LOOPHOLING ALL THE WAY: LANGUAGE MODELING WITH PURE SELF-CONDITIONING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent work proposes continuous flow language models (FLMs) to address the limitations of discrete diffusion models (DDMs): at every step, DDMs sample tokens independently, which hurts few-step generation, and discard the uncertainty of the denoiser's predictions (*information collapse*). Empirically, on reasoning tasks such as Sudoku and GSM8K, FLMs at best match standard DDMs and fall well behind DDMs with self-conditioning (SC). In particular, Loopholing DDMs (LDDMs) mitigate information collapse by propagating an SC state across steps, along the discrete tokens. While SC is traditionally seen as an auxiliary heuristic to improve performance, its empirical success makes us wonder whether **we could define a continuous generative model based purely on self-conditioning?** We answer affirmatively with LAW ("Loopholing All the Way"), which propagates the *expected embedding* under the denoiser's predictions across steps, decoding the tokens once, at the end. On hard Sudoku, LAW comes close to LDDMs (90.4% vs. 92.7%), and on Maze, it is slightly more accurate than all baselines with few steps. On TinyGSM, fine-tuning a pre-trained Duo checkpoint with LAW increases the GSM8K accuracy from 18.6% to 55.2%. On OpenWebText, trained from scratch, LAW matches or improves over $\mathbb{S}$-FLM with the standard DiT architecture.

1 INTRODUCTION

Although autoregressive models (ARMs) dominate language modeling (Vaswani et al., 2023; Kaplan et al., 2020; Meta, 2024; Gemini Team et al., 2025), their generation process is inherently sequential and their left-to-right factorization can be limiting for tasks that need bidirectional context (Papadopoulos et al., 2024; Kitouni et al., 2024; Nagarajan et al., 2025). Discrete diffusion models (DDMs) (Austin et al., 2023; Campbell et al., 2022; Sahoo et al., 2024; Lou et al., 2024; Ou et al., 2024; Shi et al., 2025; Sahoo et al., 2025; Schiff et al., 2025; von Rütte et al., 2025) instead use a bidirectional architecture and can refine tokens in parallel. For tractability, DDMs sample each position independently, and discard the denoiser predictions after sampling, which causes two problems. First, the per-position factorization hurts the few-steps generation performance (Deschenaux & Gulcehre, 2024; Feng et al., 2025). Secondly, discarding previous predictions means that the associated uncertainty is lost (*information collapse*; Jo et al., 2025).

On the other hand, continuous diffusion language models (Dieleman et al., 2022; Li et al., 2022; Gulrajani & Hashimoto, 2023; Jo & Hwang, 2025; Batzolis et al., 2026; Chemseddine et al., 2026; Chen et al., 2026; Deschenaux & Gulcehre, 2026; Hu et al., 2026; Lee et al., 2026a; Potaptchik et al., 2026; Roos et al., 2026; Shen et al., 2026; Suleymanzade et al., 2026; Yang et al., 2026; Yoo et al., 2026) avoid the factorized sampling by integrating an ODE or SDE from noise to data (Lipman et al., 2023; Liu et al., 2022a; Albergo & Vanden-Eijnden, 2023). However, they tend to match standard DDMs on reasoning. For example, FLM (Lee et al., 2026a) and $\mathbb{S}$-FLM (Deschenaux & Gulcehre, 2026) remain close to MDLM and Duo on Sudoku and Maze, and $\mathbb{S}$-FLM matches MDLM on GSM8K only with greedy generation.

In contrast, self-conditioning (SC; Chen et al., 2022; Strudel et al., 2022) substantially improves the performance of diffusion language models. Loopholing Discrete Diffusion Models (LDDMs; Jo et al., 2025) and DiffusionGemma (Google DeepMind, 2026) feed the previous predictions or hidden representations of the denoiser back as input, and Rozonoyer et al. (2026) train with self-

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.

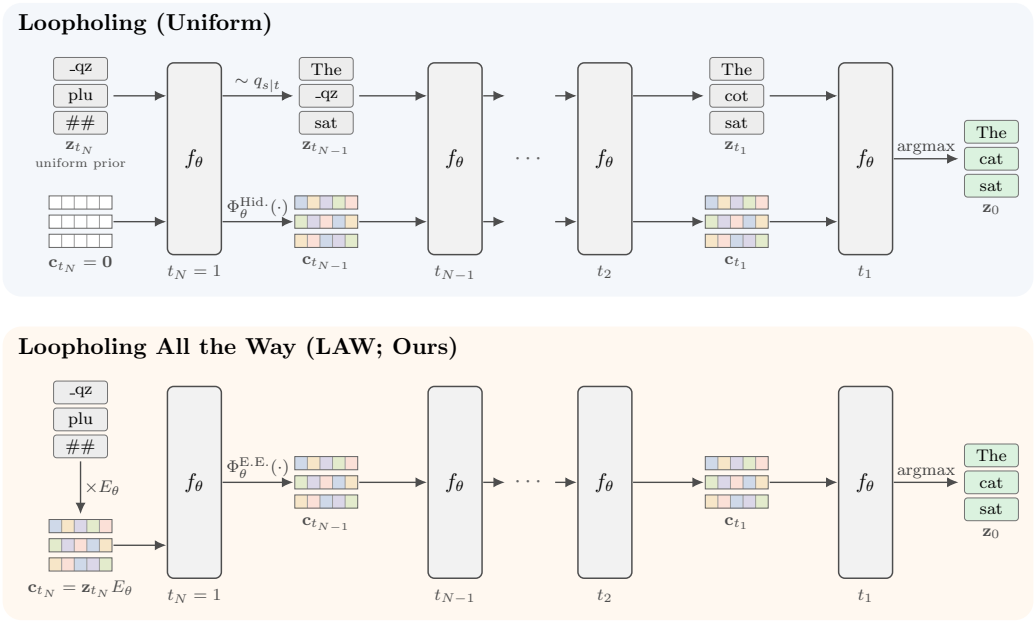


Figure 1: **Overview.** Sampling runs from left to right, applying the denoising backbone f_θ at every step. Boxed words are discrete tokens, colored cells SC states, white cells zero-valued states, and ellipses $(\cdots)$ omitted steps. *Top: Loopholing (LDDM; Jo et al., 2025).* Each generation step samples discrete tokens and passes them to the next step alongside an SC state (5). *Bottom: LAW.* Each generation step forwards only an SC state, computed from the denoiser's hidden representation by the state map $\Phi_\theta^{\text{E.E.}}$ (7). The initial tokens are drawn from the uniform prior, and we decode back to discrete tokens only at the very end (9).

conditioning using truncated backpropagation through time. On hard Sudokus, LDDMs solve 92.7% of the puzzles, against 58.4% for Duo and at most 45.0% for FLMs (Sec. 4.1). So far, despite its empirical success, SC always plays an auxiliary role. This makes us wonder:

"Can we define a generative model based purely on self-conditioning?"

We answer this question positively with **"Loopholing All the Way"** (LAW, Figure 1).

Contributions. **(1)** We propose LAW, **a continuous generative model based purely on self-conditioning**. Like uniform diffusion models (UDMs), LAW starts from random tokens. Unlike (L-)DDMs, LAW propagates only the *expected embedding* under the denoiser's prediction across steps. It thus retains the denoiser's uncertainty: each step uses the model's predictions from the previous step, without sampling discrete tokens in between. Unlike continuous flows, LAW requires no Gaussian corruption or ODE solver. **(2)** The choice of propagated state is crucial. Passing the final-layer hidden representation, as in LDDMs, performs worse than the expected embedding (Suppl. C.3). Furthermore, perturbing the SC state stabilizes training, and we propose *temperature regularization* (12) as a simple way to do so. **(3)** Empirically, LAW performs similarly to LDDMs on hard Sudoku (90.4% vs. 92.7%) and is slightly ahead with few steps on Maze (Sec. 4.3). On TinyGSM, fine-tuning a pre-trained Duo checkpoint for $\sim$20k steps on a single GPU increases the GSM8K accuracy from 18.6% to 55.2% (Sec. 4.2), with 46.0% in as few as 16 steps (i.e. *Number of Function Evaluations*; NFE). On OpenWebText, trained from scratch, LAW reaches a lower Gen. PPL than diffusion and flow baselines at low entropy with 1024 NFE, although MDLM, Duo, and $\mathbb{S}$-FLM remain ahead at higher entropy (Sec. 4.4).

2 BACKGROUND

Notation. We denote by $\mathcal{V}$ a finite vocabulary of size $|\mathcal{V}|$ and identify each token with a one-hot vector in $\{0,1\}^{|\mathcal{V}|}$. A sequence $\mathbf{x} \in \mathcal{V}^L$ consists of L tokens, with $\mathbf{x}^\ell$ as its ℓ-th element. We write $\Delta^{|\mathcal{V}|-1}$ for the probability simplex, $\mathrm{Cat}(\cdot\,;\mathbf{p})$ for the categorical distribution with parameter $\mathbf{p} \in \Delta^{|\mathcal{V}|-1}$, and $\mathbf{1}$ for the all-ones vector. The denoiser $\mathbf{x}_\theta : \mathcal{V}^L \times [0,1] \to (\Delta^{|\mathcal{V}|-1})^L$ predicts a per-position distribution over the vocabulary. We write $E_\theta \in \mathbb{R}^{|\mathcal{V}| \times d}$ for the learned embedding matrix, where d is the embedding dimension. A sequence of tokens $\mathbf{z} \in \{0,1\}^{L \times |\mathcal{V}|}$ is embedded as $\mathbf{z}E_\theta \in \mathbb{R}^{L \times d}$. We write $\mathrm{sg}[\cdot]$ for the stop-gradient operator and $\mathrm{LN}(\cdot)$ for LayerNorm (Ba et al., 2016).

2.1 UNIFORM DIFFUSION MODELS (UDMS)

DDMs define a fixed *forward* process that gradually corrupts clean tokens into noise and learn to reverse it (Austin et al., 2023; Campbell et al., 2022). We study the *uniform* variant, whose forward process corrupts each token independently towards a uniform draw over the vocabulary (Hoogeboom et al., 2021; Lou et al., 2024; Schiff et al., 2025), as in Duo (Sahoo et al., 2025). The noise level at time t is determined by α_t, where $\alpha : [0,1] \to [0,1]$ is a monotonically decreasing schedule with $\alpha_0 \approx 1$ and $\alpha_1 \approx 0$. The noisy tokens are sampled from the forward process, $\mathbf{z}_t \sim \prod_{\ell=1}^{L} q_t(\cdot \mid \mathbf{x}^\ell)$, with marginal

$$q_t(\mathbf{z}_t^\ell \mid \mathbf{x}^\ell) = \mathrm{Cat}\Big(\mathbf{z}_t^\ell;\ \alpha_t\,\mathbf{x}^\ell + (1-\alpha_t)\tfrac{1}{|\mathcal{V}|}\mathbf{1}\Big)\,, \tag{1}$$

so $\mathbf{z}_t^\ell$ remains clean with probability α_t and is otherwise resampled uniformly from the vocabulary. The denoiser $\mathbf{x}_\theta(\mathbf{z}_t, t)$ composes the embedding matrix E_θ, a stack of Transformer blocks f_θ, and an affine projection g_θ (the language-modeling head), followed by a softmax:

$$\underbrace{\mathbf{z}_t}_{\mathcal{V}^L} \xmapsto{\cdot\, E_\theta} \underbrace{\mathbf{z}_t E_\theta}_{\mathbb{R}^{L\times d}} \xmapsto{f_\theta(\cdot,\,t)} \underbrace{\mathbf{h}_t}_{\mathbb{R}^{L\times d}} \xmapsto{g_\theta} \underbrace{g_\theta(\mathbf{h}_t)}_{\mathbb{R}^{L\times|\mathcal{V}|}} \xmapsto{\text{softmax}} \underbrace{\mathbf{x}_\theta(\mathbf{z}_t,t)}_{(\Delta^{|\mathcal{V}|-1})^L}, \tag{2}$$

where $\mathbf{h}_t$ is the final-layer hidden representation. Given clean sequences $\mathbf{x} \sim q_{\text{data}}$, the denoiser is trained to minimize the diffusion *Negative Evidence Lower Bound* (NELBO), an upper bound on the expected negative log-likelihood that decomposes into per-token losses $\mathcal{L}_{\text{tok}}$,

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x}\sim q_{\text{data}},\, t\sim\mathcal{U}[0,1],\, \mathbf{z}_t\sim q_t(\cdot|\mathbf{x})}\left[\sum_{\ell=1}^{L} \mathcal{L}_{\text{tok}}\big(\mathbf{x}_\theta^\ell(\mathbf{z}_t,t),\,\mathbf{x}^\ell,\,\mathbf{z}_t^\ell,\,\alpha_t\big)\right], \tag{3}$$

with the closed forms of $\mathcal{L}_{\text{tok}}$ and $q_{s|t}$ in Suppl. A.1 (Sahoo et al., 2025). To generate, UDMs plug $\mathbf{x}_\theta$ into the closed-form categorical *denoising posterior* $q_{s|t}(\mathbf{z}_s^\ell \mid \mathbf{z}_t^\ell, \mathbf{x}^\ell)$, $s < t$, in place of $\mathbf{x}^\ell$. Let $t_n = n/N$, $n = 0,\ldots,N$, be the uniform sampling time grid, where N is the number of function evaluations (NFE). The initial tokens $\mathbf{z}_1$ are drawn from the uniform prior $\mathrm{Cat}(\cdot\,;\tfrac{1}{|\mathcal{V}|}\mathbf{1})$, and $q_{s|t}$ is then applied N times (ancestral sampling):

$$\mathbf{z}_{t_{n-1}}^\ell \sim q_{t_{n-1}|t_n}\Big(\mathbf{z}_{t_{n-1}}^\ell \;\Big|\; \mathbf{z}_{t_n}^\ell,\ \mathbf{x}_\theta^\ell(\mathbf{z}_{t_n},t_n)\Big)\,, \qquad n = N,\ldots,1. \tag{4}$$

2.2 LOOPHOLING DISCRETE DIFFUSION MODELS (LDDMS)

The sampling step (4) only propagates the discrete tokens $\mathbf{z}_s$ to the next step. Thus, the predictions $\mathbf{x}_\theta(\mathbf{z}_t,t)$ and their uncertainty are discarded after sampling (*information collapse*). LDDMs (Jo et al., 2025) propagate a normalized version of the final-layer hidden representation $\mathbf{h}_t$ (2) across sampling steps, alongside the sampled tokens. Starting from $\mathbf{c}_1 = \mathbf{0}$, one LDDM sampling step from time t to s maps the pair $(\mathbf{z}_t, \mathbf{c}_t)$ of tokens and self-conditioning (SC) state to $(\mathbf{z}_s, \mathbf{c}_s)$:

$$\underbrace{(\mathbf{z}_t,\,\mathbf{c}_t)}_{\mathcal{V}^L\times\mathbb{R}^{L\times d}} \xmapsto{f_\theta(\mathbf{z}_t E_\theta + \mathbf{c}_t,\ t)} \underbrace{\mathbf{h}_t}_{\mathbb{R}^{L\times d}} \xmapsto{\left(q_{s|t}\,\circ\,\mathrm{softmax}\,\circ\, g_\theta,\ \Phi_\theta^{\text{Hid.}}\right)} \underbrace{(\mathbf{z}_s,\,\mathbf{c}_s)}_{\mathcal{V}^L\times\mathbb{R}^{L\times d}}, \tag{5}$$

where the last transition samples $\mathbf{z}_s$ from the posterior $q_{s|t}(\cdot \mid \mathbf{z}_t,\ \mathrm{softmax}(g_\theta(\mathbf{h}_t)))$ and computes the SC state $\mathbf{c}_s = \Phi_\theta^{\text{Hid.}}(\mathbf{h}_t)$, with $\Phi_\theta^{\text{Hid.}}(\mathbf{h}) = \mathrm{LN}(\mathbf{h})$. LDDMs train this propagation with

SC (Chen et al., 2022): on 90% of the training iterations, a first forward pass takes $(\mathbf{z}_t, \mathbf{0})$ as input to produce the SC state $\mathbf{c}_t = \mathrm{sg}\big[\Phi_\theta^{\mathrm{Hid.}}(\mathbf{h}_t)\big]$, and only a second pass, taking $(\mathbf{z}_t, \mathbf{c}_t)$, minimizes the NELBO (3). On the remaining 10%, LDDMs use a standard UDM training iteration with $\mathbf{c}_t = \mathbf{0}$. During training, $\mathbf{c}_t$ is thus computed at the same time t, whereas during sampling it comes from the previous, noisier step. Despite this slight train–test mismatch, the approach works well empirically (Jo et al., 2025, and Sec. 4).

