ON THE DESIGN SPACE OF CONTINUOUS DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous diffusion language models (CDLMs) promise to bring the benefits of continuous flow modelling to language, including accelerated sampling and controllable generation. The field has grown rapidly, and the capabilities of CDLMs have begun to rival those of discrete diffusion. Yet, unlike for continuous modalities such as images or videos, our grasp of the levers that matter when training CDLMs remains limited. In this work, we focus on two of these levers, namely self-conditioning and embedding geometry, and shed light on some of the mechanisms behind them, which in turn suggest cheaper and more effective designs. We begin by showing that self-conditioning corrects a train/inference mismatch by exposing the model to inference-like states that are otherwise rare during training. These gains can be recovered without an extra forward pass by a two-level interpolant, coined $\mathrm{DoubleCat}$, which improves TinyGSM accuracy more than threefold. Self-conditioning also aids in recognising tokens dominated by large-norm Gaussians—an issue known as token identifiability. To address it directly, we introduce discrete reveals to $\mathrm{DoubleCat}$, yielding $\mathrm{SemiCat}$, which achieves discrete diffusion-level performance on TinyGSM. At the 1.7B scale, $\mathrm{SemiCat}$ outperforms discrete diffusion on four of six likelihood-based evaluations, and surpasses one-hot flows on unconditional generation. On embedded spaces, we find that cross-entropy drives embeddings towards a geometry whose decoding error rate closely matches the one-hot one. We briefly explore a range of further options, such as hybrid corruption processes and noise schedules, including asynchronous ones. We believe this exploration can help the community better understand and advance the design space of CDLMs.

1 INTRODUCTION

Continuous diffusion language models (CDLMs) hold several promises for text generation. Ideally, they should port to the world of language modelling all of the key properties of the continuous toolbox, such as fast and controllable generation, consistency models (Song et al., 2023; Geng et al., 2025; Boffi et al., 2025b), or stochastic optimal control-based steering (Potaptchik et al., 2026a; Holderrieth et al., 2026; Potaptchik et al., 2025). Yet, some of the seminal advances that have managed to shrink the performance gap between continuous and discrete diffusion methods have relied instead on adaptations that deviate significantly from the continuous diffusion literature. One such technique is *self-conditioning* (Chen et al., 2023), which augments inputs using previous outputs, costing an additional forward pass during training. Yoo et al. (2026) interpret self-conditioning as a fixed-point iteration, and Shabalin et al. (2026) understand it through the lens of multi-modality. Yet, it remains unclear which deficiencies of CDLMs it corrects, and whether these can be addressed more cheaply. Another innovative line of development lies in embedded methods (Deschenaux & Gulcehre, 2026; Chemseddine et al., 2026; Hu et al., 2026; Chen et al., 2026; Batzolis et al., 2026), which alleviate the computational and memory cost of one-hot methods (Roos et al., 2026; Lee et al., 2026; Potaptchik et al., 2026b). They rely on carefully chosen noise schedules. However, it remains unclear how such schedules interact with the learnt embeddings.

In this work, we show first that self-conditioning addresses a discrepancy between inference- and train-time trajectories (Section 3.1), and increases the density of otherwise under-sampled regions at train-time (consistent with Shabalin et al. 2026). This gain can be recovered by a simple two-level interpolant, called $\mathrm{DoubleCat}$, which requires no additional forward pass of the model, and which

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.

more than triples the accuracy on TinyGSM (Liu et al., 2023) from 4.42% to $\mathbf{16.00}\%$. Moreover, we find that self-conditioning helps in *token identifiability* (Pynadath et al., 2026) (Section 4.3): given the noisy state, the model cannot easily identify the expected endpoint of the trajectory, due to the norm mismatch between the noise and the token's embedding, whereas self-conditioning is supported on the convex hull formed by the embeddings. Replacing that signal with a better representation of the state recovers a substantial part of the performance gain at no additional training cost.

We also address the token identifiability issue in $\mathrm{DoubleCat}$ (Section 3.2). By introducing discrete stochastic reveals akin to CANDI (Pynadath et al., 2026), the model is able to condition correctly on parts of its context, helping identifiability. This method, called $\mathrm{SemiCat}$ (semi-continuous categorical flow), further boosts the performance up to $\mathbf{32.90}\%$, *reaching the discrete diffusion baseline of about* 33.74%. At the 1.7B scale, $\mathrm{SemiCat}$ outperforms discrete diffusion models on 4 out of 6 likelihood-based benchmarks and matches them on a fifth one; it also surpasses one-hot flows in unconditional generation quality (Section 5).

We also cover embedded methods. Because of architectural decisions, one-hot flows effectively operate on an embedded space with an anisotropic noise defined by the embeddings' geometry (Section 4.1), on which we can *directly train*. We demonstrate that embeddings may not be trained with cross-entropy naively, as it maximises their spread. *This itself induces a one-hot-like geometry on the embedded space when considering its corresponding decoding error rate* (Section 4.2). This may explain why carefully crafted schedules are needed. We argue that training should update the geometry of all embeddings jointly.

Lastly, we briefly explore other avenues such as asynchronous schedules, dynamic time allocation, hybrid methods, and acceleration, thus covering a large part of the design space of CDLMs. We believe that our results on our benchmarks show the promise of continuous and hybrid methods, and unveil important avenues left to explore to understand continuous diffusion for discrete data.

2 Background

Let $V \subset \mathbb{N}$ be the finite vocabulary of tokens, and $\mathbf{x} = (\mathbf{x}^1, \dots, \mathbf{x}^L) \in V^L$ a sequence of length $L \in \mathbb{N}^\star$. Our goal is to model some data distribution, p_{data}, given independent and identically distributed samples therefrom. Model parameters are denoted by θ. A uniform distribution over I is denoted by $\mathcal{U}(I)$. We defer to Appendix C.1 a brief presentation of discrete diffusion models.

Flow-based generative modelling. A common option for the modelling of p_{data} is through the dynamic transport of measure (Lipman et al., 2023; Liu et al., 2022; Albergo et al., 2023). Given a simple starting distribution, p_0, and a target distribution $p_1 \equiv p_{\mathrm{data}}$, the aim is to create a probability path between the two, $(p_t)_{t\in[0,1]}$, via an ODE: $\dot{x}_t = v_t(x_t),$ with $x_0 \sim p_0$.

Let $I : [0,1] \times \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}^d$ be an interpolant of the form $I_t(X_0, X_1) = \alpha_t X_0 + \beta_t X_1$, for some differentiable α and β, $X_0 \sim p_0$, $X_1 \sim p_1$, and such that $I_0(X_0, X_1) = X_0$ and $I_1(X_0, X_1) = X_1$. By choosing $v_t(x) = \mathbb{E}[\dot{I}_t \mid I_t = x]$, then $\mathrm{Law}(x_t) = \mathrm{Law}(I_t)$ for any t, making x_t available *simulation-free* through I_t. It thus reduces the problem of generative modelling to the regression:

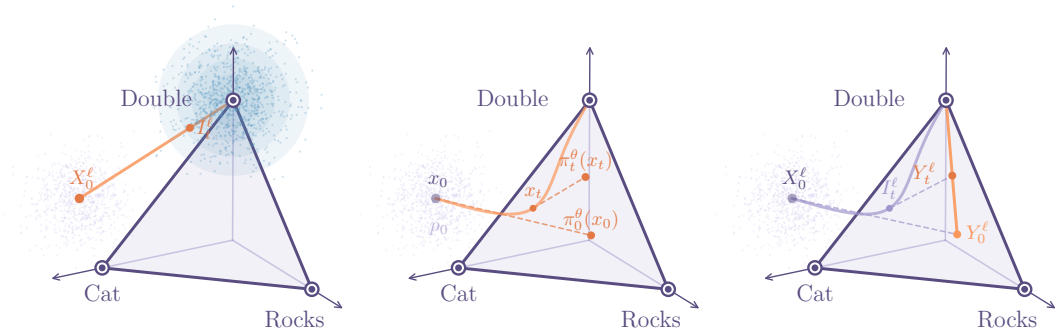
$$\mathcal{L}(\theta) = \mathbb{E}_{X_0, X_1, t} \left\| v_t^\theta(I_t) - \dot{I}_t \right\|_2^2. \tag{1}$$

Sampling is done by integrating v^θ. When $\beta_t \equiv 1 - \alpha_t$, we write $I_t(X_0, X_1) = (1-\gamma_t)X_0 + \gamma_t X_1$.

