman died in March. An alliance with Monsanto executives has heard from the blame for calling the company to be world representing that and move on.
But the event is this week a two big in mind. Nintendo is offering new options. LEXUE release a game plan for HTCfall will have a few console releases available for the end this week, by downloadable info, site. Seriously what? Is the news and the hype?!? Your dad’s dedication to an effort to ruin their beloved home run, console LENENTOCK is on the future of Earth and making your controller, and surely something that will still be known for a child. Nintendo has happened been previously known as the North Series releasing “The Video Boy,” but have you had problems in your making lately? One time being in 2017 the console will only have multi- unaled console releases and for (the-LE to your andent Evil today? Definitely, that the console title brought Granew Valley is struggling for survival.. not currently developed franchise creator Carmian James Rodriguez, has lost greatly than it out as the original next release as well as 3 consoles that will have a mechanical and extremely expensive and most powerful to across the board spectrum. Western Games in able to add additional drugs to the flagship Good Company, with what are more complex systems, game details – no same is ever around the world. now (thank you your franchise creator). Write article is published below.**<|endoftext|><|endoftext|>**WHAT WAS AN exciting START CHONER NOW OCT\{}textbf{?} ON @ AND Facebook!
I’ve been in the thoroughly review our marketing materials taking stock of games for the console release of new products.
Today is waiting on YouTube channel The Marketing partners will want to continue. Go to Ede Shckel, both the International Center of Chronic Disease Research,, USC Labs and the EU CARES are Generally. free The packages are released long production times\{}dots{}by April day and they will expire the March 15th 2017 from As time as time the we’re available.
This week. people who order themselves are using an additional convenience code which includes a time “inclusion” with silver packaging and Customer items with Retail Packs. They can purchase a limited Retail Box (we tested in other cases).
Stock Display display with 9..5” and display slots will be still ready to apply a higher availability for. . Take multiple display slots with RGB, and . This was anticipated. a problem, but something in the packaging schedule has to be early on scheduled. it will work. a date approx 10 days.
THE ENDLY IN LI You just talked about**<|endoftext|>**

LDLM-FM, Sampling Steps: 2 Gen.PPL: **77.8** | Entropy: **5.50**

<|endoftext|>.
"Unless they are paying for more severe serious conditions, whether they are sick and are sick or the pre- just getting very ready for the fast lap’s commute," he said. "It was important for the sick to to the paid, to going to the and, and people care."
Deterts

GM had made a big boost since NASCAR grabbed an average 562 rating two years ago, when it increased profit margins and an earnings margin by more than 700,917,000,000 drivers – more than of companies' driving operations.
Sprint, 500 drivers made 9,000 million U.S.-mile trips and earning more than 100 percent in fiscal 2014 when pre-tax years began to close. Today, VW is paying an earnings of $107,000.
VW had the lowest-paid drivers in some and other fines – including $86,000 – last year.
"Housy work has a threat to the world and everyone is fighting for it," Maria Lohrke told The Business News. "The red is better."
Daniela Kirshzko founded the Wind Wind Group – a company of Wind Energy – to spin off as an owner in Atlanta.
[WSJ]**<|endoftext|>**US
effectively banned carbon dioxide emissions, taking subvention to power generation and other fossil fuels.
The C.C's super committee convened to America's historic tax debate on 18 December 2017, beginning with a series of protests in the global cities Monday morning. But after 2 degrees Celsius Fahrenheit on the slightly warmer day, the debate will reach Oct. 15th conclusion on the comprehensive regulatory framework for the year.
CO2 also continues to rise in US warming of nearly 17 years above 1990–an important global record of greenhouse-gas per unit2. The 2C is a mere 2 percent of the US electricity mix, significantly higher than a carbon dioxide-alission ratio of 1.5 percent of per fossil gas increase by 17 percent over the same period.
If climate declines ever lead to a sustained record, the feds disavow the efforts for a tax package to curb carbon pollution as a slam- blow.
The hit of low-carbon electricity has been a dire concern. Scientists say there is no greenhouse gas in the nation's electricity. The Smithsonian Institution of Sc Studies has warned that climate emissions could soon end, and the Obama warned that a tax increase will not reduce emissions from fossil fuels. The institute also said they are focusing on climate in an unprecedented way and the energy industry is struggling to end the carbon crisis. He suggested today that CO2 will cause more carbon levels to be absorbed to greenhouse gases as fossil fuel burning will lead-ause, which could affect the world's economy of human consumption.
Earlier last year, the administration announced a plan to reduce and decomarbonphasize coal standards.
Solar power is the first year to the focus liberalization of electricity regulations, to reduce the level of expensive single-use fuels. The administration has announced changes that work to also allow the federal government to gradually scale out further. Renewable plants are phased out annually each year, costing $10 billion less than $10 billion. Meanwhile, the renewable fuel nuclear fleet has been the result of a slew of industrial pollution: emissions of greenhouse gas levels increased more than half. When climate change (CO2) is "anomalously dangerous, the world to limiting fossil fuel emissions and an unprecedented risks.
Price declines in the late-2000s have been detrimental for power markets.
New York officials use the insult of climate for coal to undermine human health. So far that it is allowed to power. Hillary Pelosi recently responded to the energy crisis by nastily about " "sheltering"
insurance companies. The, said: "Without the impact of body-burning and cleanerable generation, carbon-generating electricity generation would be banned sufficiently to compensate for emissions they were using them within the end of the century."
The energy committee has convened a meeting today of the Nation's Energy Regulatory Administration) to investigate the continued rise in electricity consumption and emissions. The administration, House of Commonwealth-to-Hill (Bath.))bought) to launch an investigation into where coal power plants take place. The Radio and Energy Administration group conducted a multi-year study this year, concluding that the high prices are associated with cleaner and more efficient energy policies to reduce overall electricity consumption. If the coal price falls off, it should be avoided.
The electricity industry is in bad shape, and investors are suffering a lot of action. There are concerns that the National Energy Financial Conservation Administration (AMFC) has been interfering with that concern. A recent report last year found high levels the world consumption of power generation, including carbon-2, which calls for potential regulations to cope for increased climate change emissions. In 2015, the government pledged to reduce its funding by cutting capital financing programs to do $1**<|endoftext|>**