3 Loopholing All the Way (LAW)

In LDDMs, SC only complements the discrete tokens, which are still sampled at every step (Sec. 2.2). Given the gains of SC, we ask whether it can define a generative model on its own. As such, LAW is a *pure self-conditioning* model: it propagates only the SC state across steps and decodes tokens once, after the final step. Since the SC state is the only input of the denoiser, we use the *expected embedding* under the denoiser's prediction, which stays in the space of token embeddings. We present the generative process first (Sec. 3.1), since it motivates the training procedure (Sec. 3.2).

3.1 Generative Process

Initialization. We obtain the initial SC state $\mathbf{c}_1$ by embedding tokens drawn from the uniform prior:

$$\mathbf{c}_1 = \mathbf{z}_1 E_\theta, \qquad \mathbf{z}_1^\ell \sim \mathrm{Cat}\left(\cdot\,; \frac{1}{|\mathcal{V}|}\mathbf{1}\right). \tag{6}$$

Continuous State Updates. For $n = N, \dots, 2$, we pass the SC state $\mathbf{c}_{t_n}$ through the denoising backbone and map the resulting hidden representation into the next state:

$$\underbrace{\mathbf{c}_{t_n}}_{\mathbb{R}^{L\times d}} \xmapsto{f_\theta(\cdot, t_n)} \underbrace{\mathbf{h}_{t_n}}_{\mathbb{R}^{L\times d}} \xmapsto{\Phi_\theta^{\mathrm{E.E.}}(\cdot)} \underbrace{\mathbf{c}_{t_{n-1}}}_{\mathbb{R}^{L\times d}}. \tag{7}$$

The backbone receives the SC state alone, so no tokens are sampled between steps. LAW uses the *expected embedding* (E.E.) under the denoiser's prediction:

$$\underbrace{\Phi_\theta^{\mathrm{E.E.}}(\mathbf{h}; \tau)}_{\in \mathbb{R}^{L\times d}} = \underbrace{\mathrm{softmax}\left(\frac{g_\theta(\mathbf{h})}{\tau}\right)}_{\in \mathbb{R}^{L\times |\mathcal{V}|}} \underbrace{E_\theta}_{\in \mathbb{R}^{|\mathcal{V}|\times d}}, \tag{8}$$

where the softmax is applied over the vocabulary dimension and $\tau > 0$ is a temperature. $\Phi_\theta^{\mathrm{E.E.}}$ computes, at each sequence position, a linear combination of token embeddings weighted by the token probabilities predicted by the denoiser. In contrast, LDDMs (Jo et al., 2025) carry $\Phi_\theta^{\mathrm{Hid.}}(\mathbf{h})$ (Sec. 2.2), which is added to the token embeddings (5), whereas the SC state of LAW replaces them. The temperature τ is randomized during training as a regularizer (Sec. 3.2). At sampling time we use $\tau = 1$. This is distinct from the sampling temperature T, which we vary in Sec. 4.4 and which only scales the logits of the final decoding step (9).

Decoding. Let $\mathbf{p} = \mathrm{softmax}\big(g_\theta(f_\theta(\mathbf{c}_{t_1}, t_1))/T\big)$ denote the last denoiser prediction, scaled by the sampling temperature T ($T = 1$ unless stated otherwise). A decoder η maps the prediction to discrete tokens. We consider a *raw* and an *argmax* decoder:

$$\eta_{\mathrm{raw}}(\mathbf{p})^\ell \sim \mathrm{Cat}\big(\cdot\,; \mathbf{p}^\ell\big), \qquad \eta_{\max}(\mathbf{p})^\ell = \arg\max_{v\in\mathcal{V}} p_v^\ell. \tag{9}$$

With the argmax decoder, besides the initial draw, sampling is deterministic, so all randomness comes from the prior. For conditional generation, we fix the prompt tokens to their clean embeddings and only generate the suffix. See Algo. 1 for the pseudocode.

3.2 Training the Continuous Trajectory

During sampling, the denoiser consumes SC states that it produced itself (7). Since backpropagating through the whole trajectory is infeasible, we start from noisy tokens $\mathbf{z}_t$ (1) and unroll the SC

recursion M times. We train with the NELBO (3), to enable the fine-tuning of pre-trained UDM checkpoints. With $\mathbf{x}_\theta(\mathbf{c}, t) = \mathrm{softmax}\big(g_\theta(f_\theta(\mathbf{c}, t))\big)$ the prediction from an SC state, the recursion is:

$$\mathbf{c}^{(1)} = \mathbf{z}_t E_\theta, \qquad \mathbf{c}^{(i+1)} = \mathrm{sg}\Big[\Phi_\theta^{\mathrm{E.E.}}\big(f_\theta(\mathbf{c}^{(i)}, t_i);\, \tau\big)\Big], \qquad t_i = t - (i-1)\,\delta, \tag{10}$$

where $\delta = 1/N$ is inversely proportional to the maximal number of sampling steps (e.g. 1024 for a sequence of 1024 tokens). Counterintuitively, the stop-gradient sg between passes can stabilize training, e.g., on OpenWebText (Suppl. A.6 and Suppl. C.8). Each pass consumes the SC state of the previous pass, and we average the NELBO (3) over all passes:

$$\mathcal{L}_{\mathrm{LAW}}(\theta) = \mathbb{E}_{\mathbf{x}\sim q_{\mathrm{data}},\, t,\, \mathbf{z}_t\sim q_t(\cdot|\mathbf{x}),\, \tau}\left[\frac{1}{M}\sum_{i=1}^{M}\sum_{\ell=1}^{L}\mathcal{L}_{\mathrm{tok}}\Big(\mathbf{x}_\theta^\ell\big(\mathbf{c}^{(i)}, t_i\big),\ \mathbf{x}^\ell,\ \mathbf{z}_t^\ell,\ \alpha_t\Big)\right]. \tag{11}$$

Only the time conditioning changes across the M passes. The denoiser predictions are penalized against the same $\mathbf{z}_t$ and time t. We sample $t \sim \mathcal{U}[(M-1)\delta, 1]$ to ensure $t_M \geq 0$.

Regularization. Training on self-produced continuous states can be unstable. When fine-tuning pre-trained checkpoints, the GSM8K accuracy degrades after its peak (Sec. 4.2). To stabilize training, both when fine-tuning and when training from scratch, we perturb the SC state $\mathbf{c}^{(i+1)}$ (10) before the next pass. We propose **temperature regularization**, which randomly samples the temperature τ of $\Phi_\theta^{\mathrm{E.E.}}$ during training. The loss is computed on the untempered predictions $\mathbf{x}_\theta(\mathbf{c}^{(i)}, t_i)$ (11); τ only scales the logits used to compute the next SC state. The temperature τ is shared per sequence and sampled as

$$\tau \sim \tfrac{1}{2}\,\mathcal{U}\big(\rho^{-1}, 1\big) + \tfrac{1}{2}\,\mathcal{U}(1, \rho)\,. \tag{12}$$

The factor $\rho \geq 1$ sets the range of τ, which is equally likely to sharpen ($\tau < 1$) or flatten ($\tau > 1$) the SC state. Since the perturbation is symmetric around $\tau = 1$, the denoiser cannot undo it with a constant rescaling of its logits. We also compare with *Gaussian* and *spherical perturbations* of the SC state, described in Suppl. A.7.

Architecture. LAW uses the same denoiser architecture as standard UDMs, a Diffusion Transformer (DiT) with pre-normalization (Peebles & Xie, 2023; Lou et al., 2024), so it can be trained from scratch or fine-tuned from a pre-trained UDM checkpoint. For language modeling on OpenWebText (Sec. 4.4), we also experiment with Sandwich-Swin blocks (Liu et al., 2022b; Geiping et al., 2025), which additionally normalize the output of each attention and MLP sublayer before adding it back to the residual stream. We also optionally add a learned *mode embedding* to the time conditioning, which indicates whether the denoiser receives the embeddings of noisy tokens or an SC state. See Suppl. A.5 and Suppl. A.8 for details.

4 Experiments

We evaluate LAW on four benchmarks, each assessing a different capability: Sudoku (Sec. 4.1) tests global constraints with a unique solution, GSM8K (Cobbe et al., 2021) (Sec. 4.2) reasoning on school math word problems, Maze (Sec. 4.3) long-range planning with possibly several optimal paths, and OpenWebText (Sec. 4.4) unconditional language modeling. Full details, including the samplers and decoders of all methods, are in Suppl. B.

4.1 Reasoning on Sudoku

Experimental Setup. We follow prior work (Kim et al., 2025) and generate 48k training and 2k validation puzzles with unique solutions (Alp, 2024). We consider three difficulty levels, determined by the number of visible digits (easy: $40/81$, medium: $35/81$, hard: $30/81$), and we report the *exact-match* accuracy. The denoiser is implemented with an 8-layer DiT with Rotary Positional Encoding (RoPE; Su et al., 2023), and all methods are trained from scratch. We provide full experimental details in Suppl. B.3.

FLOP-Matched Training. We match the training FLOPs of LAW, LDDMs, and the discrete diffusion baselines. Following the standard rule that a backward pass costs about twice a forward pass, if a forward pass costs F, a standard UDM update costs about $3F$. LDDMs add a forward pass without gradients on a fraction p of the updates (Sec. 2.2), for an average cost of $(3+p)F$. Therefore, we train LDDMs for $3/(3+p)$ as many updates. With $p = 0.9$, $3/(3+p) \approx 0.769$. When training LAW with M passes, each update costs about $3MF$, since each pass requires a forward and a backward computation, with or without a stop-gradient between passes. To match the FLOP budget, LAW processes $1/M$ as many training examples as the baseline: it always trains for fewer updates, and for $M \geq 4$ possibly with a smaller batch (Table 3).

Table 1: *Exact-match* accuracy (%) on Sudoku with 180 NFE. Baseline numbers are taken from prior work (Deschenaux & Gulcehre, 2026). Best per column in **bold**, second best underlined.

Model	Easy	Med.	Hard
Autoregressive			
Sample	13.9	5.1	0.6
Greedy	14.6	5.1	1.0
Discrete Diffusion			
MDLM (Sahoo et al., 2024)	92.0	77.1	30.2
Duo (Sahoo et al., 2025)	96.3	84.7	58.4
LDDM Masked (Jo et al., 2025)	98.3	95.8	85.2
LDDM Uniform (Jo et al., 2025)	**99.5**	**98.8**	**92.7**
Continuous Models			
CANDI (Pynadath et al., 2025)	79.3	45.9	16.7
CANDI pure Gaussian	63.9	41.9	12.5
FLM (Lee et al., 2026a)	94.2	82.7	44.5
$\mathbb{S}$-FLM Deschenaux et al. (2026)	94.8	85.2	45.0
*LAW (**Ours**), $M = 2$*			
no regularization	99.2	97.0	88.6
temperature reg. ($\rho = 3$)	97.3	95.6	90.4

Results. Table 1 shows that with 180 NFE, LAW achieves 90.4% accuracy on hard Sudoku puzzles, compared to 45.0% for $\mathbb{S}$-FLM (Deschenaux & Gulcehre, 2026). Additionally, it outperforms MDLM and Duo with ancestral sampling. Although LDDMs perform slightly better, LAW remains competitive while decoding only at the final step. We ablate the number of passes, the FLOP-matching and the temperature regularization in Table 3. In this sweep, $M = 2$ tends to perform best at a fixed FLOP budget.

4.2 Reasoning on GSM8K

Experimental Setup. We train on TinyGSM (Liu et al., 2023), a dataset of 11.8M synthetic math problems, and evaluate on the GSM8K test set (Cobbe et al., 2021). For each test problem, we generate and execute one Python program. We use a 12-layer DiT denoiser and the `smollm` tokenizer (Allal et al., 2025), with a context length of 512 and padding for shorter programs. All baselines are trained from scratch for 250k steps with a batch size of 512, except LDDM, which is trained for 192k steps to match the training FLOPs (Sec. 4.1). We consider two settings for LAW. First, we fine-tune it from the Duo baseline checkpoint (250k steps) for 50k steps on a single H200 GPU with a batch size of 64. Second, we train it from scratch with the same training FLOPs as the baselines, i.e., 125k steps with $M = 2$ (Suppl. C.4)[1]. We provide full experimental details in Suppl. B.4.

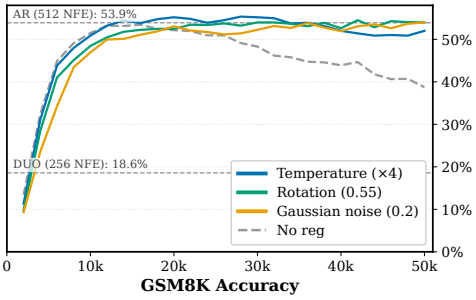
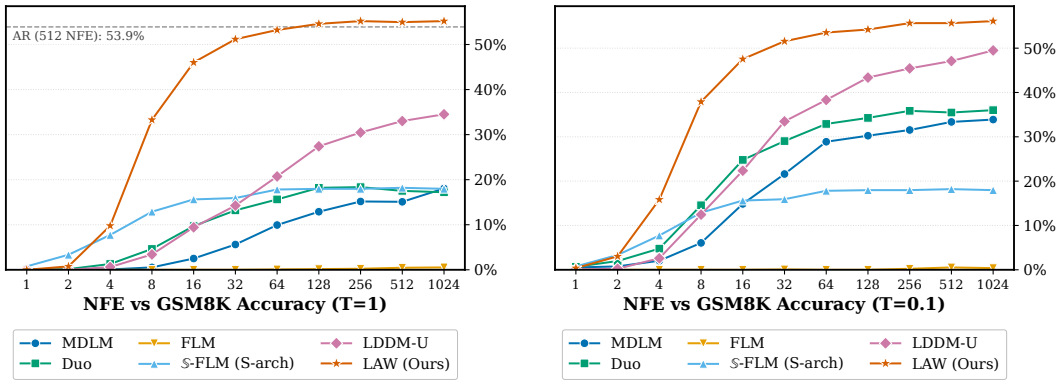


Figure 2: **GSM8K accuracy over fine-tuning steps.** Best setting of each regularizer of Sec. 3.2, against an unregularized run. All curves are evaluated with 256 NFE, and the horizontal line is Duo (Sahoo et al., 2025) with 256 NFE, at 18.6%.

Regularization. Without regularization, the accuracy of LAW first increases and then declines (Figure 2), which suggests overfitting. With the regularizers, the accuracy instead saturates after about 20k steps. We sweep the temperature range $\rho \in \{2, 3, 4, 5\}$ (12), as well as the Gaussian noise scale $\sigma \in \{0.05, 0.1, 0.2, 0.25\}$ and the rotation fraction $\gamma \in \{0.3, 0.5, 0.55, 0.6\}$ of the regularizers in Suppl. A.7. The figure shows the best setting of each regularizer, with the full sweeps in Suppl. C.2. Increasing the dropout probability is ineffective (Suppl. C.2).

[1]Fine-tuning the autoregressive, Duo, and MDLM baselines for an additional 50k steps on a single GPU substantially reduced their accuracy. We therefore report their results after 250k steps of regular training.



(a) Sampling temperature $T = 1$.

(b) Sampling temperature $T = 0.1$.

Figure 3: **GSM8K accuracy against NFE** at sampling temperature $T = 1$ (left) and $T = 0.1$ (right). LAW is fine-tuned from the Duo checkpoint with temperature regularization ($\rho = 4$) and evaluated after 20k fine-tuning steps. The baselines are trained from scratch for 250k steps, except LDDM-U, which is trained for 192k steps to match the training FLOPs. $\mathbb{S}$-FLM ($\mathbb{S}$-arch.) uses the top-1 velocity sampler (Suppl. B.2) The dashed line in (a) is the AR model with sampling (53.9%, 512 NFE).