Flow-based language modelling. Flow modelling has also been adapted to the setting where $p_{\mathrm{data}}$is a discrete distribution in V^L (Eijkelboom et al., 2025; Roos et al., 2026; Lee et al., 2026; Potaptchik et al., 2026b). One choice of modelling is that of *one-hot flows*, where the final distribution is one-hot encoded into $\mathbb{R}^{|V|}$ (*i.e.*, $X_1^\ell = e_{\mathbf{x}^\ell}$). Using the endpoint parametrisation of the generating vector field, $v_t(x) = \dot{\gamma}_t \frac{\mathbb{E}[X_1 | I_t = x] - x}{1 - \gamma_t}$, it is possible to train using a cross-entropy loss:

$$\mathcal{L}(\theta) = -\mathbb{E}_{X_0, X_1, t} \sum_{\ell=1}^{L} \log \left\langle X_1^\ell, \pi_t^{\theta,\ell}(I_t) \right\rangle, \tag{2}$$

where $\pi^\theta : [0,1] \times \mathbb{R}^{L\times|V|} \longrightarrow \mathbb{R}^{L\times|V|}$ is a denoising network. In this work, we also consider the possibility of evolving the flow on an embedded space, which is first embedded through a trainable



(a) One-hot flow, train-time. (b) One-hot flow, inference. (c) DoubleCat, train time.

Figure 1: **(a)** One-hot flows are always exposed to nearly clean samples for $t \approx 1$, and only track the clean target, X_1^ℓ. **(b)** At inference, they track instead the dynamic predictions of their denoiser, π^θ, which always lies on the interior of the simplex. **(c)** DoubleCat addresses this issue at train-time by simply adding an additional corruption process on the output space of its denoiser, the simplex.

codebook (Deschenaux & Gulcehre, 2026; Chemseddine et al., 2026; Hu et al., 2026; Chen et al., 2026). We denote by $\mathcal{E} \subseteq \mathbb{R}^d$ the embedded space, with $d \leq |V|$, and by $\mathsf{emb} : V \longrightarrow \mathcal{E}$ the embedding function. Let W be the matrix such that $W_v = \mathsf{emb}(v)$.

Time schedule. The choice of time schedule is crucial for one-hot flows (Lee et al., 2026; Potaptchik et al., 2026b). Following Davis et al. (2026); Potaptchik et al. (2026b), we refer to and use by default for one-hot flows a *mixture schedule*, $\gamma_t = \lambda\gamma_t^{\mathrm{DER}} + (1-\lambda)t$, where γ^{DER} is the *decoding error rate* (DER) schedule (Pynadath et al., 2026; Sahoo et al., 2025; Lee et al., 2026), with $\lambda = 0.75$. It makes uniform progress in the DER (Pynadath et al., 2026, eq. (10)):

$$P_e(t) \coloneqq \frac{1}{L}\sum_{\ell=1}^{L} \mathbb{P}(\operatorname{argmax} I_t^\ell \neq \operatorname{argmax} X_1^\ell). \tag{3}$$

Hybrid methods. Pynadath et al. (2026) introduce a hybrid flow, CANDI, that continuously evolves each state of the sequence until some randomly sampled unmasking time, τ^ℓ, per position. Formally, for each ℓ, we let i.i.d. $\tau^\ell \sim \mathcal{U}(0,1)$; for $t < \tau^\ell$, $I_t^\ell(X_0, X_1) = (1-\gamma_t)X_0^\ell + \gamma_t X_1^\ell$, and $I_t^\ell(X_0, X_1) = X_1^\ell$ otherwise. The clean token is directly revealed as a one-hot encoding at τ^ℓ.[1] Agarwal et al. (2026) introduce a similar construction via Flow Maps (Boffi et al., 2025b;a).

Plaid. Introduced in Gulrajani & Hashimoto (2023) and recently revisited in Yang et al. (2026), Plaid is a Variational Diffusion Model (VDM; Kingma et al., 2021) for language modelling, trained by minimising the negative evidence lower bound (NELBO):

$$\begin{aligned} -\log p_\theta(\mathbf{x}) \leq \quad & \mathcal{L}_{\mathrm{NELBO}}(\theta) \coloneqq \mathrm{KL}\left(q(X_0 \mid X_1) \,\|\, p(X_0)\right) - \mathbb{E}_{X_1} \textstyle\sum_{\ell=1}^{L} \log \left\langle X_1^\ell, \pi_t^{\theta,\ell}(I_t) \right\rangle \\ & + \tfrac{1}{2}\mathbb{E}_{t,I_t}\left[\mathrm{SNR}'(t)\, \|\mathsf{e}_\theta(I_t, t) - \mathsf{emb}(\mathbf{x}_1)\|^2\right], \end{aligned} \tag{4}$$

where $\mathbf{e}_\theta : \mathcal{E}^L \times [0,1] \longrightarrow \mathcal{E}^L$ is given by $\mathbf{e}_\theta^\ell : (x,t) \longmapsto \pi_t^{\theta,\ell}(x)W$ for each ℓ, $p \equiv p_0$ (Gaussian prior), and SNR' being the derivative of the *signal-to-noise ratio*, $\mathrm{SNR} : [0,1) \longrightarrow \mathbb{R}, t \longmapsto \beta_t^2/\alpha_t^2$. The schedules are learnt, following VDM.

Self-conditioning. Self-conditioning has been shown to be extremely beneficial for CDLMs (Chen et al., 2023; Hu et al., 2026; Chen et al., 2026; Batzolis et al., 2026). It extends the denoiser to accept an additional input, $\tilde{\pi}^\theta : [0,1] \times \mathbb{R}^d \times \mathbb{R}^d \longrightarrow \mathbb{R}^d$, and feeds the model its outputs from the previous inference step when generating samples. At train time, it is provided to the model with a probability p_{sc}: let $C \sim \mathrm{Ber}(p_{\mathrm{sc}})$, let $\hat{x}_1 = C \cdot \mathrm{sg}(\tilde{\pi}_t^\theta(I_t, \mathbf{0}))$ be the self-conditioning, where sg is the stop-gradient operator, training then uses $\tilde{\pi}_t^\theta(I_t, \hat{x}_1)$.

[1] Note the reversal in time convention to match the flow matching one, and note that we here ignore the difference in interpolants incurred by typical diffusion schedules.

3 ENHANCING ONE-HOT FLOWS

In this section, we identify two issues that one-hot flows suffer from. First, they lack trajectory ambiguity: they are trained on clean sequences at $t \approx 1$, which mismatches the statistics of their inference trajectories. Second, they are unable to identify the tokens that they are generating, which we attribute to the high norm Gaussian noise. We show two simple interventions, DoubleCat and SemiCat, respectively, both of which drastically improve the resulting performance.

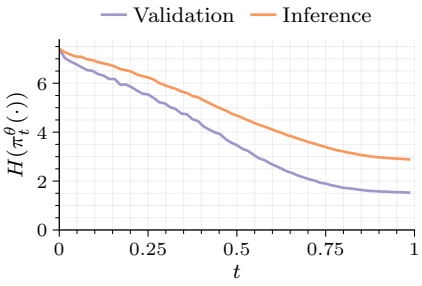


Figure 2: Entropy of a one-hot flow's denoiser, π^θ, along validation set and inference trajectories.