Figure 4: **Unconditional OpenWebText samples** from LDLM-FM at one and two steps, drawn from the operating points of Figure 3 and annotated with each sample's own generative perplexity and token entropy. Each is a full 1024-token generation, shown verbatim.

LDLM-FM, Sampling Steps: 4 Gen.PPL: **60.8** | Entropy: **5.28**

<|endoftext|> we must of the go function, easily encapsulated by the helpers class Slow :
lift-allpathing :: Slow; main :: data -> SoftState
just = file.map(Data, State :: SoftState, Id => {}, Int : Data -> Size -> SoftState })
main =
exose.getData().new(Go, { () => State, StateState })
But in order to gain our attention to it, we'll need to try to define a memological class. First, we need a generic class of Go, which contains no constructor that won't be the same as described by libraries.
On the other hand, the main function of go lists the available resources (Net) depending on your actions you perform. We also can add:
pre.map(points, points):
bind_size = 0, this.size, points, count 1, count 1)
Relude. need you at least explain how we implemented it Lickle-allending can be divided into three types of memological: points describing actions you want to be doing; points describing activities (intuing, etc.), and the four types (hot, cold, noisy, etc.). By default, you can use one of the memological classes to define a function of the lazy-allending instance, which shows the Haskell code in the simple, below example. (As you above, check the memological code in mobile apps the first time.) In any case, memological classes-ending is widely used in smartw languages as a means of delivering efficient, efficient, quick and efficient actions. We'll show the theory in terms of the goal of the be achieved with lazy-all–ending below.
The flow concept of lazy-ending in action
Although you can liven over the basic features of a path function, it isn't often quite easy to the core of us, but they\{}textbf{?}re unable to do a thorough analysis of that concept. In the example above, imagine that drugs that are usually the hottest and heaviest, versus the drugs that usually the most mildest. Instead, there's an interval test like this: Given that after one odd drink, it changes nothing more than 0. So if you can take these two values to values, the user will say, "ob nothing, g, n" only when the right values are sufficient, and if the test changes like this, "Wont's nothing, but no (no, no, n) no" in the data.
It's easy to use the measure in the sense that "the easiest thing to do" to no longer have with relative andness. Fold-allending also aims to move this concept into the realm of programming programming. Let's try out a generic form of routing.
Deep into a generic form of routing, let's look at the basics of composition. We'll consider a container pool in which an item can add a certain number of resources. A it may specify the different elements depending how large elements it can be, but you can translate them into the same resources we want to be global:
In a queue, a queue specifies that we can accept simple type values (strings, int arrays, integer power point numbers, etc.) without items being stashed in container elements; in a cluster, a can specify how large elements in a collection, relative to its required resource and state properties; in a cluster, a queue describes a given item remaining in the collection; in a queue, a queue describes the relative state of each item. Each item has a linear description of where it is located, relative to any required resource. In a cluster, the listing of an item is still in the binary, and so you may specify values relative to body properties (hot and not). Simpliness of these concepts is that different relationships between a collections overlap in size and complexity. Before we can categorize a collection to anything else, let's consider how different clusters represent a particular object, and assign certain common states to them:

global – that is, “real queue” or the “natural queue” in the “soft-allwflow queue;
soft queue – a queue, waiting for items to be grouped into portions of soft arrays and soft clusters. The advantage of lazy-ending is that it offers too many different states – the use of individual items in this case, for example, soft queue or “high-level queue” when applied to a single collection – but the only difference with soft queues is that you both forget the values and miss them entirely.
In my post discussed in this series, I’ll like to offer you as to how to see them all in practical terms. To skip ahead**<|endoftext|>**

LDLM-FM, Sampling Steps: 8 Gen.PPL: **44.1** | Entropy: **5.60**

<|endoftext|> regulations.**<|endoftext|>**Charles McClESPN (; June 18, 1947) is an American chef, businessman, and author.[5]
Former Representative of Texas, he was the Democratic State Senator of Texas. Besides serving as Democratic United States Senator, he served in two major presidential administrations, including Ronald Reagan and George H.W. Bush. McLESPN also pushed for stricter restrictions on refugees and foreign terrorists traveling into the US. In 1968, George F. Bush and Tim Sasse introduced their House bill, Zeta, which was passed in South Carolina. The bill would target suspected foreign terrorist organizations, and targeted the members of Islamic State or other terrorist groups. As Bush noted in 2006, "Such laws would reduce more than any foreign terrorist group in the state of Texas."[6]
1969 presidential cycle [edit]
Charles McLESPN lost a Senate race in the 1972 Republican presidential primary, losing to candidates Charles Hart and Jim Naughrey in the race against President-elect Walter Mondale. In the 1970s, McLicken embarked on large-scale farming in Montgomery County, Texas, eventually becoming the first rancher and congressman to serve in Congress.[7]
In 1970 McLicken lived his family in Austin, Texas, where he cultivated 10 million acres (8,000 acres in 2006).[8]
LESPN was noted for making his own diet of mixedillas: mashed potatoes, kale eggs, Thai chili peppers, eggs, and eggs for added weight. The potatoes are savy, cold, hard-cooked and often groan quickly at room temperature. However, a study found that it is activated by anysin "a protein in the gluccomoid pathway, which helps us grow without ever entering our digestive system." He was credited for his apparent mis-contribution of the finding that feeding all raw vegetables was another effective alternative which, if not successful, was to "kill the potatoes." A team of Harvard researchers has described such a diet as a viable and effective alternative.[10]
Ron Paul has repeatedly called it "an alternative," while Sen. Ron Paul and Governor Mitt Romney have stated with every day he would oppose the diet.[2]
Kale chicken in Wichita Falls, Texas, on April 24, 2015,
Charles McLESPN delivers a speech marking the 50th anniversary of dinner while entering the White House in 2009.
The family later settled near the island of New Zealand, which they described as Brother Sam.[3] They settled in Chapel Hill, North Carolina in 1973, in 1979 but they divorced,[2] in 2009.
Linner’s family’s development began in 1967 at the Keauga County High School in Chapel Hill, North Carolina, Texas, in 1984.
Biography [edit]
Charles was the chief advocate for the family for mixed potatoes that he was eating over lunch that day during dinner. He made an offer to give to his family for a teenage girl whom he had recently visited. Taking lunch at the school, the family said he would soon write a report to her family and a teacher in the school.
Charles McLESPN on his own vegetarian cooking plan.
He is widely known for his anti-vegan diet, his service paved the way for several recent contributions to his second son, Ronald A. Arappens.[12] After years as a Republican president, Arnold Arappes joined the Democratic Party after four years in the United States politics, according to the Los Angeles Times.
In 2007, the British journalist Thomas J. Rawnenoe described McLaughlin as a "vampire, pathological liar and fake prophet."[13]
Politico News issued a statement confirming, "Charles McLaughlin passed away today in Austin, Texas, 48 hours after his heart attack on January 28 2013. He was 89."[14][15]
The medical media, both food experts, and biostiologists have attributed Charles’ death to an alternative diet, with a Los Angeles Times declaring "Kenninner was a simple vegetarian diet and had "little choice but manual groaning."
In a 1998 US federal gag lawsuit filed by The Daily Mail, Johnson Soup Company refused to feed Charles McLicken for not completing the diet of potatoes during the mid-1940s. Another man, also from Long Island, died suddenly of cancer.
Linner is regarded as one of the most influential writers of all time, with The Daily Review of London also naming him as a key figure in winning three Academy Awards for best novel.[16]
He released animated film The Naked Man in which he talks about him being raped by an African American man who broke his leg, as well as claiming that the attackers would throw him out of their window. He ripped off his ears, tore his triceps and swathed his forearms in clear plastic weded to his forearms like a machine gun. He and other camera crew ran through the scene to ask if the young boy was hungry. "Did you know**<|endoftext|>**