Number of Function Evaluations. Figure 3 compares LAW, fine-tuned for 20k steps with temperature regularization ($\rho = 4$), with the baselines from 1 to 1024 NFE, at sampling temperatures $T = 1$ and $T = 0.1$. LAW needs few steps: At $T = 1$, its accuracy increases from 33.3% at 8 NFE to 51.2% at 32 NFE, and saturates around 55% from 128 NFE onward. From 128 NFE, LAW is more accurate than the AR model with sampling (53.9% with 512 NFE), although it remains below AR with greedy decoding (63.3%). With only 16 NFE at $T = 1$, LAW is already more accurate than previous diffusion and flow baselines using 1024 NFE. Among the baselines, LDDM-U is the most accurate, with 34.5% at $T = 1$ and 49.5% at $T = 0.1$ with 1024 NFE, followed by Duo and MDLM. $\mathbb{S}$-FLM ($\mathbb{S}$-arch.) saturates below 20% after 16 NFE, and FLM stays close to 0% at both temperatures. Lowering the sampling temperature strongly improves the discrete diffusion baselines, whereas the accuracy of LAW changes little.

Training from Scratch. We also train LAW from scratch with the same training FLOPs as LDDM-U and LDDM-M (Suppl. C.4). At $T = 1$, LAW is on par with or ahead of LDDM-M up to 256 NFE, and reaches 30.8% at 1024 NFE, compared to 32.3% for LDDM-M and 34.5% for LDDM-U. At $T = 0.1$, it reaches 42.3% at 1024 NFE, compared to 48.7% and 49.5%, respectively. Hence, when trained from scratch, pure self-conditioning comes close to self-conditioned DDMs but performs worse, and on GSM8K, fine-tuned from a pre-trained UDM, LAW is the most accurate diffusion/flow model.

4.3 Path finding on Maze

Experimental Setup. We evaluate shortest-path finding on mazes inspired by Maze-Hard (Wang et al., 2025). We generate 1k training and 1k evaluation mazes on 20×20 grids (Lehnert et al., 2024) and keep only mazes whose shortest path spans at least 71 cells. Each cell is a `wall`, `empty`, `start`, `goal`, or `path` token, and, as for Sudoku, each sequence concatenates the unsolved grid with a corrupted grid (800 tokens). While mazes can have several shortest paths, Wang et al. (2025) check whether the generated path matches the one marked in the target. Instead, we report the *valid-solution* accuracy: the fraction of mazes for which the generated grid preserves the maze and marks any shortest path from `start` to `goal` (Suppl. B.5; exact match in Suppl. C.5). All methods use an 8-layer DiT ($d = 256$, 8 heads, RoPE) trained from scratch with the same FLOP budget (Sec. 4.1; details in Suppl. B.5).

Results. LAW is the most accurate model in few steps: from 2 to 64 NFE, it outperforms all baselines (Figure 4). Only with 128 NFE or more does LDDM-M overtake it, by at most 2.8 points

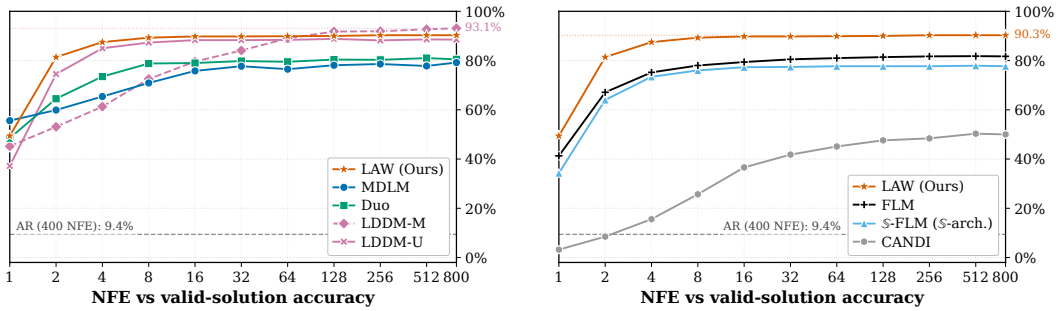


Figure 4: **Maze-Hard *valid-solution* accuracy against NFE.** Fraction (%) of the 1,000 evaluation mazes for which a valid solution is generated (Suppl. B.5). All methods are trained with matched budget following Sec. 4.1. LAW uses $M = 4$ passes, batch size 128 and 50k training steps and decodes with η_{raw}. The gray dashed line is the AR model with greedy decoding (9.4%). **Left:** LAW compared to discrete diffusion baselines. **Right:** LAW compared to continuous diffusion baselines.

(93.1% vs. 90.3% at 800 NFE). In detail, LAW ($M = 4$, batch size 128, 50k training steps) reaches 90.3% from 256 NFE onward. Among discrete diffusion models, LDDM-U stays 1–2.5 points below LAW from 4 NFE onward, while LDDM-M needs 64 NFE to match the accuracy LAW reaches with far fewer steps. Duo and MDLM are below LAW from 2 NFE onward. Among the continuous baselines, the most accurate is FLM, with 81.8% at 512 NFE.

4.4 Language Modeling

Experimental Setup. We train on OpenWebText (Gokaslan & Cohen, 2019) with the GPT-2 tokenizer (Radford et al., 2019) and a context length of 1024, at global batch size 512 with the optimizer configuration of Suppl. B.1. We compare to AR, MDLM (Sahoo et al., 2024), Duo (Sahoo et al., 2025), FLM (Lee et al., 2026a), $\mathbb{S}$-FLM and $\mathbb{S}$-FLM ($\mathbb{S}$-arch.) (Deschenaux & Gulcehre, 2026), all trained from scratch for 400k steps. We train LAW from scratch with $M = 2$ passes (10) for 200k steps, which matches the training FLOPs of the baselines. The denoiser is a 12-layer DiT with embedding dimension $d = 768$ and 12 attention heads. The reported LAW model additionally uses Sandwich-Swin blocks (Suppl. A.5), the mode embedding (Suppl. A.8), and temperature regularization with $\rho = 3$ (12), and we ablate these choices in Suppl. C.8.

Gen. PPL / Entropy Frontier. Repetitive generations can improve the Gen. PPL without improving sample quality (Dieleman et al., 2022; Veličković et al., 2026), and different models have different optimal sampling temperatures. We therefore evaluate the Gen. PPL and the average unigram entropy across a range of sampling temperatures T, and plot the resulting frontier for each model and NFE (Pynadath et al., 2025) (details in Suppl. B.7). At low entropy and NFE $= 1024$, LAW reaches a lower Gen. PPL than all diffusion and flow baselines and approaches AR (Figure 5). At both NFE $= 64$ and NFE $= 1024$, LAW also improves over $\mathbb{S}$-FLM with the standard DiT across most of the entropy range. At higher entropy, MDLM, Duo, and $\mathbb{S}$-FLM ($\mathbb{S}$-arch.) remain ahead.

Fine-Tuning and Ablations. Unlike on TinyGSM, fine-tuning Duo with LAW yields low-diversity samples on OpenWebText (unigram entropy ≤ 4.34; Suppl. C.9), possibly because unlike TinyGSM, generation is unconditional. Thus we train from scratch. In this case, Sandwich-Swin blocks and the mode embedding increase the sample entropy, and the stop-gradient stabilizes training. For example, when backpropagating through both passes, the hidden representations collapse, as indicated by a high hidden-state correlation, and the entropy and the Gen. PPL fluctuate strongly during training (Suppl. C.8).

5 Related Work

Self-Conditioning. Self-conditioning feeds the denoiser's previous prediction back as input (Chen et al., 2022; Strudel et al., 2022). In discrete diffusion, LDDMs (Jo et al., 2025) and Diffu-

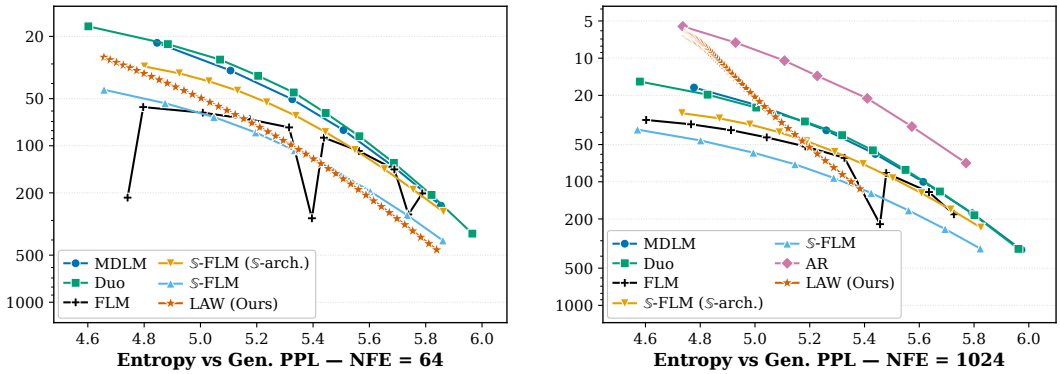


Figure 5: **Gen. PPL (↓) / Entropy (↑) Frontier on OpenWebText** at NFE $= 64$ (left) and NFE $= 1024$ (right). Each curve is obtained by sweeping over the sampling temperature T (Suppl. B.7 and Suppl. B.6). AR generates one token per NFE, so it only appears at NFE $= 1024$.

sionGemma (Google DeepMind, 2026) carry a continuous state across steps. Relay (Rozonoyer et al., 2026) trains the SC pathway with truncated backpropagation through time, whereas LAW stops the gradient between passes, which stabilizes training on OpenWebText (Suppl. C.8). Like LAW, SUNDAE (Savinov et al., 2022) unrolls the denoiser over several passes during training, but operates on discrete tokens. Soft-Masked Diffusion (Hersche et al., 2026) feeds masked positions a convex combination of the mask embedding and the top-k most confident token embeddings. All of these methods still sample discrete tokens between steps and use any continuous state as an auxiliary input. LAW carries only the SC state, at all positions, and decodes once, after the final step (Suppl. A.3).

Continuous Language Models. Continuous diffusion and flow language models generate by integrating an ODE or SDE. They corrupt token embeddings or one-hot vectors with Gaussian noise (Li et al., 2022; Gulrajani & Hashimoto, 2023; Lee et al., 2026a; Potaptchik et al., 2026; Roos et al., 2026; Chen et al., 2026; Hu et al., 2026; Yoo et al., 2026) or operate on the hypersphere (Deschenaux & Gulcehre, 2026; Chemseddine et al., 2026). Like these models, LAW has a continuous sampling trajectory, but it is trained with the uniform discrete corruption of UDMs and requires neither Gaussian noise nor an ODE solver. On GSM8K, the flow baselines at most match DDMs, whereas LAW, fine-tuned from Duo, is more accurate than the diffusion and flow baselines (Sec. 4.2).

Looped Models. Looped models reuse layers of the network to vary its depth at inference (Carballo-Castro et al., 2026; Lee et al., 2026b; Miele et al., 2026; Suleymanzade et al., 2026). Looping is orthogonal to LAW: it changes the computation within a denoising step, whereas LAW changes what is carried between steps.

6 Conclusion and Limitations

We asked whether a generative model can be based purely on self-conditioning, and proposed LAW: starting from random tokens, it propagates only the SC state (the expected embedding under the denoiser's prediction) and decodes once, after the final step. Our results suggest that pure self-conditioning can compete with self-conditioning as an auxiliary input: on hard Sudoku, LAW comes close to LDDMs (90.4% vs. 92.7%) and far exceeds continuous models (at most 45.0%), and on Maze, it is the most accurate model with 2 to 64 NFE. Such a simple method goes surprisingly far, and we see LAW as especially promising for post-training: fine-tuning a pre-trained Duo checkpoint for ∼20k steps on TinyGSM increases the GSM8K accuracy from 18.6% to 55.2%, with 46.0% in only 16 NFE. For pre-training on text, the picture is mixed: trained from scratch, LAW reaches 30.8% on GSM8K at 1024 NFE, compared to 32.3% for LDDM-M and 34.5% for LDDM-U. On OpenWebText, up to an entropy of about 5.0 with 1024 NFE, it achieves a lower Gen. PPL than the diffusion and flow baselines, while MDLM, Duo, and 𝕊-FLM (𝕊-arch.) remain ahead at higher entropy. Finally, our experiments are at academic scale.

AI USE STATEMENT

We acknowledge the use of LLMs during the preparation of the manuscript. LLMs assisted in the writing of the paper and code, and certain experiments were submitted using agentic tools. The main ideas were proposed without the use of LLMs. We take full responsibility for the content, claims, citations and conclusions. The TinyGSM dataset is fully generated by GPT-3.5 (as explained by Liu et al., 2023).

REFERENCES

Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants, 2023. URL https://arxiv.org/abs/2209.15571.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

Ali Alp. Sudoku puzzle generator. https://github.com/alicommit-malp/sudoku, 2024.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2023. URL https://arxiv.org/abs/2107.03006.

Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

Junyeob Baek, Mingyu Jo, Minsu Kim, Mengye Ren, Yoshua Bengio, and Sungjin Ahn. Generative recursive reasoning. *arXiv preprint arXiv:2605.19376*, 2026.

Georgios Batzolis, Mark Girolami, and Luca Ambrogioni. Cobit: Language modeling with bitstream diffusion, 2026. URL https://arxiv.org/abs/2605.07013.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models, 2022. URL https://arxiv.org/abs/2205.14987.

Alba Carballo-Castro, Julianna Piskorz, Paulius Rauba, Mihaela van der Schaar, and Pascal Frossard. Recursive scaling in masked diffusion models, 2026. URL https://arxiv.org/abs/2606.18022.

Jannis Chemseddine, Gregor Kornhardt, and Gabriele Steidl. Spherical flows for sampling categorical data. *arXiv preprint arXiv:2605.05629*, 2026.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning, 2022. URL https://arxiv.org/abs/2208.04202.

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, 2021. URL https://arxiv.org/abs/2110.14168.

Justin Deschenaux and Caglar Gulcehre. Promises, outlooks and challenges of diffusion language modeling, 2024. URL https://arxiv.org/abs/2406.11473.

Justin Deschenaux and Caglar Gulcehre. Language modeling with hyperspherical flows. *arXiv preprint arXiv:2605.11125*, 2026.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data, 2022. URL https://arxiv.org/abs/2211.15089.

Guhao Feng, Yihan Geng, Jian Guan, Wei Wu, Liwei Wang, and Di He. Theoretical benefit and limitation of diffusion language model, 2025. URL https://arxiv.org/abs/2502.09622.

Jonas Geiping, Sean McLeish, Neel Jain, John Kirchenbauer, Siddharth Singh, Brian R. Bartoldson, Bhavya Kailkhura, Abhinav Bhatele, and Tom Goldstein. Scaling up test-time compute with latent reasoning: A recurrent depth approach, 2025. URL https://arxiv.org/abs/2502.05171.

Gemini Team, Rohan Anil, Sebastian Borgeaud, et al. Gemini: A family of highly capable multimodal models, 2025. URL https://arxiv.org/abs/2312.11805.

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

Google DeepMind. DiffusionGemma. https://deepmind.google/models/gemma/diffusiongemma/, 2026. Model documentation, accessed August 2026.

Ishaan Gulrajani and Tatsunori B. Hashimoto. Likelihood-based diffusion language models, 2023. URL https://arxiv.org/abs/2305.18619.

Aric A. Hagberg, Daniel A. Schult, and Pieter J. Swart. Exploring network structure, dynamics, and function using networkx. In Gaël Varoquaux, Travis Vaught, and Jarrod Millman (eds.), *Proceedings of the 7th Python in Science Conference*, pp. 11 – 15, Pasadena, CA USA, 2008.

Michael Hersche, Samuel Moor-Smith, Thomas Hofmann, and Abbas Rahimi. Soft-masked diffusion language models. In *International Conference on Learning Representations*, 2026. URL https://arxiv.org/abs/2510.17206.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions, 2021. URL https://arxiv.org/abs/2102.05379.

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.

Jaehyeong Jo and Sung Ju Hwang. Continuous diffusion model for language modeling, 2025. URL https://arxiv.org/abs/2502.11564.

Mingyu Jo, Jaesik Yoon, Justin Deschenaux, Caglar Gulcehre, and Sungjin Ahn. Loopholing discrete diffusion: Deterministic bypass of the sampling wall, 2025. URL https://arxiv.org/abs/2510.19304.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models, 2020. URL https://arxiv.org/abs/2001.08361.

Jaeyeon Kim, Seunggeun Kim, Taekyun Lee, David Z. Pan, Hyeji Kim, Sham Kakade, and Sitan Chen. Fine-tuning masked diffusion for provable self-correction, 2025. URL https://arxiv.org/abs/2510.01384.