Benchmark. We mostly rely on TinyGSM (Liu et al., 2023): the model is trained on generating Python solutions to grade school mathematical problems, and is then evaluated on GSM8K (Cobbe et al., 2021). To understand failure modes of trained models, we measure the percentage of programs that run, the percentage that have name errors (referencing non-existent variables), and the final accuracy. We use the settings of Kim et al. (2026). Finally, retained methods are also evaluated on the standard OpenWebText (OWT) (Gokaslan & Cohen, 2019) unconditional generation task on which we report the generative perplexity (Gen. PPL) as measured by GPT-2 (Radford et al., 2019) alongside the entropy of generated samples, reported as Pareto fronts. See Appendix D.1 for more detail.

3.1 TRAJECTORY AMBIGUITY

For a one-hot flow, at times near one, the model is only trained on sequences in a very small radius around the final endpoint. Concretely, near $t \approx 1$, I_t is a highly concentrated Gaussian with covariance $(1-\gamma_t)^2\mathrm{I}$ and mean $\gamma_t X_1$. At inference time, the model follows the learnt denoiser, π^θ, which has non-zero entropy for any t (Figure 2), and therefore never points towards a clean vertex of the simplex. This creates a mismatch between inference and training, where the model, instead of following a prediction for X_1^ℓ, follows the path defined by the denoiser $(\pi_t^\theta(x_t))_{t\in[0,1]}$ on the interior of the simplex. On the other hand, self-conditioning provides a signal with the same support as the denoiser, which perturbs the Gaussian inputs towards inference-like regions (Shabalin et al., 2026).

Double continuous corruption. We introduce DoubleCat, a doubly categorical flow, with a corruption process on the simplex, tracked by a normal Gaussian one. Formally, we define

$$\begin{aligned} Y_t^\ell &= (1-\kappa_t)Y_0^\ell + \kappa_t X_1^\ell \\ I_t^\ell(X_0, X_1) &= (1-\gamma_t)X_0^\ell + \gamma_t Y_t^\ell, \end{aligned} \tag{5}$$

for some Y_0^ℓ prior, and κ a differentiable schedule. Note that this forms a valid probability path between p_0 and p_1. For inference, we use the same denoiser for both parts of the flow as an approximation. If we choose Y_0^ℓ to have mean $\frac{\mathbf{1}}{|V|}$, then the entropy evolves in a roughly similar fashion from $H(Y_0^\ell) = \log|V|$ to $H(Y_1^\ell) = 0$. We ablate the choice of prior in Table 5, and find the uniform distribution over the simplex to work best. We also ablate the schedule κ in Table 6, and retain $\kappa : t \longmapsto \sqrt{t}$. Figure 1 illustrates our intuition.

Results. The results on TinyGSM are available in Table 1. DoubleCat's performance improves by more than $3\times$ over one-hot flows. Discarding the Y_t term at inference—*i.e.*, sampling as in a usual one-hot flow—further improves the performance: up to over $4\times$ (with $T = 0.2$), with an accuracy reaching 16.00%, name error percentages lower by 18 points, and valid programs almost 19 points higher. This suggests that DoubleCat imitates the dynamics at inference of a one-hot flow, which tracks the denoiser's trajectory, $(\pi_t^\theta(x_t))_{t\in[0,1]}$, and that the simplex noise, Y_0^ℓ, biases sampling.

We also show that this mechanism, while being substantially cheaper computationally, outperforms plain self-conditioning. Nonetheless, we report that self-conditioning still improves DoubleCat too (up to 20.77% at $T = 0.5$, see Table 7); we further explore self-conditioning in Section 4.3.

Method	Accuracy (%, ↑)				Best (%)	
	$T = 1$	$T = 0.5$	$T = 0.2$	top-k=1	Runs (↑)	Name err. (↓)
Discrete diffusion baselines						
MDM	8.79	21.68	27.14	28.28	88.63	11.22
USDM	**12.43**	**29.34**	**33.06**	**33.74**	**91.96**	**7.81**
One-hot based						
Base one-hot flow	3.28	4.42	3.89	3.64	50.91	48.07
with self-cond.	6.44	6.75	8.34	6.97	61.79	37.60
DoubleCat	9.70	13.50	14.18	14.78	63.61	36.16
Base one-hot sampling	**11.45**	**15.85**	**16.00**	**15.92**	**69.67**	**30.02**
Hybrid methods, one-hot						
CANDI (Pynadath et al., 2026)	7.20	9.55	9.63	8.72	69.22	29.26
FMLM+ (Agarwal et al., 2026)	16.83	18.88	18.27	18.50	**90.52**	**9.02**
SemiCat	**17.51**	**29.34**	**32.22**	**32.90**	89.23	10.39

Table 1: TinyGSM results at 256 NFEs. Accuracy is reported per sampler; the last two columns report the best value attained over those samplers. T denotes the temperature applied to π^θ.

Takeaway. Self-conditioning increases the density of highly likely inference regions during training, diminishing the train–test mismatch. DoubleCat recovers a large part of the benefit by introducing a double-level flow, imitating self-conditioning.

3.2 Token identifiability

Despite the improvement offered by DoubleCat, the model still heavily suffers from name errors (30.02%). We attribute this to *token identifiability* (Pynadath et al., 2026): when referencing another variable, the model struggles to identify current tokens because of the large noise in x_t, and therefore cannot condition on it precisely: the Gaussian noise has an expected norm $\sqrt{|V|}$, but the signal has a norm of 1.

To mitigate this issue, we add a *revealing* mechanism akin to the one in CANDI: at random times τ^ℓ, a token is revealed as a one-hot input. This way, the model can cleanly condition on revealed inputs. The resulting method is SemiCat, for which we provide more formal details in Appendix B. In Table 1, SemiCat produces only 10.39% runs with name errors and programs that run 89.23% of the time, *and reaches the discrete diffusion level with an accuracy of* **32.90**%, on TinyGSM. We note that revealing inputs à la CANDI is not sufficient on its own. On OWT, SemiCat also dominates the Gen. PPL/Entropy Pareto frontier of one-hot flows at all NFEs (Figure 5).

Takeaway. Large Gaussian noise makes the tokens hard to identify from the current state, x_t. Introducing reveals like CANDI considerably helps in this regard.

While SemiCat obtains high performance, its stochastic reveals and piecewise continuous trajectories make it a harder candidate to accelerate using typical *continuous* techniques. We discuss in Appendix F.2 the possibility of designing asynchronous schedules (*i.e.*, schedules per token position, ℓ) to alleviate this issue. Besides, we briefly explore more general hybrid corruption processes in Appendix D.5, and show that SemiCat can obtain competitive performance compared to FMLM+ when distilling our model using D-MMD (Hoogeboom et al., 2026; Salimans et al., 2024) in Appendix D.6, in the few-step regime.