Figure 5: **Unconditional OpenWebText samples** from LDLM-FM at four and eight steps, under the same protocol as Figure 4.

GSM8K solutions generated by LDLM-FM-A at $N{=}8$. Over the 128 evaluation rounds behind Figure 2, 1004 of the 1319 test problems ($76.1\,\%$) are solved in at least one round, while single-sample accuracy over these rounds is $31.8\,\%$. Only 20 are solved in every round and 315 in none. Most failures are therefore unreliability rather than incapacity, which is what repeated sampling recovers. Even the failures on never-solved problems are inconsistent: only 26 of them yield no runnable program in any round, and the remaining 289 execute in some rounds and not in others. The examples below are grouped by how a failing solution goes wrong: a program that runs to a wrong value, or no runnable program at all. All programs are shown verbatim.

Wrong value. Brett is 14 years old. In four years his sister Angela will be three times as old as he is now. How old is Angela right now? Gold answer 38. The failing sample adds four years to Brett before tripling, and the correct sample triples his current age.

Failing sample (executes to 50)

```
def simple_math_problem() -> int:
    """
    Brett is 14 years old.
    In four years his sister
    ↪ Angela will be three times
    ↪ as old as he is now.
    How old is Angela right now?
    """
    brett_age = 14
    brett_age_in_four_years =
    ↪ brett_age + 4
    angela_age_in_4_years = 3 *
    ↪ brett_age_in_four_years
    angela_age_now =
    ↪ angela_age_in_4_years - 4
    result = angela_age_now

    return result
```

Correct sample (executes to 38)

```
def simple_math_problem() -> int:
    """
    Brett is 14 years old.
    In four years his sister
    ↪ Angela will be three times
    ↪ as old as he is now.
    How old is Angela right now?
    """
    brett_age = 14
    angela_age_future = brett_age
    ↪ * 3
    angela_age_now =
    ↪ angela_age_future - 4
    result = angela_age_now

    return result
```

Does not execute. John arm wrestles 20 people. He beats 80%. How many people did he lose to? Gold answer 4. The failing sample stops after two assignments and never returns.

Failing sample (no output)

```
def simple_math_problem() -> int:
    '''
    John arm wrestles 20 people.

    He beats 80%.

    How many people did he lose
    ↪ to?
    '''
    total_people = 20
    beaten_percent = 80
```

Correct sample (executes to 4)

```
def simple_math_problem() -> int:
    '''
    John arm wrestles 20 people.

    He beats 80%.
     How many people did he lose
    ↪ to?
    '''
    total_people = 20
    beaten_percent = 80
    beaten_people = total_people
    ↪ * beaten_percent / 100
    lost_people = total_people -
    ↪ beaten_people
    result = lost_people
    return result
```

Systematic misreading. Marcus is half of Leo's age and five years younger than Deanna. Deanna is 26. How old is Leo? Gold answer 42. The sample reads "half of Leo's age" as halving rather than doubling, so the program runs but misreads the problem:

```
def simple_math_problem() -> int:
    """
    Marcus is half of Leo's age and five years younger than Deanna.
    Deanna is 26.
    How old is Leo?
    """
    deanna_age = 26 - 5
    leo_age = deanna_age / 2
    result = leo_age

    return result
```

Corrupted text. There are 50 deer in a field. 50 percent of them are bucks. 20 percent of the bucks are 8 points. How many 8 point bucks are there? Gold answer 5. The restatement of the question in the docstring is corrupted ("areato"), and the program ends after the first assignment:

```
def simple_math_problem() -> int:
```

```
    """
    There are 50 deer in a field.
    50 50 percent of them areato.
     20 percent of the the are 8 points.
     How many 8 point points are there?
    """
    total_deer = 50
```