Ouail Kitouni, Niklas Nolte, Diane Bouchacourt, Adina Williams, Mike Rabbat, and Mark Ibrahim. The factorization curse: Which tokens you predict underlie the reversal curse and more, 2024. URL https://arxiv.org/abs/2406.05183.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M. Boffi, and Jinwoo Kim. One-step language modeling via continuous denoising, 2026a. URL https://arxiv.org/abs/2602.16813.

Sanghyun Lee, Chunsan Hong, Seungryong Kim, Jonghyun Lee, Jongho Park, and Dongmin Park. Looped diffusion language models, 2026b. URL https://arxiv.org/abs/2605.26106.

Lucas Lehnert, Sainbayar Sukhbaatar, DiJia Su, Qinqing Zheng, Paul Mcvay, Michael Rabbat, and Yuandong Tian. Beyond a*: Better planning with transformers via search dynamics bootstrapping, 2024. URL https://arxiv.org/abs/2402.14083.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-lm improves controllable text generation, 2022. URL https://arxiv.org/abs/2205.14217.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023. URL https://arxiv.org/abs/2210.02747.

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.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022a. URL https://arxiv.org/abs/2209.03003.

Ze Liu, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo. Swin transformer v2: Scaling up capacity and resolution, 2022b. URL https://arxiv.org/abs/2111.09883.

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

Meta. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Andrea Miele, Yiming Qin, Alba Carballo-Castro, Justin Deschenaux, and Pascal Frossard. Fixed-point masked generative modeling, 2026. URL https://arxiv.org/abs/2605.31215.

Vaishnavh Nagarajan, Chen Henry Wu, Charles Ding, and Aditi Raghunathan. Roll the dice & look before you leap: Going beyond the creative limits of next-token prediction. *arXiv preprint arXiv:2504.15266*, 2025.

Lorenzo Noci, Sotiris Anagnostidis, Luca Biggio, Antonio Orvieto, Sidak Pal Singh, and Aurelien Lucchi. Signal propagation in transformers: Theoretical perspectives and the role of rank collapse, 2022. URL https://arxiv.org/abs/2206.03126.

Jingyang Ou, Shen Nie, Kaiwen Zheng, et al. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint*, 2024.

Vassilis Papadopoulos, Jérémie Wenger, and Clément Hongler. Arrows of time for large language models, 2024. URL https://arxiv.org/abs/2401.17505.

William Peebles and Saining Xie. Scalable diffusion models with transformers, 2023. URL https://arxiv.org/abs/2212.09748.

Peter Potaptchik, Jason Yim, Adhi Saravanan, Peter Holderrieth, Eric Vanden-Eijnden, and Michael S. Albergo. Discrete flow maps, 2026. URL https://arxiv.org/abs/2604.09784.

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. Candi: Hybrid discrete-continuous diffusion models, 2025. URL https://arxiv.org/abs/2510.22510.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 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, 2026. URL https://arxiv.org/abs/2602.12233.

Benjamin Rozonoyer, Jacopo Minniti, Dhruvesh Patel, Neil Band, Avishek Joey Bose, Tim G. J. Rudner, and Andrew McCallum. Learned relay representations for forward-thinking discrete diffusion models, 2026. URL https://arxiv.org/abs/2605.22967.

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, 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, 2025. URL https://arxiv.org/abs/2506.10892.

Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation, 2022. URL https://arxiv.org/abs/2112.06749.

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P. de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models, 2025. URL https://arxiv.org/abs/2412.10193.

Junzhe Shen, Jieru Zhao, Ziwei He, and Zhouhan Lin. Codar: Continuous diffusion language models are more powerful than you think, 2026. URL https://arxiv.org/abs/2603.02547.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data, 2025. URL https://arxiv.org/abs/2406.04329.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and Rémi Leblond. Self-conditioned embedding diffusion for text generation, 2022. URL https://arxiv.org/abs/2211.04236.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding, 2023. URL https://arxiv.org/abs/2104.09864.

Ayhan Suleymanzade, Chanhyuk Lee, Floor Eijkelboom, Nicholas M. Boffi, İsmail İlkan Ceylan, and Jinwoo Kim. Thinking with looped flows, 2026. URL https://arxiv.org/abs/2609.11801.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL https://arxiv.org/abs/1706.03762.

Petar Veličković, Federico Barbero, Christos Perivolaropoulos, Simon Osindero, and Razvan Pascanu. Perplexity cannot always tell right from wrong, 2026. URL https://arxiv.org/abs/2601.22950.

Dimitri von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. Generalized interpolating discrete diffusion, 2025. URL https://arxiv.org/abs/2503.04482.

Guan Wang, Jin Li, Yuhao Sun, Xing Chen, Changling Liu, Yue Wu, Meng Lu, Sen Song, and Yasin Abbasi Yadkori. Hierarchical reasoning model. *arXiv preprint arXiv:2506.21734*, 2025. URL https://arxiv.org/abs/2506.21734.

Zhihan Yang, Wei Guo, Shuibai Zhang, Subham Sekhar Sahoo, Yongxin Chen, Arash Vahdat, Morteza Mardani, and John Thickstun. Continuous diffusion scales competitively with discrete diffusion for language, 2026. URL https://arxiv.org/abs/2605.18530.

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, 2026. URL https://arxiv.org/abs/2607.00714.

CONTENTS

1 Introduction **1**

2 Background **3**

2.1 Uniform Diffusion Models (UDMs) 3

2.2 Loopholing Discrete Diffusion Models (LDDMs) 3

3 Loopholing All the Way (LAW) **4**

3.1 Generative Process 4

3.2 Training the Continuous Trajectory 4

4 Experiments **5**

4.1 Reasoning on Sudoku 5

4.2 Reasoning on GSM8K 6

4.3 Path finding on Maze 7

4.4 Language Modeling 8

5 Related Work **8**

6 Conclusion and Limitations **9**

A Method Details **16**

A.1 Uniform Diffusion: Posterior and Objective 16

A.2 Sampling Algorithm 16

A.3 Comparison to Loopholing Discrete Diffusion Models (LDDMs) 16

A.4 Ablating the Backbone Input and State Map 17

A.5 Pre-Norm and Sandwich-Swin Blocks 17

A.6 Backpropagation Through Passes 17

A.7 Gaussian and Spherical Regularization 18

A.8 Mode Embedding 18

B Experimental Setup **18**

B.1 Training Hyperparameters 18

B.2 Baselines 18

B.3 Sudoku 19

B.4 TinyGSM 19

B.5 Maze 20

B.6 OpenWebText 21

B.7 Generative Perplexity and Entropy 22

B.8 Hidden-State Correlation 22

C Additional Results **22**

C.1 Sudoku: Full Results 22

C.2 TinyGSM: Regularization Sweeps 23

C.3 TinyGSM: Backbone Input and State Map Ablations 23

C.4 TinyGSM: Training from Scratch 24

C.5 Maze: Exact Match vs. Valid Solution 24

C.6 Maze: FLOP-Matching Schemes and Temperature Regularization 25

C.7 Maze: Fine-Tuning from Duo 26

C.8 OpenWebText: Training from Scratch 26

C.9 OpenWebText: Fine-Tuning 27

C.10 OpenWebText: Samples 33

A METHOD DETAILS

A.1 UNIFORM DIFFUSION: POSTERIOR AND OBJECTIVE

We reproduce the exact UDM quantities of Duo (Sahoo et al., 2025), with $\alpha_{t|s} = \alpha_t/\alpha_s$. The denoising posterior iterated in (4) is

$$q_{s|t}(. \mid \mathbf{z}_t^\ell, \mathbf{x}^\ell) = \mathrm{Cat}\left(.; \frac{|\mathcal{V}|\alpha_t \mathbf{z}_t^\ell \odot \mathbf{x}^\ell + (\alpha_{t|s} - \alpha_t)\mathbf{z}_t^\ell + (\alpha_s - \alpha_t)\mathbf{x}^\ell + (1-\alpha_{t|s})(1-\alpha_s)\mathbf{1}/|\mathcal{V}|}{|\mathcal{V}|\alpha_t \langle \mathbf{z}_t^\ell, \mathbf{x}^\ell \rangle + 1 - \alpha_t}\right). \tag{13}$$

This posterior induces the per-sequence NELBO whose average over $\mathbf{x}^{1:L} \sim q_{\text{data}}$ is (3), $\mathrm{NELBO}(q, p_\theta; \mathbf{x}^{1:L}) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\, q_t(\mathbf{z}_t^{1:L}|\mathbf{x}^{1:L};\alpha_t)} \sum_{\ell\in[L]} \mathcal{L}_{\text{tok}}(\mathbf{x}_\theta^\ell(\mathbf{z}_t^{1:L}, t), \mathbf{x}^\ell, \mathbf{z}_t^\ell, \alpha_t)$, whose per-token loss is

$$\mathcal{L}_{\text{tok}}(\mathbf{x}_\theta^\ell(\mathbf{z}_t^{1:L}, t), \mathbf{x}^\ell, \mathbf{z}_t^\ell, \alpha_t) = -\frac{\alpha_t'}{|\mathcal{V}|\alpha_t}\Bigg[\frac{|\mathcal{V}|}{\bar{\mathbf{x}}_r^\ell} - \frac{|\mathcal{V}|}{(\bar{\mathbf{x}}_\theta^\ell)_r} - \left(\zeta_t \mathbb{1}_{\mathbf{z}_t^\ell = \mathbf{x}^\ell} + \mathbb{1}_{\mathbf{z}_t^\ell \neq \mathbf{x}^\ell}\right) \sum_j \log \frac{(\bar{\mathbf{x}}_\theta^\ell)_r}{(\bar{\mathbf{x}}_\theta^\ell)_j}$$
$$- |\mathcal{V}| \frac{\alpha_t}{1-\alpha_t} \log \frac{(\bar{\mathbf{x}}_\theta^\ell)_r}{(\bar{\mathbf{x}}_\theta^\ell)_i} \mathbb{1}_{\mathbf{z}_t^\ell \neq \mathbf{x}^\ell} - \left((|\mathcal{V}|-1)\zeta_t \mathbb{1}_{\mathbf{z}_t^\ell = \mathbf{x}^\ell} - \frac{1}{\zeta_t}\mathbb{1}_{\mathbf{z}_t^\ell \neq \mathbf{x}^\ell}\right) \log \zeta_t\Bigg]. \tag{14}$$

Here $\bar{\mathbf{x}}^\ell = |\mathcal{V}|\alpha_t \mathbf{x}^\ell + (1-\alpha_t)\mathbf{1}$, $\bar{\mathbf{x}}_\theta^\ell = |\mathcal{V}|\alpha_t \mathbf{x}_\theta^\ell(\mathbf{z}_t^{1:L}, t) + (1-\alpha_t)\mathbf{1}$, α_t' is the time derivative of α_t, $r = \arg\max_{j\in[|\mathcal{V}|]} (\mathbf{z}_t^\ell)_j$ is the nonzero entry index of $\mathbf{z}_t^\ell$, $\zeta_t = \frac{1-\alpha_t}{|\mathcal{V}|\alpha_t + 1 - \alpha_t}$ and i is the index with $\mathbf{x}_i^\ell = 1$.

A.2 SAMPLING ALGORITHM

Algo. 1 summarizes the sampler of LAW (Sec. 3.1).

Algorithm 1 Sampling with LAW

Require: Denoiser f_θ, embedding E_θ, LM head g_θ, state map $\Phi_\theta^{\text{E.E.}}$, decoder η, sampling temperature T, time grid $t_N > \cdots > t_1$

$\mathbf{z}_1 \sim \mathrm{Cat}(\cdot\,; |\mathcal{V}|^{-1}\mathbf{1})$ ▷ Draw from the uniform prior
$\mathbf{c} \leftarrow \mathbf{z}_1 E_\theta$ ▷ Embed to initialize the state
for $n = N$ **to** 2 **do**
  $\mathbf{h} \leftarrow f_\theta(\mathbf{c}, t_n)$ ▷ Denoiser forward pass
  $\mathbf{c} \leftarrow \Phi_\theta^{\text{E.E.}}(\mathbf{h})$ ▷ Propagate the SC state (8)
end for
$\mathbf{h} \leftarrow f_\theta(\mathbf{c}, t_1)$ ▷ Final forward pass
return $\mathbf{z}_{t_0} = \eta(\mathrm{softmax}(g_\theta(\mathbf{h})/T))$ ▷ Decode tokens (9)

A.3 COMPARISON TO LOOPHOLING DISCRETE DIFFUSION MODELS (LDDMS)

LAW and LDDMs both carry an SC state across denoising steps. They differ in how the backbone uses this state, which affects both sampling and training, and in the state map that computes it (8).

Sampling. At each step, an LDDM samples discrete tokens from the posterior $q_{s|t}$ and feeds the backbone the sum of their embeddings and the SC state (5). We denote this input *add*. LAW instead feeds the backbone the SC state alone (7), which we denote *replace*. It never samples tokens between steps and decodes only once, after the final step (9).

Training. Both methods corrupt the data once, $\mathbf{z}_t \sim q_t$, run several forward passes, and stop the gradient through the SC state between passes. On 90% of the updates, LDDMs run two passes at time t: the first pass receives $\mathbf{z}_t E_\theta$ with a zero SC state and computes the SC state $\mathbf{c}_t$, and the second pass receives $\mathbf{z}_t E_\theta + \mathbf{c}_t$. The NELBO loss is applied only to the second pass. On the remaining updates, LDDMs train a single pass with a zero SC state (Sec. 2.2). In contrast, LAW unrolls M passes at successive times t_i (10) and averages the NELBO over all of them (11), so every pass contributes to the loss.

State Map. LDDMs carry the layer-normalized hidden representation ($\Phi_\theta^{\text{Hid.}}$), while LAW carries a linear combination of token embeddings, weighted by the predicted token probabilities ($\Phi_\theta^{\text{E.E.}}$) (8).

A.4 Ablating the Backbone Input and State Map

Suppl. A.3 identifies two choices that distinguish LAW from LDDMs: the backbone input (*add* or *replace*) and the state map ($\Phi_\theta^{\text{Hid.}}$ or $\Phi_\theta^{\text{E.E.}}$). LAW is ($\Phi_\theta^{\text{E.E.}}$, *replace*). To isolate the effect of each choice, we ablate both, which gives three alternatives: ($\Phi_\theta^{\text{Hid.}}$, *replace*), ($\Phi_\theta^{\text{E.E.}}$, *add*), and ($\Phi_\theta^{\text{Hid.}}$, *add*). With *add*, the sampler is that of LDDMs regardless of the state map, so ($\Phi_\theta^{\text{E.E.}}$, *add*) only changes the state that is carried alongside the tokens. With *replace*, the trajectory is fully continuous, and ($\Phi_\theta^{\text{Hid.}}$, *replace*) carries the hidden state instead of the expected embedding. Although ($\Phi_\theta^{\text{Hid.}}$, *add*) uses the same sampler and state map as LDDMs, it is not an LDDM because it is trained with our objective (11): it runs both passes on every update and applies the loss to both, whereas LDDMs run the first pass on only 90% of the updates and apply the loss only to the second pass. We fine-tune LAW and the three ablations on TinyGSM from the same Duo checkpoint, with our objective (11) and $M = 2$ passes, and compare them in Suppl. C.3.

A.5 Pre-Norm and Sandwich-Swin Blocks

Pre-Norm. We use the Diffusion Transformer (DiT) backbone with the DiT adaLN-Zero block (Peebles & Xie, 2023; Lou et al., 2024):

$$
\begin{aligned}
a &= \mathrm{Attn}\Big(\mathrm{Mod}\big(\mathrm{LN}_1(\mathbf{u});\,\gamma_1,\beta_1\big)\Big)\\
\mathbf{u} &\leftarrow \mathbf{u} + \lambda_1 \odot \mathrm{Dropout}(a)\\
m &= \mathrm{MLP}\Big(\mathrm{Mod}\big(\mathrm{LN}_3(\mathbf{u});\,\gamma_2,\beta_2\big)\Big)\\
\mathbf{u} &\leftarrow \mathbf{u} + \lambda_2 \odot \mathrm{Dropout}(m)
\end{aligned}
$$

Each block conditions on the noise level α_t through $\boldsymbol{\nu} = \mathrm{SiLU}(\mathrm{Emb}(\alpha_t))$. A per-block linear map $W^{(\ell)}$ produces six modulation vectors $(\beta_1, \gamma_1, \lambda_1, \beta_2, \gamma_2, \lambda_2) = W^{(\ell)}\boldsymbol{\nu}$, applied as $\mathrm{Mod}(\mathbf{u}; \gamma, \beta) = (1+\gamma)\odot\mathbf{u} + \beta$ for the scales and shifts, and as a gate λ on each branch output. We initialize $W^{(\ell)} = 0$, so $\lambda_1 = \lambda_2 = 0$ and every block starts as the identity. Attn denotes multi-head self-attention with RoPE (Su et al., 2023) and its output projection.