4 Designing embeddings

One-hot flows are costly, especially given that modern LM vocabulary sizes can exceed 200k tokens, and incur a large memory and computational penalty, requiring the materialisation and manipulation of an $L \times |V|$ tensor per data point in a batch. It is therefore natural to depart from this representation

Method	Accuracy (%, ↑)				Best (%)	
	$T = 1$	$T = 0.5$	$T = 0.2$	top-k=1	Runs (↑)	Name err. (↓)
Ambient sphere						
$d = 16$	6.97	9.40	8.95	8.87	58.83	40.71
$d = 128$	7.88	9.25	9.63	10.24	56.48	42.76
$d = 768$	**8.64**	**12.89**	**13.12**	**12.28**	**66.72**	**32.75**
Plaid, $d = 16$						
self-cond.	**19.71**	**28.43**	11.30	2.81	**84.38**	9.17
without self-cond.	8.19	10.84	11.98	**11.14**	72.25	26.91
Plaid, $d = 128$						
self-cond.	17.89	22.44	**16.22**	10.46	82.94	12.66
without self-cond.	6.75	9.55	9.48	9.78	62.77	36.54
Plaid, $d = 768$						
self-cond.	17.97	24.56	12.96	5.00	78.32	**3.56**
without self-cond.	4.62	5.23	8.79	8.04	57.70	41.77
Plaid variants, $d = 16$*, without self-cond.*						
with posterior DoubleCat	12.89	15.85	13.87	14.18	74.07	25.32
stochastic codebook	9.55	16.15	18.35	18.42	68.31	31.08
with posterior as self-cond.	10.84	16.53	17.97	18.35	67.10	32.07
self-cond. at inference	**16.60**	**18.04**	**19.86**	**20.24**	**76.12**	**23.12**

Table 2: TinyGSM results at 256 NFE. Plaid learns its log-SNR endpoints; the variants all pin them. "Without self-cond." refers to checkpoints trained without self-conditioning.

and work on smaller embedded spaces. We show that their DER can become problematic with pure cross-entropy training, and that self-conditioning aids in token identifiability.

4.1 Your one-hot flow is secretly embedded

The public one-hot flows for language modelling (Roos et al., 2026; Lee et al., 2026) use the adapted DiT architecture (Peebles & Xie, 2023) from Lou et al. (2024), with a simple linear projection for the input, which we denote by $W \in \mathbb{R}^{|V| \times d}$. This means that the first layer of the architecture after the projection receives an input distributed as

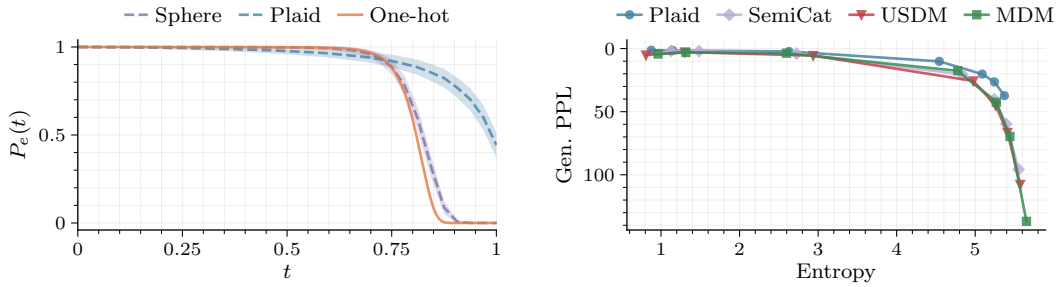
$$\forall \ell \in \{1, \dots, L\}, \qquad (I_t W)_\ell \sim \mathcal{N}\left(\gamma_t X_1^\ell W, (1-\gamma_t)^2 W^\top W\right). \tag{6}$$

This effectively shows that one-hot flows operate on an embedded space defined by their projection matrix, but which also modifies the injected noise, giving it a covariance structure defined by the Gram matrix of the embeddings, $W^\top W$. Note that Pynadath et al. (2026) make a similar observation and use it to accelerate inference. This naturally holds for DoubleCat and SemiCat too. We discuss how to *train* SemiCat directly on the embedded space in Appendix E.1, and attain the same end performance but with a more unstable training due to a Cholesky decomposition.

4.2 Your embedded flow is secretly one-hot

Considering embedded methods, we know that DoubleCat and SemiCat on the embedded space from Section 4.1 obtain high performance, and Plaid (Gulrajani & Hashimoto, 2023) does too. In contrast, denoted as "ambient sphere", we train (with default cross-entropy) a simple flow interpolating between a Gaussian and the final unit-normalised embedding, *i.e.*, $\|\mathsf{emb}(v)\| = 1$ for any $v \in V$. Mathematically, we write $I_t^\ell(X_0, X_1) = (1-t)X_0^\ell + t\mathsf{emb}(X_1^\ell)$. In Table 2, the ambient sphere underperforms Plaid with self-conditioning; besides, we note that the ambient sphere also largely underperforms on OWT, with a very high Gen. PPL (beyond 200) at matched data entropy.

The axis that helps us understand these results is the training objective. The simple cross entropy objective of the ambient sphere is pathological in two ways: *(i)* The model only updates the embeddings of the tokens involved in the batch, thus not updating the entire geometry of $\mathcal{E}$. Plaid updates all embeddings through its MSE diffusion term, and SemiCat and DoubleCat do so through the covariance structure of their embedded noise. *(ii)* Cross-entropy ignores the geometry of the embeddings, and spreads tokens apart on the sphere approximately recovering the one-hot geometry in the



(a) Decoding error rates, P_e.

(b) Gen. PPL/Entropy Pareto fronts on OWT.

Figure 3: DER of embedded geometries, and the quality/diversity trade-off across methods on OWT. **(Left)** The DER of a trained sphere model on OWT and that of Plaid both with $d = 768$, against the known one-hot DER. **(Right)** Gen. PPL/Entropy Pareto frontiers on OWT, with Plaid for $d = 16$.

DER sense (Pynadath et al., 2026; Sahoo et al., 2025), *i.e.*, $P_e(t)$ tends towards that of the one-hot case. To see this, observe that the DER for embeddings can be generalised to

$$P_e(t) \coloneqq \frac{1}{L}\sum_{\ell=1}^{L} \mathbb{P}(\operatorname{argmin}_v \left\| I_t^\ell - \mathsf{emb}(v) \right\|^2 \neq \operatorname{argmax} X_1^\ell). \tag{7}$$

Proposition 1 exhibits that the DER (7) depends solely on the interactions $\mathsf{emb}(j) - \mathsf{emb}(i)$ and $\langle \mathsf{emb}(i), \mathsf{emb}(j) \rangle$ for any $i, j \in V$. In Appendix E.3, we informally discuss how the cross-entropy term separates tokens, while the diffusion term of Plaid adds an attraction component. Moreover, Appendix E.2 shows that unit-norm embeddings maximally spread in dimension d have inner product terms that tend to 0 as d increases, and in turn the DER tends to the one-hot case.

Empirically, in Figure 3a, we measure through Monte Carlo estimates the DER, P_e, across time for a trained ambient sphere and a Plaid model ($d = 768$) on OWT. The measured and one-hot DER curves are indistinguishable except for a small gap at $t \approx 0.85$. Remark that Plaid's embeddings never induce $P_e(t) = 0$ even at $t = 1$ due to the learnt VDM schedule endpoints (see Appendix C.2). At $d = 768$, the ambient sphere's spreading of tokens makes them more identifiable, yielding lower name errors and higher run rate when compared with Plaid without self-conditioning.

Takeaway. Cross-entropy training spreads tokens apart, inducing the same DER as a one-hot flow. This may explain why prior work requires more advanced schedules (Deschenaux & Gulcehre, 2026; Hu et al., 2026; Chen et al., 2026); an alternative is updating the geometry of all embeddings jointly.

4.3 Self-conditioning on embedded spaces

In Table 2, the performance of the Plaid runs without self-conditioning *monotonically worsens* with d. This substantiates our findings: in higher dimensions, the discrepancy between the train- and inference- time statistics, I_t^ℓ and x_t^ℓ, is further widened. We then turn to adapting DoubleCat.

Naively porting DoubleCat to Plaid, by simply tracking $Y_0^\ell = \mathbf{v}^\top W$ until the true $Y_1^\ell = \mathsf{emb}(X_1^\ell)$, for $\mathbf{v} \sim \mathcal{U}(\Delta^{|V|})$, breaks. This happens because the model is given an input noisier than expected by the statistics of the ELBO, and applying the large weight SNR' to a non-zero loss makes the model diverge. To solve this, we attempted two techniques that provide more stability. Note that we freeze the endpoints of the learnt VDM schedule for additional stability.

The first one avoids having non-zero terms near $t \approx 1$: we instead flow the model towards its token-wise Bayesian posterior expectation at that point, $I_t^\ell = x_t^\ell$, and time t, assuming a uniform prior over the tokens and unit-normalised embeddings. That is to say, we set

$$D_t^\ell \coloneqq \mathbb{E}\left[\mathsf{emb}(X_1^\ell) \mid I_t^\ell = x_t^\ell\right] = \operatorname{softmax}_{v \in V}\left(\frac{\gamma_t \left\langle \mathsf{emb}(v), x_t^\ell \right\rangle}{(1-\gamma_t)^2}\right) W, \tag{8}$$

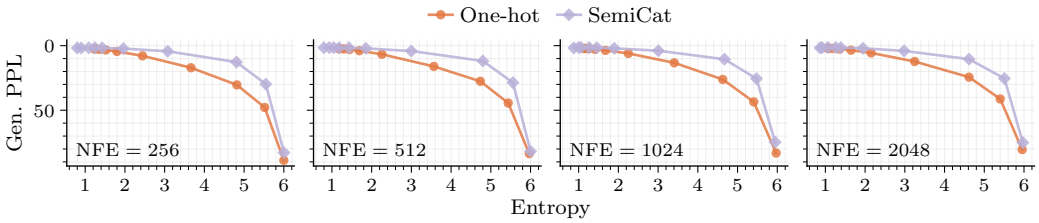


Figure 4: Gen. PPL as measured by Qwen 2.5 7B for models trained on Nemotron with sequence lengths of 2048 tokens, using the TikToken tokeniser.

and construct the input instead as $\tilde{I}_t^\ell = (1-\gamma_t)X_0^\ell + \gamma_t D_t^\ell$. Note that we recover similar statistics, since Y_0^ℓ is deterministically $\frac{\mathbf{1}}{|V|}^\top W$ and $Y_1^\ell = \mathsf{emb}(X_1^\ell)$. As shown in Table 2, this modestly improves over the base 11.98%, reaching 15.85%. The second one is denoted by "stochastic codebook" in Table 2, where we defined embeddings to be Gaussians, regularised to $1/d$ variance per coordinate and unit-norm mean. The goal is to increase the density of under-trained regions by this added stochasticity. This is most effective at low temperatures, but does not recover the self-conditioning performance. This resembles the noising tricks of Hu et al. (2026); Meshchaninov et al. (2026).

Self-conditioning and token identifiability. We hypothesise that self-conditioning *also* enhances token identifiability. To show that, instead of removing self-conditioning outright, we attempt to replace it with a cheaper proxy at train time, which is rendered possible with small d. While the token identifiability issue is less severe in small dimension, the mismatch in norms between the Gaussian and the embedding of norm *one* is still present. In contrast, the self-conditioning signal,

$$\hat{x}_1 = \mathbf{e}_\theta(x_t, \mathbf{0}, t) \equiv \mathbb{E}_{X_1 \sim \pi_t^\theta(\cdot, \mathbf{0}, t)} \left[\mathsf{emb}(X_1) \mid I_t = x_t\right], \tag{9}$$

lies exactly on the convex hull formed by the embeddings. Instead of self-conditioning, we feed the model the Bayesian posterior expectation (*i.e.*, (8)), $\hat{x}_1 = D_t$, at train time. At inference, we try both the posterior and true self-conditioning. Since D_t is deterministic given $I_t = x_t$, it adds no information to the input, and gains must come from a better representation of the state. This approach attains 18.35% (vs. 11.98% without self-conditioning), and 20.24% with true self-conditioning at inference—at no additional forward pass cost at train-time.

Takeaway. Self-conditioning aids token identifiability as it holds a more identifiable representation of the state, on the convex hull formed by the embeddings.

Cross-method Pareto frontiers are available in Figure 7, and for 1024 NFE in Figure 3b.

5 Scaling SemiCat

We train a larger SemiCat model following the protocol of Davis et al. (2026); Sahoo et al. (2026) closely: the architecture is the same DiT (Peebles & Xie, 2023; Lou et al., 2024) with 1.7B parameters, trained on the Nemotron Pretraining dataset (Basant et al., 2025) for 2.1T tokens using the TikToken tokeniser (OpenAI, 2023), with sequences of length 2048. We use $256\times$ H100 GPUs for the pre-training run with a batch size of 10 per GPU. We also train a one-hot flow baseline on the same settings (Davis et al., 2026). See Appendix D.1 for more details.

Likelihood-based benchmarks. We evaluate our model over the same multiple-choice question answering (MCQA) benchmarks as in Davis et al. (2026); Sahoo et al. (2026): ARC-easy (Clark et al., 2018), BoolQ (Clark et al., 2019), OBQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2019), RACE (Lai et al., 2017) and SIQA (Sap et al., 2019). Instead of using an ELBO, we find the plain cross-entropy loss to be a better proxy for the likelihood for both the one-hot flow and SemiCat.

The results are available in Table 3. SemiCat outperforms our one-hot flow baseline on every MCQA task, and *outperforms the discrete diffusion baselines on ARC-E by* 8.4 *points, on BoolQ*

Method	ARC-E	BoolQ	OBQA	PIQA	RACE	SIQA
Chance	24.7	50.4	26.6	51.6	24.2	32.2
LLaDA-8B-Base (Nie et al., 2025)	–	–	–	74.4	–	–
AR† *(autoregressive)*	72.7	71.9	40.4	78.1	36.2	41.9
AR⋆ *(autoregressive)*	73.1	63.6	40.2	76.9	36.3	43.5
Duo† *(uniform diffusion)*	53.4	59.6	**33.0**	62.7	35.0	39.0
MDLM† *(masked diffusion)*	50.5	62.8	32.0	62.2	34.7	39.2
One-hot flow	42.6	37.9	26.8	56.4	28.2	36.4
SemiCat *(ours)*	**61.8**	**64.8**	30.2	**70.3**	**35.3**	**40.4**

Table 3: Zero-shot accuracy (%) on likelihood-based benchmarks. ⋆Models from Davis et al. (2026). †Models from Sahoo et al. (2026). Both with 1.7B parameters, trained on Nemotron for 2.1T tokens. SemiCat and one-hot flow likelihoods are estimated by the cross-entropy loss of the continuation (see eqs. (2) and (12)). Best non-AR results at 1.7B parameters per column are in **bold**.

by 2.0*, PIQA by* 7.6*, and on SIQA by* 1.2. On OBQA, our results are behind by 2.8 points, and we slightly exceed Duo and MDLM on RACE (+0.3).

Unconditional generation. Following the setting of Davis et al. (2026), SemiCat outperforms one-hot flows on unconditional generation, as measured by the Gen. PPL/Entropy Pareto frontiers provided in Figure 4. We generate sequences of 2048 tokens, and sweep over NFEs and temperature. The Gen. PPL is computed using Qwen 2.5 7B (Yang et al., 2025). At all NFEs, SemiCat dominates the one-hot flow baseline.

6 Related Work

Continuous diffusion for language modelling. Gaussian interpolation on one-hot encodings was introduced from a variational perspective in VFM (Eijkelboom et al., 2025), and has been shown to be amenable to self-distillation and steering by CFM/FMLM/DFM (Roos et al., 2026; Lee et al., 2026; Potaptchik et al., 2026b). Yoo et al. (2026) analyse self-conditioning through a fixed-point iteration lens; Shabalin et al. (2026) do so through that of multi-modality.