Sandwich-Swin. Following prior work (Liu et al., 2022b; Geiping et al., 2025), we apply two additional normalization steps to the branch output before gating:

$$
\begin{aligned}
a &= \mathrm{LN}_2\Big(\mathrm{Attn}\big(\mathrm{Mod}(\mathrm{LN}_1(\mathbf{u});\,\gamma_1,\beta_1)\big)\Big)\\
\mathbf{u} &\leftarrow \mathbf{u} + \lambda_1 \odot \mathrm{Dropout}(a)\\
m &= \mathrm{LN}_4\Big(\mathrm{MLP}\big(\mathrm{Mod}(\mathrm{LN}_3(\mathbf{u});\,\gamma_2,\beta_2)\big)\Big)\\
\mathbf{u} &\leftarrow \mathbf{u} + \lambda_2 \odot \mathrm{Dropout}(m)
\end{aligned}
$$

A.6 Backpropagation Through Passes

In (10), the SC state fed to the next pass is detached, $\mathbf{c}^{(i+1)} = \mathrm{sg}\big[\Phi_\theta^{\text{E.E.}}(\mathbf{h}^{(i)};\tau)\big]$, where $\mathbf{h}^{(i)} = f_\theta(\mathbf{c}^{(i)}, t_i)$ is the final-layer hidden representation of the i-th pass. As a result, the i-th term $\mathcal{L}^{(i)}$ of the loss (11) only updates the parameters through the i-th forward pass, and the earlier passes are treated as a fixed input.

Alternatively, we can remove the stop-gradient and set $\mathbf{c}^{(i+1)} = \Phi_\theta^{\text{E.E.}}(\mathbf{h}^{(i)};\tau)$. The i-th loss term then depends on all passes $1,\dots,i$, and its gradient also flows through the SC states into the earlier passes. For $M = 2$, the gradient of the second term with respect to the output of the first pass can be expressed as:

$$
\frac{\partial \mathcal{L}^{(2)}}{\partial \mathbf{h}^{(1)}} = \frac{\partial \mathcal{L}^{(2)}}{\partial \mathbf{c}^{(2)}}\,\frac{\partial \Phi_\theta^{\text{E.E.}}(\mathbf{h}^{(1)};\tau)}{\partial \mathbf{h}^{(1)}}, \tag{15}
$$

which is zero with the stop-gradient. Without it, the first pass is trained not only to predict the clean tokens, but also to produce an SC state that is useful to the second pass. We use the stop-gradient by default and compare both variants on OpenWebText, when training from scratch (Figure 16) and when fine-tuning (Figure 18).

A.7 Gaussian and Spherical Regularization

Besides temperature regularization (Sec. 3.2), we consider two perturbations of the SC state, replacing each $\mathbf{c}$ with $\widetilde{\mathbf{c}}$ before the next pass (10). First, **Gaussian regularization**, which adds isotropic noise, $\widetilde{\mathbf{c}}^\ell = \mathbf{c}^\ell + \sigma\boldsymbol{\epsilon}^\ell$ with $\boldsymbol{\epsilon}^\ell \sim \mathcal{N}(\mathbf{0}, \mathbf{I}_d)$, as in stochastic recursive reasoning models (Baek et al., 2026). Second, inspired by Hyperspherical Flow Models (Deschenaux & Gulcehre, 2026; Chemseddine et al., 2026), we propose **spherical regularization** that perturbs the direction of the state without changing its norm. Writing $\hat{\mathbf{c}}^\ell = \mathbf{c}^\ell / \|\mathbf{c}^\ell\|$ for the normalized state at position ℓ, $\hat{\mathbf{r}}^\ell$ for a uniformly random unit direction, and $\Omega^\ell = \arccos\langle \hat{\mathbf{c}}^\ell, \hat{\mathbf{r}}^\ell \rangle$ for the angle between them, spherical interpolation by a fraction $\gamma \in [0, 1]$ rotates the state toward $\hat{\mathbf{r}}^\ell$:

$$\widetilde{\mathbf{c}}^\ell = \|\mathbf{c}^\ell\| \, \frac{\sin\big((1-\gamma)\Omega^\ell\big)\, \hat{\mathbf{c}}^\ell + \sin\big(\gamma\Omega^\ell\big)\, \hat{\mathbf{r}}^\ell}{\sin \Omega^\ell}. \tag{16}$$

A.8 Mode Embedding

At the first pass (10) and the first sampling step, the denoiser receives token embeddings, whereas at every later pass it receives the SC state. The mode embedding tells the denoiser which of the two it is processing. We learn two vectors $\mathbf{m}_0, \mathbf{m}_1$ and add one of them to the noise-level conditioning of Suppl. A.5, $\boldsymbol{\nu} = \mathrm{SiLU}(\mathrm{Emb}(\alpha_t)) + \mathbf{m}_k$, with $k = 0$ for token embeddings and $k = 1$ for SC states. We initialize $\mathbf{m}_0 = \mathbf{m}_1 = \mathbf{0}$, so that fine-tuning from a checkpoint without mode embedding, such as Duo, starts from the same function (Suppl. C.9).

B Experimental Setup

B.1 Training Hyperparameters

All models are trained with AdamW with a learning rate of 3×10^{-4}, $(\beta_1, \beta_2) = (0.9, 0.999)$, zero weight decay, 2.5k warmup steps followed by a constant learning rate, and gradient clipping at 1.0. We maintain an exponential moving average of the weights with decay 0.9999 and use it for evaluation. All DiT backbones use dropout 0.1. The Sudoku and Maze models use 8 attention heads, and the TinyGSM and OpenWebText models use 12.

B.2 Baselines

We use the same baselines on all benchmarks (Sudoku, TinyGSM, Maze, and OpenWebText). Beyond the shared hyperparameters of Suppl. B.1, we use the default hyperparameters of the original implementations unless stated otherwise.

- **AR.** Causal Transformer trained with next-token prediction on the target, conditioned on the full prompt when there is one (e.g. the unsolved grid on Maze).
- **MDLM** (Sahoo et al., 2024). Masked diffusion with the SUBS parameterization, the log-linear schedule, and no time conditioning.
- **Duo** (Sahoo et al., 2025). Uniform diffusion with the log-linear schedule and time conditioning.
- **LDDM** (Jo et al., 2025). The masked (LDDM-M) and uniform (LDDM-U) variants are built on MDLM and Duo, respectively. We describe LDDMs in Sec. 2.2 and compare them with LAW in Suppl. A.3.
- **FLM** (Lee et al., 2026a). Gaussian flow on one-hot token representations, with time conditioning on $t \in [0, 1]$.

- **CANDI** (Pynadath et al., 2025). Hybrid discrete–continuous diffusion with mixing coefficient 0.5, $\sigma_{\min} = 0.01$, $\sigma_{\max} = 2$, and percentile scheduling with bounds 0.01 and 0.25.
- **$\mathbb{S}$-FLM and $\mathbb{S}$-FLM ($\mathbb{S}$-arch.)** (Deschenaux & Gulcehre, 2026). Spherical flow trained with the cross-entropy loss and the log-linear schedule truncated at $\alpha_{\max} = \alpha^\star(0.1)$, which depends on the vocabulary size $|\mathcal{V}|$ and the embedding dimension d (Deschenaux & Gulcehre, 2026, Eq. (17)). $\mathbb{S}$-FLM uses the standard DiT, whereas $\mathbb{S}$-FLM ($\mathbb{S}$-arch.) uses the Sphere-DiT backbone with nGPT initialization.

Architecture. All diffusion and flow models, including $\mathbb{S}$-FLM, use the standard DiT (Peebles & Xie, 2023) (Suppl. A.5), except FLM and CANDI, which use the DiT variants of their original implementations at the same size, and $\mathbb{S}$-FLM ($\mathbb{S}$-arch.), which uses the Sphere-DiT. AR uses the same Transformer with causal attention and without time conditioning. LAW uses the same DiT as Duo, so a Duo checkpoint can be loaded into LAW without any change. On OpenWebText, LAW additionally uses Sandwich-Swin blocks and a mode embedding (Sec. 3.2).

Sampling. Unless stated otherwise, the baselines use the following samplers:

- **MDLM**, **Duo**: ancestral sampling with the posterior of each process, sampling the tokens at the last step.
- **LDDM**: ancestral sampling where the SC state of each step is carried to the next one.
- **FLM**: Euler integration of the flow. When there is a prompt, it is projected back to its clean value at the last step.
- **CANDI**: the CANDI sampler of the original implementation.
- **$\mathbb{S}$-FLM**, **$\mathbb{S}$-FLM ($\mathbb{S}$-arch.)**: the sampling setup of the original implementation. At each step, the velocity only points toward the embedding of the most likely token (top-1 velocity), and the tokens are decoded with an argmax at the last step.

B.3 SUDOKU

Dataset. Following prior work (Kim et al., 2025; Deschenaux & Gulcehre, 2026), we generate 48k training and 2k validation puzzles with unique solutions (Alp, 2024). The puzzles have three difficulty levels, determined by the number of visible digits: easy (40/81), medium (35/81), and hard (30/81). We report the *exact-match* accuracy.

Tokenization. Each Sudoku grid is tokenized as a 1D sequence: the partial grid, a beginning-of-sequence token, then the solution. Missing digits are encoded with an empty-cell token, and rows are separated by a row-separator token. The vocabulary contains 12 tokens, and every sequence has length 180. The partial grid serves as the prompt and is never corrupted during training.

Training. Duo models are trained for 20k updates at global batch size 256, using the shared hyperparameters of Suppl. B.1. Every LAW configuration in Table 3 follows the FLOP-matching rule of Sec. 4.1: with M passes, LAW processes $1/M$ as many training examples as Duo, by reducing either the batch size or the number of updates. We train each configuration with and without temperature regularization ($\rho = 3$). The baseline results in Table 1 are taken from prior work (Deschenaux & Gulcehre, 2026). Every LAW model is trained on a single A100 80GB GPU.

Sampling. We sample with 180 NFE. The baselines use the samplers of Suppl. B.2, and LAW uses Algo. 1 with the $\eta_{\max}$ decoder (9).

B.4 TINYGSM

Dataset. We train on TinyGSM (Liu et al., 2023), which pairs 11.8M synthetic grade-school math questions with Python programs that solve them, and evaluate on the GSM8K test set (Cobbe et al., 2021). A generated solution is correct if running the program returns the reference answer.

Tokenization. We tokenize with the HuggingFaceTB/SmolLM-135M tokenizer and discard examples longer than 512 tokens. Shorter examples are padded to 512 tokens. Each sequence is a question–program pair: the question serves as the prompt, and the loss is applied only to the program.

Training from Scratch. Every model is trained on 8 GH200 GPUs (2 nodes $\times$ 4 GPUs), and all models trained from scratch use a global batch size of 512 and the shared hyperparameters of Suppl. B.1. AR, MDLM, Duo, FLM, CANDI, and $\mathbb{S}$-FLM ($\mathbb{S}$-arch.) are trained for 250k steps. We match the training FLOPs of the other models to this budget, following the rule of Sec. 4.1. LDDMs run a second forward pass on 90% of the updates, so an LDDM update costs about 1.3 times a Duo update, and we train LDDM-U and LDDM-M for 192k steps. LAW runs $M = 2$ passes on every update, so an update costs twice a Duo update, and we train LAW for 125k steps. Suppl. C.4 compares LAW and the LDDMs trained from scratch.

Fine-Tuning. In Sec. 4.2, we fine-tune the Duo checkpoint trained for 250k steps with LAW for 50k steps at batch size 64, and report the checkpoint after 20k steps. The regularization sweeps (Suppl. C.2) and the backbone input and state map ablations (Suppl. C.3) use the same fine-tuning setup. Fine-tuning is done on a single H200 GPU.

Sampling. For each GSM8K test problem, we generate and execute one program. We sweep the NFE over $\{1, 2, 4, \dots, 1024\}$. The baselines use the samplers of Suppl. B.2, and LAW uses Algo. 1 with the η_{raw} decoder (9).

B.5 MAZE

Dataset. We regenerate Maze-Hard (Wang et al., 2025) on 20×20 grids with an existing maze generator (Lehnert et al., 2024). Each cell is independently a `wall` with probability $p \sim \mathcal{U}(0.3, 0.5)$. The `start` and the `goal` are placed uniformly at random on two distinct `empty` cells. We keep only solvable mazes whose shortest path contains at least 71 cells, including the `start` and the `goal`. The original Maze-Hard (Wang et al., 2025) uses 30×30 grids and a threshold of 110 cells. We use 20×20 grids for computational reasons, which shortens the sequences from 1800 to 800 tokens. On our grids, a threshold of 71 cells keeps approximately the same fraction of generated mazes. We generate 1,000 training and 1,000 evaluation mazes with seed 42. The shortest paths of the evaluation mazes contain between 70 and 89 moves, with a median of 72. The target is one shortest path, found by breadth-first search with a fixed neighbor order. However, a maze can have several shortest paths. We verify all 2,000 targets independently of the generator by recomputing the shortest path length with the NetworkX library (Hagberg et al., 2008), and every target passes this check.

Tokenization. Each maze is tokenized as a 1D sequence: the unsolved grid followed by the solved grid, each flattened row by row, from top to bottom and left to right. Each cell is one of five tokens: `wall`, `empty`, `start`, `goal`, or `path`. With an additional padding token, the vocabulary contains 6 tokens, plus a mask token for MDLM, LDDM-M, CANDI, and AR. Every sequence has length $2 \times 20^2 = 800$. The unsolved grid serves as the prompt: it is never corrupted during training, the loss is computed only on the solved grid, and the sampler resets the prompt to its clean value at every step.

Training Budget. We follow the FLOP-matching rule of Sec. 4.1. All baselines are trained for 100k steps at batch size 256, except LDDMs, which are trained for 76.9k steps to account for their extra forward pass. Since an update of LAW with M passes costs M times as much as a baseline update, LAW processes $1/M$ as many training examples. We achieve this in two ways: (i) with a fixed batch size of 256 and $100\text{k}/M$ steps, or (ii) with a fixed number of 50k steps and batch sizes of 256, 128, and 64 for $M = 2$, 4, and 8, respectively. Table 2 lists the budgets of all LAW runs.

Training. Every model is trained from scratch on a single A100 80GB GPU. The baselines follow Suppl. B.2; for $\mathbb{S}$-FLM ($\mathbb{S}$-arch.), $|\mathcal{V}| = 6$ and $d = 256$ give $\alpha_{\max} = 0.121$. LAW trains with the objective (11), with a stop-gradient between the M passes (Suppl. A.6). We train LAW from scratch with $M \in \{2, 4, 8\}$ under both budget schemes, with and without temperature regularization

Table 2: Training budgets of the LAW runs on Maze-Hard. The budget in sample-passes is passes $\times$ batch size $\times$ updates. All baselines use batch size 256 and 100k updates (for the baselines, 25.6M sample-passes is computed as: $1 \times 256 \times 100\text{k} = 25.6\text{M}$), or 76.9k updates for LDDMs. The fine-tuned LAW runs start from the Duo checkpoint after 50k updates (12.8M sample-passes).

Model	Batch	Updates	Sample-passes
LAW, $M = 2$	256	50k	25.6M
LAW, $M = 4$ (fixed batch)	256	25k	25.6M
LAW, $M = 4$ (fixed updates)	128	50k	25.6M
LAW, $M = 8$ (fixed batch)	256	12.5k	25.6M
LAW, $M = 8$ (fixed updates)	64	50k	25.6M
LAW from Duo, $M = 2$	256	25k	12.8M + 12.8M
LAW from Duo, $M = 4$	256	12.5k	12.8M + 12.8M
LAW from Duo, $M = 8$	256	6.25k	12.8M + 12.8M

($\rho = 3$). We also fine-tune LAW with $M \in \{2, 4, 8\}$ from the EMA weights of the Duo checkpoint after 50k of its 100k updates, for $50\text{k}/M$ updates at batch size 256. This adds 12.8M sample-passes to the 12.8M of Duo pre-training, so the total budget matches that of the other runs.

Sampling. We evaluate on the 1,000 evaluation mazes. Unless stated otherwise, all diffusion and flow models sample with 800 NFE at sampling temperature $T = 1$. The AR model decodes the 400 cells of the solution grid one at a time with a KV cache, i.e., with 400 NFE. We report both greedy and sampled decoding. The baselines use the samplers of Suppl. B.2, and LAW uses Algo. 1 with the η_{raw} decoder (9). We sweep the NFE by evaluating on a number of steps in $\{1, 2, 4, \ldots, 512, 800\}$ and keep all other settings fixed.

Metrics. Prior work reports *exact-match* accuracy on Maze-Hard (Wang et al., 2025). Since a maze can have several shortest paths, *exact match* can underestimate the accuracy: a correct shortest path that differs from the target is counted as wrong (Suppl. C.5). We therefore report the *valid-solution* accuracy, i.e., the fraction of evaluation mazes for which the generated grid (i) reproduces every `wall`, `empty`, `start`, and `goal` cell of the maze, and (ii) marks a shortest path from the `start` to the `goal`. To check (ii), let n be the number of cells on the target path (i.e., the length of a shortest path), including the `start` and the `goal`. We check that the cells labeled `path` lie on `empty` cells of the maze and form a connected path of exactly n cells.

B.6 OpenWebText

We train on OpenWebText (Gokaslan & Cohen, 2019) with the GPT-2 tokenizer (Radford et al., 2019) and a context length of 1024.

Training from Scratch. We train on 16 GH200 GPUs (4 nodes $\times$ 4 GPUs) with a global batch size of 512 and the shared hyperparameters of Suppl. B.1. All models use the 12-layer DiT of Sec. 4.4. The baselines are trained for 400k steps. LAW runs $M = 2$ passes on every update, so we train it for 200k steps to match the training FLOPs of the baselines, following the rule of Sec. 4.1. By default, the SC state is detached between passes, so each pass is trained on a stop-gradient input (Suppl. A.6). In Suppl. C.8, we ablate this choice by backpropagating through both passes. We further ablate pre-norm (default DiT) against Sandwich-Swin blocks (Suppl. A.5), temperature regularization with $\rho \in \{3, 4\}$ (12), and the mode embedding (Suppl. A.8).

Fine-Tuning. We fine-tuned LAW for 50k steps, starting from Duo checkpoints trained for 200k steps, with the same hyperparameters as training from scratch.

Sampling. For the Gen. PPL / entropy frontier of Figure 5, we sample with 64 and 1024 NFE and sweep the sampling temperature over $T \in \{0.30, 0.35, \ldots, 3.00\}$ for LAW and $T \in \{0.70, 0.75, \ldots, 1.20\}$ for the baselines (Suppl. B.7). For the training curves of Suppl. C.8 and Suppl. C.9, we sample with 256 NFE at $T = 1$. Suppl. C.10 shows samples drawn with 64 and 1024 NFE. LAW uses Algo. 1 with the η_{raw} decoder (9).

B.7 Generative Perplexity and Entropy

We follow the evaluation protocol of $\mathbb{S}$-FLM (Deschenaux & Gulcehre, 2026) and compare LAW to AR, MDLM, Duo, FLM, $\mathbb{S}$-FLM, and $\mathbb{S}$-FLM ($\mathbb{S}$-arch.) at NFE $\in \{64, 1024\}$. Since an AR model generates one token per NFE, we only evaluate it at NFE $= 1024$. We sweep the sampling temperature over the ranges of Suppl. B.6. The samples of LAW change little over the range of the baselines, so we use a wider range for LAW. For each (T, NFE) we draw $N = 512$ samples of length $L = 1024$.

Generative Perplexity. We score each sample x with GPT-2 Large (Radford et al., 2019), ignoring tokens from the first end-of-text onward:

$$\mathrm{PPL}_{\mathrm{gen}}(x) = \exp\left(-\frac{1}{L}\sum_{i=1}^{L}\log p_{\mathrm{GPT2}}(x_i \mid x_{<i})\right). \tag{17}$$

We report the mean over the N samples.

Unigram Entropy. Low perplexity can come from repetitive text, so we also report the mean unigram entropy

$$H = -\frac{1}{N}\sum_{n=1}^{N}\sum_{v\in\mathcal{V}}\frac{c(v, x^{(n)})}{L}\log\frac{c(v, x^{(n)})}{L}, \tag{18}$$

where $c(v, x^{(n)})$ counts token v in sample $x^{(n)}$.

Plotted Range. Sweeping T gives one curve per method in Figure 5. We keep points with $4.5 \leq H \leq 6.0$ and $\mathrm{PPL}_{\mathrm{gen}} \leq 500$, dropping degenerate settings.

B.8 Hidden-State Correlation

As in prior work (Geiping et al., 2025), we measure representation collapse (Noci et al., 2022) with the correlation of the hidden states across positions. We take the final-layer hidden representation $\mathbf{h} \in \mathbb{R}^{L\times d}$ (2), i.e. the input to the LM head g_θ, at the final sampling step. We center each position over the embedding dimension, $\tilde{\mathbf{h}}^{\ell} = \mathbf{h}^{\ell} - \frac{1}{d}\sum_{j=1}^{d} h_j^{\ell}\,\mathbf{1}$, and compute the mean cosine similarity over all pairs of distinct positions,

$$r = \frac{1}{L(L-1)}\sum_{\ell\neq\ell'}\frac{\langle\tilde{\mathbf{h}}^{\ell}, \tilde{\mathbf{h}}^{\ell'}\rangle}{\|\tilde{\mathbf{h}}^{\ell}\|\,\|\tilde{\mathbf{h}}^{\ell'}\|}. \tag{19}$$

We report r averaged over the samples. A value close to 1 means that all positions share the same hidden state, i.e. the representations have collapsed.

C Additional Results

C.1 Sudoku: Full Results

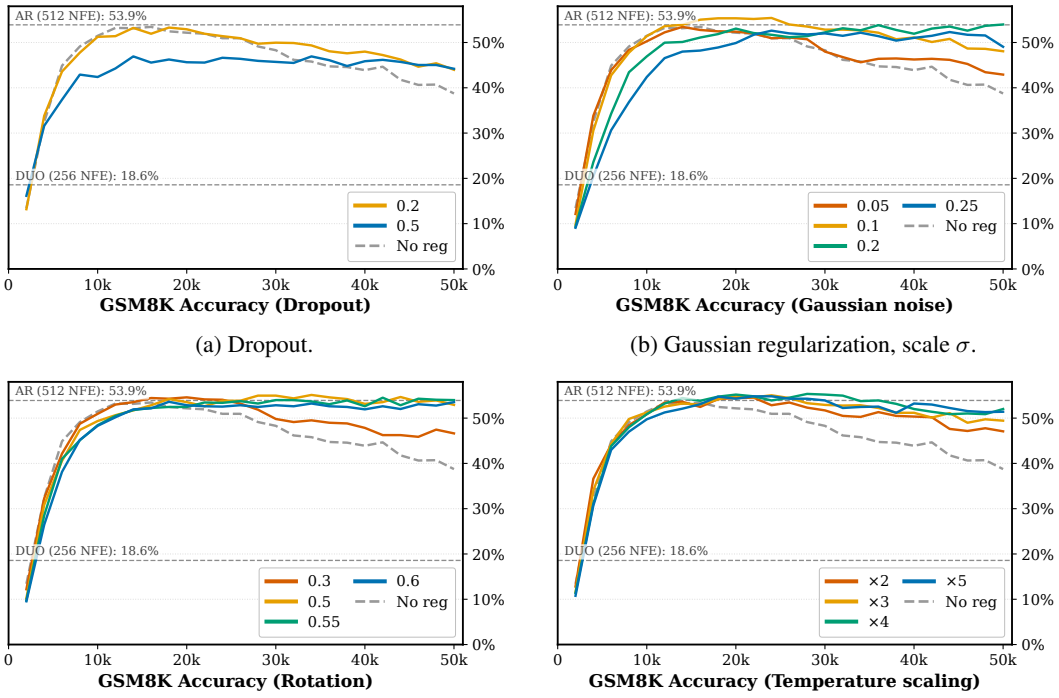
Table 3 reports the *exact-match* accuracy on Sudoku for LAW of Sec. 4.1, including the LAW runs with different numbers of passes M under the same FLOP budget, with and without temperature regularization.

Table 3: **LAW ablations on Sudoku.** *Exact-match* accuracy (%) with 180 NFE for different numbers of passes M and FLOP-matching schemes (Sec. 4.1), with and without temperature regularization. All runs use the same FLOP budget. Best per column among LAW runs in **bold**, second best underlined.

Regularization	M	Batch	Updates	Easy	Med.	Hard
None	2	256	10k	<u>99.2</u>	**97.0**	<u>88.6</u>
	4	128	10k	98.6	<u>95.9</u>	85.8
	4	256	5k	99.1	94.6	86.7
	8	64	10k	97.8	95.3	86.4
$\rho = 3$	2	256	10k	**99.3**	95.6	**90.4**
	4	128	10k	98.9	94.9	84.3
	4	256	5k	98.9	95.8	82.5
	8	64	10k	97.8	89.7	80.0

C.2 TinyGSM: Regularization Sweeps

Figure 6 reports the full regularization sweeps summarized in Sec. 4.2, each against an unregularized run and the Duo checkpoint at 256 NFE. Increasing dropout does not prevent the late decline, whereas perturbing the SC state does: temperature regularization reaches the highest peak accuracy, and spherical regularization the flattest curve.



(a) Dropout.

(b) Gaussian regularization, scale σ.

(c) Spherical regularization, fraction γ.

(d) Temperature regularization, factor ρ.

Figure 6: **Regularization sweeps on TinyGSM.** GSM8K accuracy over fine-tuning steps for each regularizer of Sec. 3.2, against an unregularized run. The horizontal line is Duo with 256 NFE.

C.3 TinyGSM: Backbone Input and State Map Ablations

We fine-tune the Duo checkpoint of Sec. 4.2 with LAW and each of the three ablations of Suppl. A.4, without regularization, and evaluate the GSM8K accuracy with 256 NFE (Figure 7). All four models share the objective, $M = 2$, and the fine-tuning budget, so the comparison isolates the backbone input and the state map. The original Duo checkpoint achieves 18.6% accuracy. LAW ($\Phi_\theta^{\text{E.E.}}$, *replace*)

increases the accuracy to 53.4% after 16k training steps. With ($\Phi_\theta^{\text{E.E.}\cdot}$, *add*), the accuracy peaks at 38.5%. With ($\Phi_\theta^{\text{Hid.}\cdot}$, *add*), it reaches 23.7% after 50k fine-tuning steps and is still improving, while with ($\Phi_\theta^{\text{Hid.}\cdot}$, *replace*), it stays close to 0%. The accuracy of LAW declines after its peak, which motivates the regularizers of Sec. 3.2.

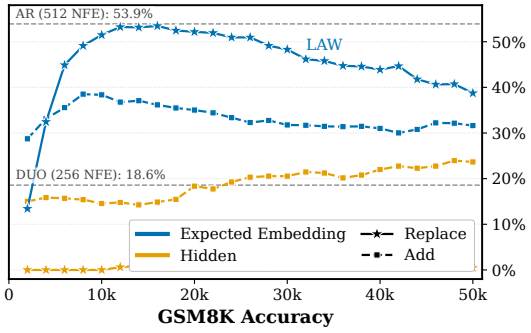
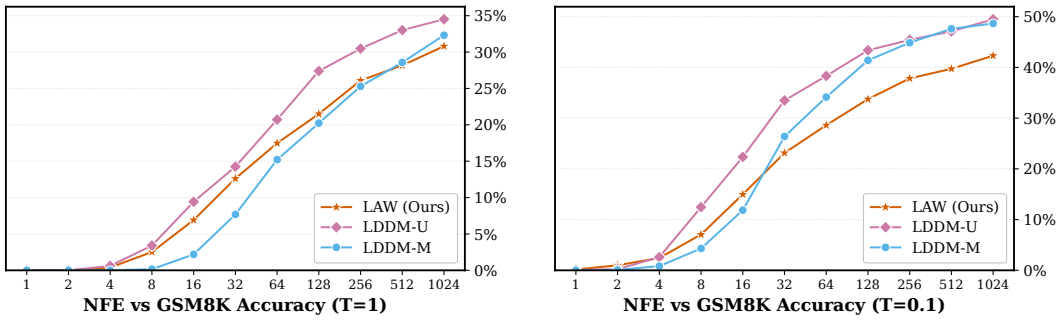


Figure 7: **Backbone input and state map ablations on TinyGSM.** GSM8K accuracy over fine-tuning steps, without regularization and with 256 NFE. Colors give the state map, expected embedding ($\Phi_\theta^{\text{E.E.}\cdot}$) or hidden state ($\Phi_\theta^{\text{Hid.}\cdot}$), and line styles the backbone input, *replace* or *add* (Suppl. A.3). The expected embedding with *replace* is LAW. The horizontal lines are the AR model and Duo.

C.4 TinyGSM: Training from Scratch

In Sec. 4.2, we obtain LAW by fine-tuning a pre-trained Duo checkpoint. Here, we instead train LAW, LDDM-U, and LDDM-M from scratch on TinyGSM with the same training FLOPs (Suppl. B.4). We compare the GSM8K accuracy against NFE at sampling temperatures $T = 1$ and $T = 0.1$ (Figure 8). LDDM-U is the most accurate model at both temperatures. At $T = 1$, LAW is ahead of or on par with LDDM-M up to 256 NFE and reaches 30.8% at 1024 NFE, compared to 32.3% for LDDM-M and 34.5% for LDDM-U. At $T = 0.1$, LAW reaches 42.3% at 1024 NFE, compared to 48.7% for LDDM-M and 49.5% for LDDM-U.



(a) Sampling temperature $T = 1$.

(b) Sampling temperature $T = 0.1$.

Figure 8: **GSM8K accuracy against NFE for models trained from scratch.** LAW ($M = 2$, 125k steps), LDDM-U, and LDDM-M (192k steps) are trained with the same FLOP budget (Suppl. C.4).

C.5 Maze: Exact Match vs. Valid Solution

We report the *valid-solution* accuracy on Maze-Hard (Suppl. B.5) because *exact match* counts every shortest path that differs from the target as wrong. Figure 9 shows such a case: LAW generates a shortest path that differs from the target, so it solves the maze but fails *exact match*. Figure 10 compares both metrics against NFE for all methods of Sec. 4.3. Since every exact match is also a valid solution, the *exact-match* accuracy never exceeds the *valid-solution* accuracy. The gap is largest for $\mathbb{S}$-FLM ($\mathbb{S}$-arch.), reaching 19 points.

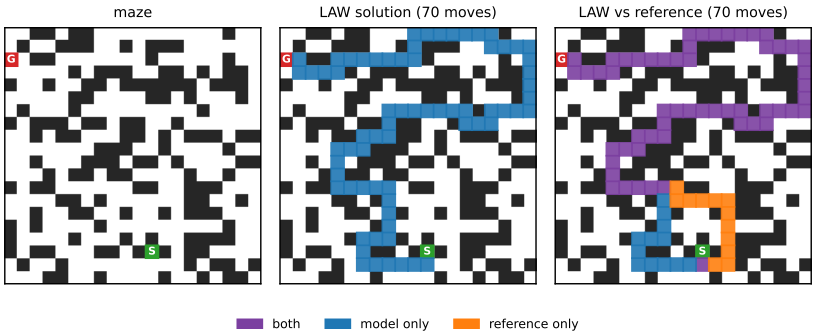
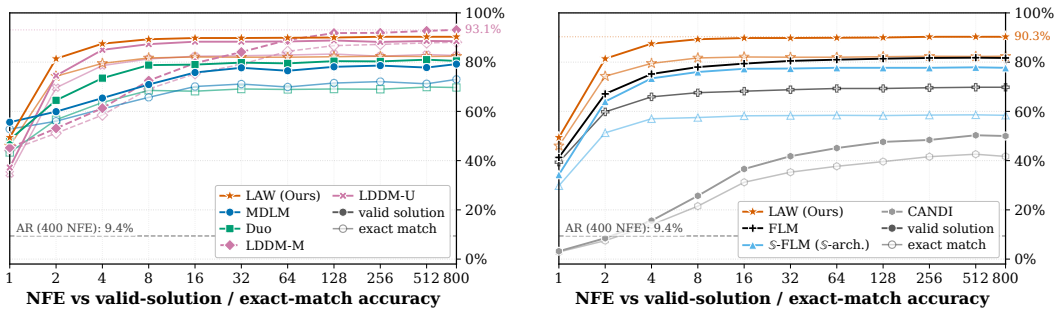


Figure 9: **A valid solution that fails *exact match*.** On this Maze-Hard evaluation maze, LAW generates a shortest path that differs from the target. **Left:** the maze. **Middle:** the path generated by LAW. **Right:** cells on both paths (purple), only on the LAW path (blue), and only on the target path (orange). Both paths take 70 moves, so the generated path solves the maze, but *exact match* counts it as wrong.