Embedded space design. Since those advances, $\mathbb{S}$-FLM and von Mises-Fisher flows (Deschenaux & Gulcehre, 2026; Chemseddine et al., 2026) were the first to show reasonable performance on TinyGSM for continuous methods, by flowing on the sphere but by taking different probability path constructions. In parallel, geometrically simpler approaches have been developed. LangFlow (Chen et al., 2026) use a simple linear projection, and ELF (Hu et al., 2026) attempt learnt embeddings but have found using a pre-trained encoder to be much more effective. Meshchaninov et al. (2026) show how to leverage a pretrained AR model to obtain strong embeddings.

We provide additional references in Appendix A.

7 Conclusion

In this work, we show that self-conditioning aids in two important aspects of CDLMs: trajectory ambiguity (increasing the density of inference-like points at train time) and token identifiability (the ability to identify the expected endpoint of the trajectory given its current state). For the former, DoubleCat can recover a large part of those benefits; for the latter, providing a better representation of the state, such as feeding the analytically available Bayesian posterior, helps substantially at a lower cost. We show that one-hot methods in fact evolve over an embedded space with anisotropic noise, while embedded methods trained naively recover one-hot properties in terms of their DER. SemiCat, a hybrid method, can attain discrete-diffusion-level performance on our benchmarks. At scale, SemiCat outperforms discrete diffusion on four out of six of our likelihood tasks, and surpasses one-hot flows on unconditional generation. We leave to future work accelerating these methods with the continuous toolbox, and understanding how that toolbox interacts with our design choices.

AI USE STATEMENT

In this work, we used generative AI tools for assisting in the writing of proofs, proposing or refining hypotheses, providing critical ingredients for proving mathematical claims, designing or providing feedback on research methodology or experiments, implementing methods, supporting qualitative and thematic data analysis, and interpreting results. We have not used generative AI tools for generating synthetic data sets, formulating mathematical claims, helping develop theoretical models or conceptual frameworks, assisting with translation, and cleaning and reformatting datasets. Additionally, we used generative AI tools for creating or modifying scientific figures or images, suggesting experimental parameters, creating or editing software code, brainstorming, editing the paper to improve readability, and drafting parts of the paper. We have reviewed all AI-assisted work. Any assistance with proofs, including their key ingredients, has been thoroughly re-written manually by one of the authors, and expounded to ascertain the exactitude of the claims. Proposed hypotheses have been greatly refined by the authors. Feedback provided on research methodology was scarcely requested, and was verified by two of the authors; the experiments closely follow existing work. LLM-generated code was contained in small specific evaluation and training parts, and was thoroughly reviewed by two of the authors; a majority of the code was manually written. Code generated for data analysis was also verified by each author. Drafted parts are mostly appendix material that was duly verified; LLM suggestions were taken into consideration for the writing of the main body. AI-suggested interpretations of results were checked against the data by the authors and subsequently refined. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work studies continuous diffusion language models and does not involve human subjects, personal data, or the release of new datasets. We rely on publicly available resources for all our experiments: TinyGSM (Liu et al., 2023), GSM8K (Cobbe et al., 2021), OpenWebText (Gokaslan & Cohen, 2019), Nemotron (Basant et al., 2025), ARC-easy (Clark et al., 2018), BoolQ (Clark et al., 2019), OBQA (Mihaylov et al., 2018), PIQA (Bisk et al., 2019), RACE (Lai et al., 2017), SIQA (Sap et al., 2019), GPT-2 (Radford et al., 2019), Qwen 2.5 (Yang et al., 2025). We have adhered to the ICLR Code of Ethics.

REPRODUCIBILITY STATEMENT

Proposition 1 states the assumptions made on the embeddings for that result, and the informal derivations also clearly state their assumptions (*e.g.*, Bayes-optimality, unit-norm embeddings). We follow closely existing and standard protocols for all our experiments, and we have made all our hyperparameters (including the architecture and optimisers) readily available in Appendix D.1.

REFERENCES

Manan Agarwal, Sheel Shah, Chanhyuk Lee, Jaehoon Yoo, Jerry Huang, Seunghoon Hong, Aditi Raghunathan, Jinwoo Kim, and Nicholas M. Boffi. Posterior refinement: Fast language generation via any-order flow maps, 2026. URL https://arxiv.org/abs/2606.24773.

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions, 2023. URL https://arxiv.org/abs/2303.08797.

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.

Parikshit Bansal and Sujay Sanghavi. Token time continuous diffusion for language modeling, 2026. URL https://arxiv.org/abs/2607.14106.

Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, et al. Nvidia

nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model, 2025. URL https://arxiv.org/abs/2508.14444.

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

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language, 2019. URL https://arxiv.org/abs/1911.11641.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models, 2025a. URL https://arxiv.org/abs/2406.07507.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation, 2025b. URL https://arxiv.org/abs/2505.18825.

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.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design, 2024. URL https://arxiv.org/abs/2402.04997.

Jannis Chemseddine, Gregor Kornhardt, and Gabriele Steidl. Spherical flows for sampling categorical data, 2026. URL https://arxiv.org/abs/2605.05629.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning, 2023. 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, 2026. URL https://arxiv.org/abs/2604.11748.

Chaoran Cheng, Jiahan Li, Jian Peng, and Ge Liu. Categorical flow matching on statistical manifolds, 2025. URL https://arxiv.org/abs/2405.16441.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. Boolq: Exploring the surprising difficulty of natural yes/no questions, 2019. URL https://arxiv.org/abs/1905.10044.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the ai2 reasoning challenge, 2018. URL https://arxiv.org/abs/1803.05457.

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.

Oscar Davis, Samuel Kessler, Mircea Petrache, İsmail İlkan Ceylan, Michael Bronstein, and Avishek Joey Bose. Fisher flow matching for generative modeling over discrete data, 2024. URL https://arxiv.org/abs/2405.14664.

Oscar Davis, Anastasiia Filippova, Pierre Ablin, Victor Turrisi, Amitis Shidani, Marco Cuturi, and Louis Béthune. Scaling categorical flow maps, 2026. URL https://arxiv.org/abs/2605.07820.

Justin Deschenaux and Caglar Gulcehre. Language modeling with hyperspherical flows, 2026. URL https://arxiv.org/abs/2605.11125.

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.

Floor Eijkelboom, Grigory Bartosh, Christian Andersson Naesseth, Max Welling, and Jan-Willem van de Meent. Variational flow matching for graph generation, 2025. URL https://arxiv.org/abs/2406.04843.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching, 2024. URL https://arxiv.org/abs/2407.15595.

Zhengyang Geng, Ashwini Pokle, William Luo, Justin Lin, and J. Zico Kolter. Consistency models made easy, 2024. URL https://arxiv.org/abs/2406.14548.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J. Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling, 2025. URL https://arxiv.org/abs/2505.13447.

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

Samson Gourevitch, Yazid Janati, Dario Shariatian, Umut Simsekli, Eric Moulines, Eric P. Xing, and Alain Durmus. Uniform diffusion models revisited: Leave-one-out denoiser and absorbing state reformulation, 2026. URL https://arxiv.org/abs/2605.22765.

Alex Graves, Rupesh Kumar Srivastava, Timothy Atkinson, and Faustino Gomez. Bayesian flow networks, 2025. URL https://arxiv.org/abs/2308.07037.

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

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models, 2020. URL https://arxiv.org/abs/2006.11239.

Peter Holderrieth, Douglas Chen, Luca Eyring, Ishin Shah, Giri Anantharaman, Yutong He, Zeynep Akata, Tommi Jaakkola, Nicholas Matthew Boffi, and Max Simchowitz. Diamond maps: Efficient reward alignment via stochastic flow maps, 2026. URL https://arxiv.org/abs/2602.05993.

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.