(a) Discrete diffusion.

(b) Continuous models.

Figure 10: **Maze-Hard *exact-match* vs. *valid-solution* accuracy against NFE** for the models of Figure 4.

C.6 Maze: FLOP-Matching Schemes and Temperature Regularization

The FLOP-matching rule of Sec. 4.1 can be met in two ways: by keeping the batch size fixed and dividing the number of training steps by M, or by keeping the number of steps fixed and dividing the batch size by M (Suppl. B.5). LAW benefits from more updates with a smaller batch, especially for $M = 8$ (Figure 11). We also ablate temperature regularization (Suppl. C.6) and fine-tuning from Duo (Suppl. C.7).

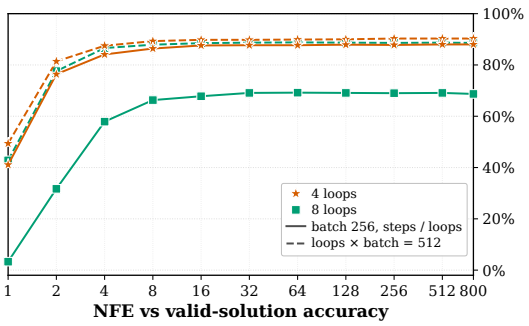


Figure 11: **Valid-solution accuracy against NFE for LAW with different FLOP-matching schemes. Left:** all runs use batch size 256 and are trained for 50k, 25k, and 12.5k steps for $M = 2$, 4, and 8, respectively. **Right:** all runs are trained for 50k steps with batch sizes 256, 128, and 64 for $M = 2$, 4, and 8, respectively, following Sec. 4.1.

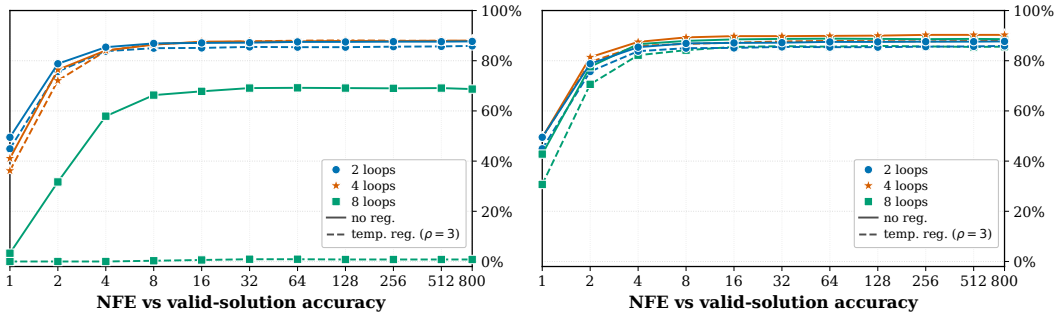


Figure 12: **Training from scratch without temperature scaling vs. with temperature scaling $\rho = 3$.** Valid-solution accuracy against NFE for LAW trained from scratch, at equal total budget. Both use batch size 256. Colors and markers give the number of passes M.

C.7 Maze: Fine-Tuning from Duo

Figure 13 compares LAW trained from scratch with batch size 256 and $100\text{k}/M$ updates to LAW fine-tuned with the same batch size for $50\text{k}/M$ updates from a Duo checkpoint pre-trained for 50k updates (Table 2). In this setting, where the budget is split evenly between Duo and LAW, fine-tuning generally does not improve performance over training from scratch. With $M = 2$ and $M = 4$, training from scratch is more accurate at 800 NFE (87.7% vs. 83.0% and 88.0% vs. 78.3%). With $M = 8$, the fine-tuned run is more accurate (76.8% vs. 68.7%), but the run from scratch then performs only 12.5k updates; with 50k updates and a smaller batch (Figure 11), it reaches 88.6%, above all fine-tuned runs. On TinyGSM, in contrast, LAW is fine-tuned for only a small fraction of the total budget (Sec. 4.2), which suggests that fine-tuning is most useful when the pre-trained model receives most of the budget. Confirming this would require further ablations on the budget split.

C.8 OpenWebText: Training from Scratch

We evaluate every 5k training steps with the EMA weights. The Gen. PPL and entropy are computed as in Suppl. B.7, from 512 samples drawn with NFE $= 256$ at $T = 1$. We also report the validation loss, i.e. the LAW training loss of Sec. 3.2 (11) evaluated on the validation split, and the hidden-state correlation (Suppl. B.8).

Temperature Regularization. Figure 14 compares $\rho = 3$ and $\rho = 4$ for both block types. We observe no large difference between the two values, apart from occasional drops in entropy with pre-norm blocks and $\rho = 4$.

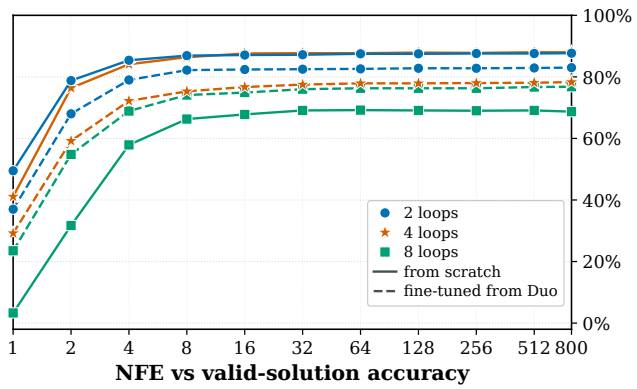


Figure 13: **Training from scratch vs. fine-tuning from Duo on Maze-Hard.** Valid-solution accuracy against NFE for LAW trained from scratch (solid) and fine-tuned from Duo (dashed), at equal total budget. Both use batch size 256. Colors and markers give the number of passes M.

Block Type and Mode Embedding. Figure 15 compares both block types, with and without the mode embedding. The mode embedding increases the entropy for both block types, with a larger gain for Sandwich-Swin blocks, and the Gen. PPL increases as well. The combination of Sandwich-Swin blocks, the mode embedding, and $\rho = 3$ gives our best-performing model, which we use in the main text (Sec. 4.4).

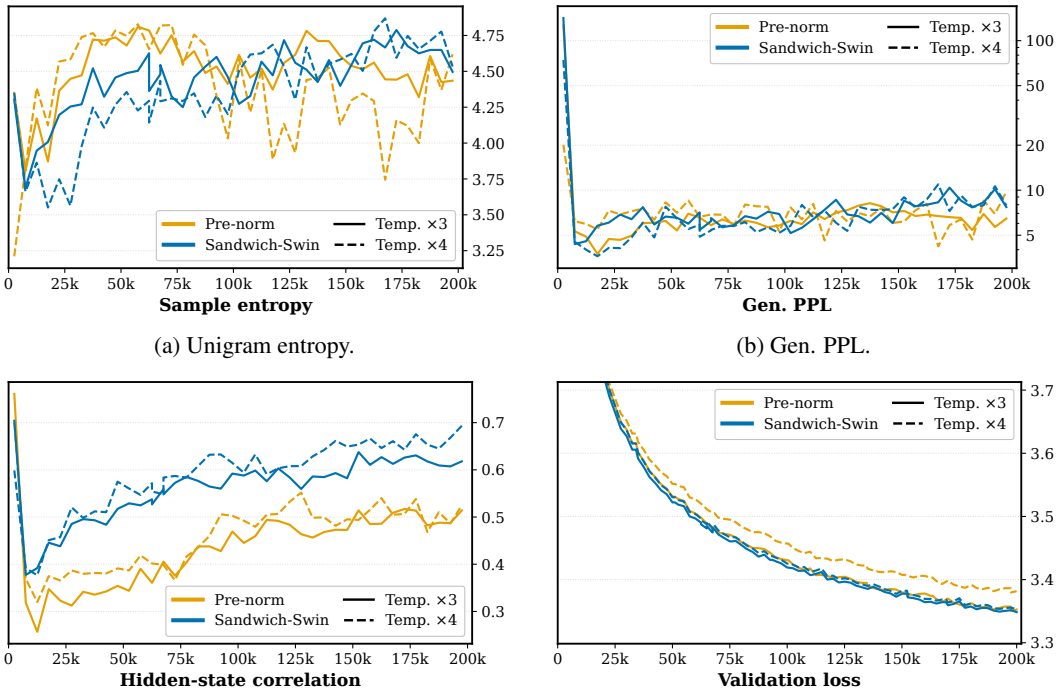
Backpropagation Through Passes. Figure 16 compares backpropagating through both passes with detaching the SC state between them (Suppl. A.6). Backpropagating through both passes makes training unstable for both block types. Between evaluations, the model oscillates between collapsed and partially recovered states, and both the entropy and the Gen. PPL fluctuate strongly. The hidden-state correlation also stays well above that of the runs with a stop-gradient. With a stop-gradient, all metrics evolve smoothly.

C.9 OpenWebText: Fine-Tuning

On TinyGSM, fine-tuning a pre-trained Duo checkpoint with LAW is enough to obtain strong results (Sec. 4.2). We first followed the same approach on OpenWebText, before training from scratch. We start from two Duo checkpoints trained on OpenWebText for 200k steps: one with the standard DiT architecture, i.e. pre-norm blocks, and one where the DiT uses Sandwich-Swin blocks instead (Suppl. A.5). The pre-norm checkpoint is an intermediate checkpoint of the Duo baseline of Sec. 4.4, which is trained for 400k steps in total. We fine-tune each checkpoint with LAW for 50k steps, keeping the block type of the checkpoint and using temperature regularization with $\rho = 3$. We also fine-tune variants with the mode embedding (Suppl. A.8). Since the mode embedding is initialized at zero, adding it to a Duo checkpoint does not change the model at the start of fine-tuning. As in Suppl. C.8, we evaluate every 5k steps with the EMA weights, using 512 samples drawn with NFE $= 256$ at $T = 1$, and report the corresponding Duo checkpoint as a reference.

In all configurations, the samples of the fine-tuned models remain low in diversity. The highest unigram entropy is 4.34, with a Gen. PPL of 12.9, obtained with Sandwich-Swin blocks, the mode embedding, and a stop-gradient between passes. Fine-tuning for 50k steps is therefore not sufficient on OpenWebText, which is why our main language modeling results (Sec. 4.4) use models trained from scratch (Suppl. C.8).

Block Type and Mode Embedding. Figure 17 compares both block types, with and without the mode embedding. At the start of fine-tuning, all models produce degenerate samples: the unigram entropy and the Gen. PPL are very low, and the hidden-state correlation is close to 1, indicating collapsed representations. The entropy then increases steadily, but remains well below that of the Duo donor checkpoint after 50k steps. Sandwich-Swin blocks recover faster than pre-norm blocks and reach a higher entropy after 50k steps. The mode embedding has little effect for either block type.



(a) Unigram entropy.

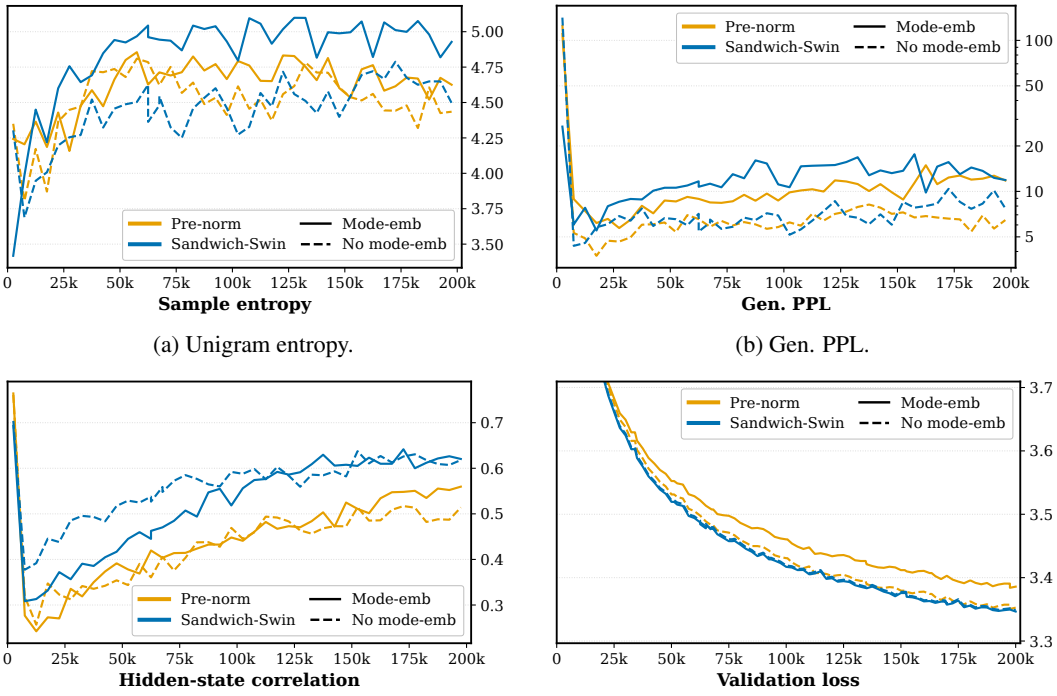
(b) Gen. PPL.

(c) Hidden-state correlation.

(d) Validation loss.

Figure 14: **Block type and temperature regularization.** Training curves of LAW with pre-norm (orange) and Sandwich-Swin (blue) blocks, trained with temperature regularization factor $\rho = 3$ (solid) or $\rho = 4$ (dashed), without the mode embedding. Samples are drawn with NFE $= 256$.

Backpropagation Through Passes. Starting from the Duo Sandwich-Swin checkpoint, we compare backpropagating through both passes with detaching the SC state (Suppl. A.6), with and without the mode embedding (Figure 18). Backpropagating through both passes reaches a slightly lower entropy and a hidden-state correlation that remains high. With the stop-gradient, the correlation decreases towards that of Duo. With the mode embedding, backpropagating through both passes also yields a higher Gen. PPL than Duo, despite a lower entropy.



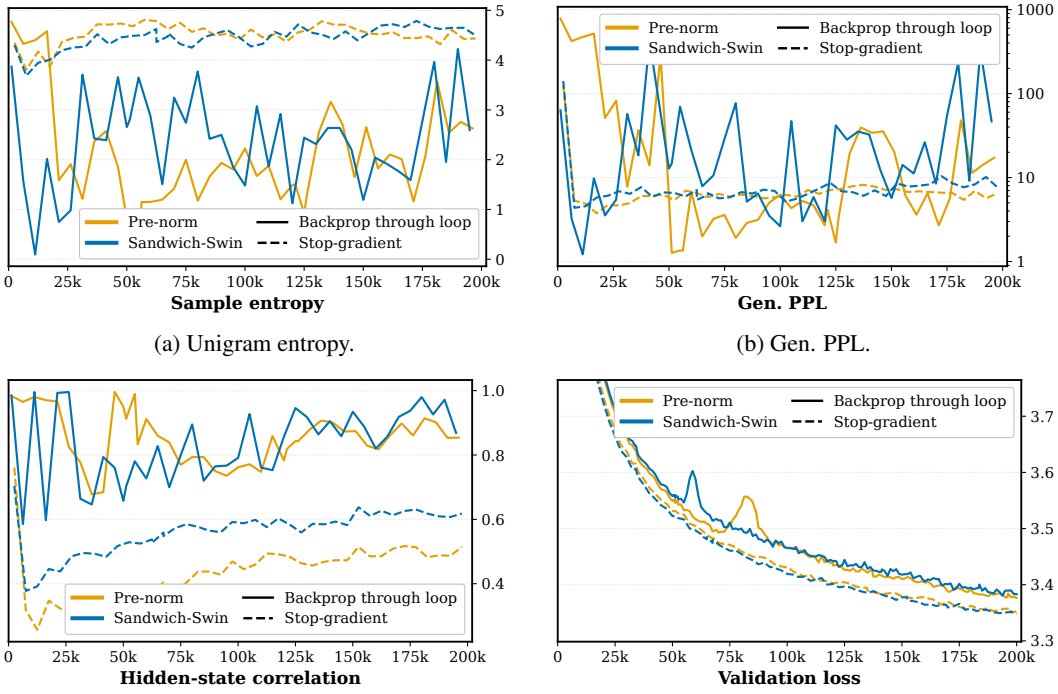
(a) Unigram entropy.

(b) Gen. PPL.

(c) Hidden-state correlation.

(d) Validation loss.

Figure 15: **Block type and mode embedding.** Training curves of LAW with pre-norm (orange) and Sandwich-Swin (blue) blocks, with (solid) and without (dashed) the mode embedding. All runs use $\rho = 3$, and samples are drawn with NFE $= 256$.



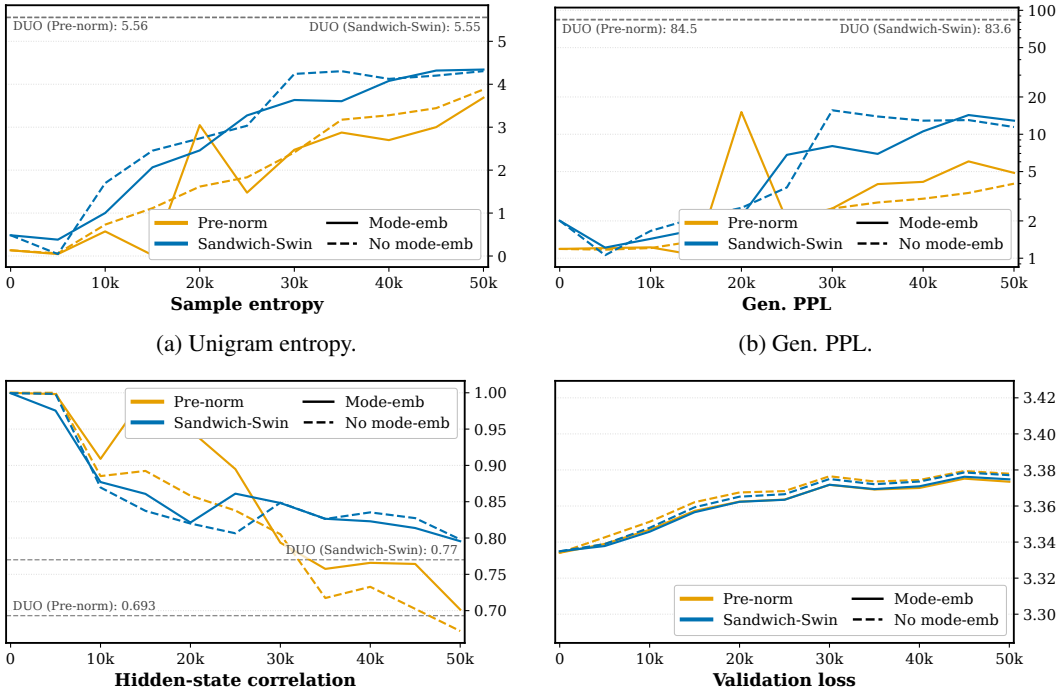
(a) Unigram entropy.

(b) Gen. PPL.

(c) Hidden-state correlation.

(d) Validation loss.

Figure 16: **Block type and gradient flow between passes.** Training curves of LAW with pre-norm (orange) and Sandwich-Swin (blue) blocks, backpropagating through both passes (solid) or detaching the SC state between them (dashed). All runs use $\rho = 3$ without the mode embedding, and samples are drawn with NFE $= 256$.



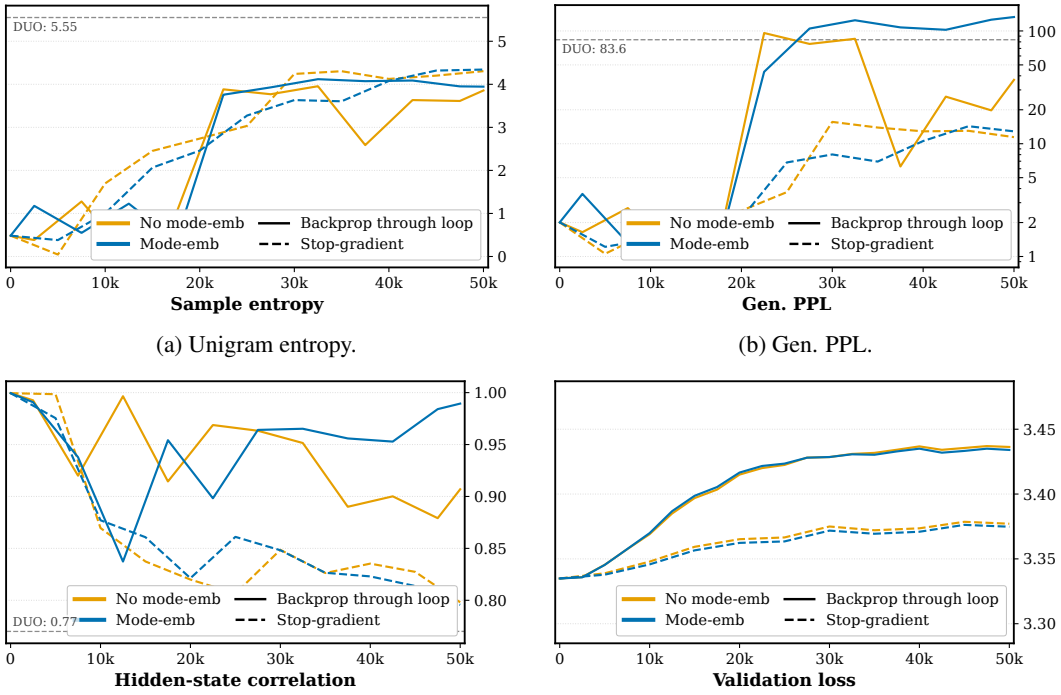
(a) Unigram entropy.

(b) Gen. PPL.

(c) Hidden-state correlation.

(d) Validation loss.

Figure 17: **Fine-tuning: block type and mode embedding.** Fine-tuning curves of LAW from Duo checkpoints with pre-norm (orange) and Sandwich-Swin (blue) blocks, with (solid) and without (dashed) the mode embedding. The horizontal lines in (a)–(c) are the corresponding Duo checkpoints. All runs use $\rho = 3$, and samples are drawn with NFE $= 256$.



(a) Unigram entropy.

(b) Gen. PPL.

(c) Hidden-state correlation.

(d) Validation loss.

Figure 18: **Fine-tuning: gradient flow between passes and mode embedding.** Fine-tuning curves of LAW from the Sandwich-Swin Duo checkpoint, backpropagating through both passes (solid) or detaching the SC state between them (dashed), with (blue) and without (orange) the mode embedding. The horizontal line in (a)–(c) is the Duo checkpoint. All runs use $\rho = 3$, and samples are drawn with NFE $= 256$.

C.10 OPENWEBTEXT: SAMPLES

We show unconditional samples of $L = 1024$ tokens from the LAW model of Sec. 4.4, trained from scratch on OpenWebText with Sandwich-Swin blocks (Suppl. A.5), a mode embedding (Suppl. A.8), and temperature regularization with $\rho = 3$ (12). The samples are drawn with sampling temperature $T = 1$.

<|endoftext|>, when the are will and from can be and before the said. ¶ "Integrated container systems is an exciting technology in the food delivery world," said Richard Fishter, vice president of network development at Compass Pointing. "We we truck and the processing of systems of and the our the delivery we with our in and food across and and convenience." ¶ CAPTION Uber CEO J.H. Hohmann delivers a speech at the Food Trucking Conference in San Jose, Calif., on Tuesday, May 21, 2017. Uber CEO J.T. Auerbach delivers a keynote at the Future Truck Conference,in San Jose, Calif., on Wednesday, May 21, 2017. Uber CEO J.T. Auerbach delivers a keynote at the Future Trucking Summit in San Jose, Calif., on Wednesday, May 20, 2017. CAPTION Lunch of includes Big King and Red duck sandwiches at Big Mac'sles Restaurant Restaurant San Francisco Bay CA Wed., 22 Ave,, at Chris by and, and Truck Truck ¶ ,., the Super truck The delivery trucks to deliver more than 400 pound pounds of fresh food delivered on truck by Michael T. Posted Mon Mon 23 Mar 2017 11:05am PDT Michael Easton, CEO, Tru Technologies Michael Beardson, who was the first person to fly an fully autonomous truck, is delivering 400 pounds of "Freshty King" sandwiches with Jacks Cheese and Mac's King Grillthe Lunch ¶ includes 100 King 400 Big King King's and sandwiches from Big Mac King's in at San Francisco Bayles's Wed Mon 23 Mar,'s Chef: Dr &,.,er and, aley,,,, for, St Fresh King Delers of at 1 Calif, C, in, ¶ ,,. ¶ and Systems Truck truck Fooda and is set to deliver more than 400 pounds pounds of fresh food delivered by truck by Michael T. Posted Mon Mon 23 Mar 2017 10:58am PDT "We don't know if automated delivery systems will be able to lift pounds of food out of one container and deliver it to people most safe delivery."."Truck Technologies has gone through rigorous and extensive testing to make sure our delivery system is operating to the highest standards of safety and predictability," CEO David Easton said.Truck uses its proprietary internal release system so the truck doesn't need to apply outside pressure to the container. Instead, air's be collected inside of the container and pressurized through a pressure pump controlled by a pressure analyzer.T Capacity Weight Weight: 190 the ¶ 000, truck ¶ A), of 8 from to's Bays truck- food delivery truck delivers 400 pound pounds fresh food through over 200 locations every day. The 400 pound haul of food travels through a variety of locations all over the country from California Francisco to Seattle. Food optionsings include: Big Big Mac's Egg Chickenamedes Sandwich with Jacks Cheese and Mac's ¶ .Theou amount pounds pounds delivery will has to in 5 grocery locations across the Golden Bay Area and is expected to go into orders by the end of next year. ¶ The truck is undergoing testing to determine if it of the food stored in its container safe enough to deliver served home to destination.Fullired Weight ::: +, 160- MORE for Jack truck Chef and delivery Truck delivery the The of truck to deliver more than 400 pound pounds of fresh food delivered by truck by Michael T. Posted Fri Mon 23 Mar 2017 11:33am PDT Michael Easton, CEO, Tru Technologies Michael Beardson, who was the first person to fly an fully autonomous truck, is delivering 400 pounds of "Freshty King" sandwiches with Jacks. Posted Fri Mon 23 Mar 2017 12:42am PDT Michael Easton, CEO, Tru Technologies Michael Beardson, who was the first person to fly an fully autonomous truck, is delivering 400 pounds of "Freshty Bacon" sandwiches with Jacks and and and.ainsTruck utilizes a patented internal safety system so the truck doesn't need to apply external pressure to the container. Instead, air is be compressed inside of the container is pressurized via a pressure chamber controlled by an air squeezer Posted Fri Mon 23 Mar 2017 2:52am PDT "We don't know how many delivery systems can lift up to 400 pounds of food out of one container and deliver it to customers in one delivery area," Operating CEO Jackner said."Truck Technologies has gone through rigorous, quality testing to make sure our delivery system is designed to meet the highest standards of safety and predictability," CEO David Easton said."The truck is undergoing testing to determine whether it holds all food packed in one container safe enough and be in to a destination."*icle Weight weights:: 140 52 K The 10 The Street's food's delivery truck delivers 400 pound pounds of fresh food through 50 different locations each day. The 400 pound pounds of food travels through a variety of locations all across the country from California Francisco to Seattle. Food optionsings include: Big's's and Chicken Sandwich with Jacks,<|endoftext|>

Figure 19: Sample (NFE $= 64$, $T = 1$) from LAW trained on OpenWebText, with unigram entropy 5.15.

<|endoftext|>seven pass rusher over the past two seasons, recording 123 tackles, three passes defensed and a forced fumbles against the 2012 Bay Saints. He currently ranks 219th on ESPN by 247Sports.com. ¶ Dandon Lawrence, DE, Ohio State ¶ The 6-foot-11, 295-pounder has recorded nine sacks and 106 tackles for loss the past two seasons, which is a marked improvement from last year's 116½ performance. He currently ranks 65th on ESPN by 247Sports.com. ¶ Devin Lawson, DE, Florida State ¶ The 6-foot-11, 305-pounder has shown consistent talent over his four-year career with the Crimson Tide, where he was selected first overall in the 1998 NFL Draft. He currently ranks 168th on ESPN by 247Sports.com. ¶ Shakeem Hunt, DT, Oklahoma State ¶ The 6-foot-3, 270-pounder has shown flashes potential as a top-five pass rusher over the past two seasons, recording 99 tackles, six passes defensed and a forced fumbles in a 33-16 win over the Tennessee Titans. ¶ Quarterfinals of the 2014 BCS College Football Championship Game will take place at 7 p.m. ET on FOX Sports 1 or FS2 from AT&T Stadium in San Antonio, Texas. The game will be broadcast live by UDFS.TV and nationally by FOXSports.com. ¶ Latavius Thomas, DT, Ohio State ¶ The 6-foot-11, 190-pounder has shown consistent production over his four-year career with the Buckeyes, where he was selected first overall in the 2004 NFL Draft. The 6-year-old has recorded 10 sacks and 63 tackles for loss the past two seasons, which marks a marked improvement over last season's 43-14 win over the New Orleans Saints. ¶ Tarell Boyd, DE, Michigan State ¶ The 6-foot-10, 180-pounder has shown consistent explosiveness the past two seasons before including a 25-19 victory against the Tennessee Titans. The 6-year-old has recorded seven sacks and 82 tackles for loss the past two seasons, which marks a marked improvement over last season's 45-31 victory over the New Orleans Saints. ¶ Joffrick Robinson, S, Arkansas Forest ¶ The 6-foot-4, 260-pounder has displayed consistent ability over a 15-year career with the Bucers, where he was selected first overall in the 1989 NFL Draft. He currently ranks 176th on ESPN by 247Sports.com and fifth by 247Sports Composite. ¶ ZQuron Robinson, DE, Ohio State ¶ The 6-foot-5, 275-pounder has shown flashes potential as a top-10 pass rusher over the past three seasons, recording 90 tackles, two passes defensed and two forced fumbles in a 30-14 win over the Pittsburgh Raiders. ¶ Quarterfinals of the 2014 BCS College Football Championship Game will take place at 7 p.m. ET on FOX Sports 1 or FS2 from AT&T Stadium in San Antonio, Texas. The game will be broadcast live by UDFS.TV and nationally on FOXSports.com. ¶ Clorman Williams, OT, Ohio State ¶ The 6-foot-3, 205-pounder has shown consistent talent over his four-year career with the Crimson Tide, where he was selected first overall in the 1995 NFL Draft. He currently ranks 180th on ESPN by 247Sports.com. ¶ Michael Gordon, DT, Arizona State ¶ The 6-foot-4, 225-pounder has shown flashes potential as a top-10 pass rusher over the past two seasons, recording 99 tackles, seven passes defensed and two forced fumbles in a 26-23 win over the Cincinnati Bears, per 247Sports Composite. ¶ Mroy Miller, DT, Florida State ¶ The 6-foot-1, 305-pounder has shown flashes potential as a top-10 pass rusher over the past two seasons, recording 84 tackles, eight passes defensed and two forced fumbles in a 31-26 win over the Tennessee Titans. The 19-year-old has shown consistent explosiveness since coming out of Stanford, where he was selected ninth overall in the 1989 NFL Draft. He currently ranks 99th on ESPN by 247Sports.com. ¶ Jordrick Alexander, NT, Ohio State ¶ The 6-foot-4, 310-pounder has shown flashes potential as a top-10 pass rusher over the past two seasons, recording 101 tackles, four passes defensed and a forced fumbles in a 33-17 win over the Baltimore Titans, per 247Sports Composite. ¶ Trevor Adams, DE, Iowa State ¶ The 6-foot-11, 340-pounder has shown flashes potential as a top-10 pass rusher over the past two seasons, recording 102 tackles and two passes defensed in<|endoftext|>

Figure 20: Sample (NFE $= 1024$, $T = 1$) from LAW trained on OpenWebText, with unigram entropy 4.83.