Emiel Hoogeboom, David Ruhe, Jonathan Heek, Thomas Mensink, and Tim Salimans. Beyond single tokens: Distilling discrete diffusion models via discrete mmd, 2026. URL https://arxiv.org/abs/2603.20155.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. Elf: Embedded language flows, 2026. URL https://arxiv.org/abs/2605.10938.

Pascal Jutras-Dubé, Patrick Pynadath, Jeremy Lu, Yuan Gao, and Ruqi Zhang. Sticky jump diffusions: A unifying view of masked, continuous, and hybrid diffusion, 2026. URL https://arxiv.org/abs/2607.10951.

Jaeyeon Kim, Jonathan Geuter, David Alvarez-Melis, Sham Kakade, and Sitan Chen. Stop training for the worst: Progressive unmasking accelerates masked diffusion training, 2026. URL https://arxiv.org/abs/2602.10314.

Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models, 2021. URL https://arxiv.org/abs/2107.00630.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. Race: Large-scale reading comprehension dataset from examinations, 2017. URL https://arxiv.org/abs/1704.04683.

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

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, 2022. URL https://arxiv.org/abs/2209.03003.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization, 2019. URL https://arxiv.org/abs/1711.05101.

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.

Viacheslav Meshchaninov, Alexander Shabalin, Egor Chimbulatov, Nikita Gushchin, Ilya Koziev, Alexander Korotin, and Dmitry Vetrov. How to train your latent diffusion language model jointly with the latent space, 2026. URL https://arxiv.org/abs/2605.07933.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering, 2018. URL https://arxiv.org/abs/1809.02789.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models, 2025. URL https://arxiv.org/abs/2502.09992.

OpenAI. tiktoken: A fast BPE tokenizer for use with OpenAI models, 2023. URL https://github.com/openai/tiktoken.

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

Peter Potaptchik, Cheuk-Kit Lee, and Michael S. Albergo. Tilt matching for scalable sampling and fine-tuning, 2025. URL https://arxiv.org/abs/2512.21829.

Peter Potaptchik, Adhi Saravanan, Abbas Mammadov, Alvaro Prat, Michael S. Albergo, and Yee Whye Teh. Meta flow maps enable scalable reward alignment, 2026a. URL https://arxiv.org/abs/2601.14430.

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

Mihir Prabhudesai and Zhengyang Geng. Latent thought flows with text compression. Technical report, Carnegie Mellon University, 2026. URL https://latent-thought-flows.vercel.app/.

Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. Candi: Hybrid discrete-continuous diffusion models, 2026. 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.

Gabriel Raya, Bac Nguyen, Georgios Batzolis, Yuhta Takida, Dejan Stancevic, Naoki Murata, Chieh-Hsin Lai, Yuki Mitsufuji, and Luca Ambrogioni. Noise scheduling as information-guided allocation in diffusion training, 2026. URL https://arxiv.org/abs/2602.18647.

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.

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.

Subham Sekhar Sahoo, Jean-Marie Lemercier, Zhihan Yang, Justin Deschenaux, Jingyu Liu, John Thickstun, and Ante Jukic. Scaling beyond masked diffusion language models, 2026. URL https://arxiv.org/abs/2602.15014.

Tim Salimans, Thomas Mensink, Jonathan Heek, and Emiel Hoogeboom. Multistep distillation of diffusion models via moment matching, 2024. URL https://arxiv.org/abs/2406.04103.

Maarten Sap, Hannah Rashkin, Derek Chen, Ronan LeBras, and Yejin Choi. Socialiqa: Commonsense reasoning about social interactions, 2019. URL https://arxiv.org/abs/1904.09728.

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, 2024. URL https://arxiv.org/abs/2412.10193.

Alexander Shabalin, Simon Elistratov, Viacheslav Meshchaninov, Ildus Sadrtdinov, and Dmitry Vetrov. Why gaussian diffusion models fail on discrete data and how to prevent it?, 2026. URL https://arxiv.org/abs/2604.02028.

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.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics, 2015. URL https://arxiv.org/abs/1503.03585.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models, 2023. URL https://arxiv.org/abs/2303.01469.

Tong Wu, Zhihao Fan, Xiao Liu, Yeyun Gong, Yelong Shen, Jian Jiao, Hai-Tao Zheng, Juntao Li, Zhongyu Wei, Jian Guo, Nan Duan, and Weizhu Chen. Ar-diffusion: Auto-regressive diffusion model for text generation, 2023. URL https://arxiv.org/abs/2305.09515.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, et al. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

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.

Huangjie Zheng, Shansan Gong, Ruixiang Zhang, Tianrong Chen, Jiatao Gu, Mingyuan Zhou, Navdeep Jaitly, and Yizhe Zhang. Continuously augmented discrete diffusion model for categorical generative modeling, 2025. URL https://arxiv.org/abs/2510.01329.

Cai Zhou, Chenxiao Yang, Yi Hu, Chenyu Wang, Chubin Zhang, Muhan Zhang, Lester Mackey, Tommi Jaakkola, Stephen Bates, and Dinghuai Zhang. Coevolutionary continuous discrete diffusion: Make your diffusion language model a latent reasoner, 2026. URL https://arxiv.org/abs/2510.03206.

A Extended related work

Earlier approaches. Earlier approaches of continuous diffusion for language modelling include CDCD (Dieleman et al., 2022), Diffusion-LM (Li et al., 2022), Bayesian Flow Networks (Graves et al., 2025).

Geometric constructions. Earlier geometric work includes Davis et al. (2024); Cheng et al. (2025), which flowed both on Fisher-Rao simplex and on the positive orthant on the sphere, but that failed to scale to larger vocabulary sizes.

Beyond token-wise embeddings. Latent Thought Flow (Prabhudesai & Geng, 2026) explores the idea of compressing the sequence length. We believe that this general direction is also very promising, but the exploration thereof should be deferred to another dedicated work. Other works consider using an existing strong AR model for encoding, such as Meshchaninov et al. (2026), which uses the contextual embeddings of GPT-2 (Radford et al., 2019). While offering better performance, the approach is naturally more expensive computationally at train time.

Hybrid methods. Besides CANDI (Pynadath et al., 2026) and FMLM+ (Agarwal et al., 2026), we note also the existence of the literature of *discrete* diffusion with continuous augmentations, for instance via latents (Zheng et al., 2025; Zhou et al., 2026). Concurrently, Jutras-Dubé et al. (2026) formalised these hybrids, together with masked and continuous diffusion, as special cases of a single process on the embedding space. Tokens form discrete anchors that the reverse process can jump onto, and the forward process determines when and where those jumps happen. In their framework, the CANDI-style hybrid underlying $\mathrm{SemiCat}$ is recovered as the identity-kernel case. We see this as a principled way to design hybrid corruption processes, complementary to the variants we explore in Appendix D.5.

B Extended SemiCat Presentation

For completeness, we detail here $\mathrm{SemiCat}$ formally. First note that the $\mathrm{DoubleCat}$ interpolants can be written as

$$I_t^{\mathsf{dc},\ell}(X_0, X_1) = (1-\gamma_t)X_0^\ell + \gamma_t(1-\kappa_t)Y_0^\ell + \gamma_t\kappa_t X_1^\ell. \tag{10}$$

Hence, the $\mathrm{SemiCat}$ interpolants:

$$I_t^{\mathsf{sc},\ell}(X_0, X_1) = I_t^{\mathsf{dc},\ell}(X_0, X_1)\mathbf{1}(t < \tau^\ell) + \mathbf{1}(t \geq \tau^\ell)X_1^\ell, \tag{11}$$

where $\mathbf{1}$ is the indicator function. An optimal model carries over tokens for which $t \geq \tau^\ell$, and so we parametrise it as such (Sahoo et al., 2024). The training therefore reduces to the following cross-entropy loss:

$$\mathcal{L}(\theta) = -\mathbb{E}_{X_0,X_1,t,\boldsymbol{\tau}} \sum_{\ell=1}^{L} \mathbf{1}(t < \tau^\ell) \log \left\langle X_1^\ell, \pi_t^{\theta,\ell}(I_t^{\mathsf{sc}}) \right\rangle, \tag{12}$$

where $\boldsymbol{\tau} = (\tau^1, \ldots, \tau^L)$. Following $\mathrm{DoubleCat}$, we ignore the $(Y_t)_{t\in[0,1]}$ terms at inference time, thus amounting to CANDI-like sampling: each still-continuous position is revealed between times t_n and t_{n+1} with probability $\frac{t_{n+1}-t_n}{1-t_n}$, committing a token drawn from π^θ, while the remaining positions follow the one-hot flow, as summarised in Algorithm 1.

C Additional background

C.1 Discrete diffusion

Discrete diffusion models corrupt a sequence by acting directly on V^L, and learn to invert that corruption. We keep the flow matching time convention of the main text: $t = 1$ carries the data and $t = 0$ the noise, so the signal level $\gamma : [0,1] \longrightarrow [0,1]$ is increasing with $\gamma_0 = 0$ and $\gamma_1 = 1$, exactly as for the flow schedules of Section 2. Many works in the discrete diffusion literature use the opposite diffusion convention.

Algorithm 1 SemiCat sampling.

Require: Denoiser π^θ, schedule γ, time grid $0 = t_0 < \cdots < t_N = 1$, temperature T.
1: $x^\ell \sim p_0$ and $r^\ell \leftarrow 0$ for all ℓ ▷ $r^\ell = 1$: revealed
2: **for** $n = 0, \dots, N-1$ **do**
3: $p \leftarrow \pi^\theta_{t_n}(x)$
4: **for** ℓ such that $r^\ell = 0$ **do**
5: $x^\ell \leftarrow x^\ell + \frac{\gamma_{t_{n+1}} - \gamma_{t_n}}{1-\gamma_{t_n}} \left(p^\ell - x^\ell\right)$ ▷ Euler step towards π^θ
6: With probability $\frac{t_{n+1}-t_n}{1-t_n}$: draw $v \sim \mathrm{Cat}\big((p^\ell)^{1/T}\big)$, set $x^\ell \leftarrow e_v$ and $r^\ell \leftarrow 1$
7: **end for**
8: **end for**
9: **return** $\left(\mathrm{argmax}_v x^\ell_v\right)_{\ell=1}^L$

The forward process. The two families below are instances of D3PM (Austin et al., 2023), which builds on Sohl-Dickstein et al. (2015); Hoogeboom et al. (2021); Ho et al. (2020), and whose continuous-time limits are continuous-time Markov chains (Campbell et al., 2022). The corruption factorises over positions, and each position either keeps its token or is replaced by a draw from a fixed reference distribution q_{ref} on V:

$$q(X_t^\ell \mid \mathbf{x}^\ell) = \gamma_t\, \delta_{\mathbf{x}^\ell} + (1-\gamma_t)\, q_{\mathrm{ref}}. \tag{13}$$

Masked diffusion models (MDMs) take $q_{\mathrm{ref}} = \delta_{\mathsf{m}}$, for an absorbing mask token $\mathsf{m} \notin V$, so that $X_t \in (V \cup \{\mathsf{m}\})^L$ while uniform-state diffusion models (USDMs) take $q_{\mathrm{ref}} = \mathcal{U}(V)$, so that $X_t \in V^L$. The two choices give different objectives and different samplers.

Masked diffusion. The denoiser takes the corrupted sequence and returns a distribution over V at every position, *i.e.*, $\pi_t^{\theta,\ell}(X_t) \in \mathbb{R}^{|V|}$. Two constraints make it exact: it puts no mass on m, and it copies positions that are already unmasked, since the forward process never changes them (Sahoo et al., 2024; Shi et al., 2025). The reverse process only unmasks at rate

$$\hat{R}_t^\ell(\mathsf{m}, j) = \frac{\dot\gamma_t}{1-\gamma_t}\, \mathbf{1}[j = \mathbf{x}^\ell], \qquad \hat{R}_t^{\theta,\ell}(\mathsf{m}, j) = \frac{\dot\gamma_t}{1-\gamma_t}\, \pi_t^{\theta,\ell}(X_t)_j, \tag{14}$$

and since $\pi_t^{\theta,\ell}(X_t)$ sums to one, the total outflow $\sum_j \hat{R}_t^{\theta,\ell}(\mathsf{m}, j)$ does not depend on θ. Under these constraints the continuous-time bound collapses to a masked cross-entropy,

$$-\log p_\theta(\mathbf{x}) \le -\int_0^1 \frac{\dot\gamma_t}{1-\gamma_t}\, \mathbb{E}_{X_t \sim q(\cdot \mid \mathbf{x})} \left[\sum_{\ell\,:\, X_t^\ell = \mathsf{m}} \log \left\langle X_1^\ell, \pi_t^{\theta,\ell}(X_t) \right\rangle \right] \mathrm{d}t, \tag{15}$$

where $X_1^\ell = e_{\mathbf{x}^\ell}$ as in Section 2, and which, for the linear schedule $\gamma_t = t$, weighs the loss by $1/(1-t)$. Sampling reverses (13) by unmasking a subset of positions per step; a token, once revealed, is never revisited. MDMs are currently the most common setting of recent diffusion language models at scale (Nie et al., 2025).

Uniform-state diffusion. With $q_{\mathrm{ref}} = \mathcal{U}(V)$ the corrupted sequence contains no marker of which positions are wrong, so the model must both locate and correct the errors. Nothing is frozen, so the sampler may revise any token at any time rather than committing once. For completeness, we give the closed forms which are those of the uniform kernel of Austin et al. (2023); Hoogeboom et al. (2021).

Let us write the marginal at position ℓ, and the same quantity with the data replaced by the denoiser's prediction as

$$q_t^\ell(j) \coloneqq \gamma_t\, \mathbf{1}[j = \mathbf{x}^\ell] + \frac{1-\gamma_t}{|V|}, \qquad q_t^{\theta,\ell}(j) \coloneqq \gamma_t\, \pi_t^{\theta,\ell}(X_t)_j + \frac{1-\gamma_t}{|V|}. \tag{16}$$

Because the uniform kernels compose $K_a K_b = K_{ab}$, the transition from a cleaner time s to a noisier time $t < s$ retains with probability γ_t/γ_s, and Bayes' rule gives the exact reverse posterior as an

element wise product

$$q(X_s^\ell \mid X_t^\ell, \mathbf{x}^\ell) \propto \left(\frac{\gamma_t}{\gamma_s}\, e_{X_t^\ell} + \left(1 - \frac{\gamma_t}{\gamma_s}\right)\frac{\mathbf{1}}{|V|}\right) \odot \left(\gamma_s\, e_{\mathbf{x}^\ell} + (1-\gamma_s)\frac{\mathbf{1}}{|V|}\right), \tag{17}$$

with $p_\theta(X_s^\ell \mid X_t^\ell)$ obtained by substituting $\pi_t^{\theta,\ell}(X_t)$ for $e_{\mathbf{x}^\ell}$. This is the kernel a finite-step sampler uses. Its continuous-time limit is a jump process: writing $\nu_t \coloneqq \dot\gamma_t/\gamma_t$ for the corruption rate, the noising process leaves each position at rate ν_t towards $\mathcal{U}(V)$, and time reversal gives, for $j \neq i$

$$\hat R_t^\ell(i,j) = \frac{\nu_t}{|V|}\cdot\frac{q_t^\ell(j)}{q_t^\ell(i)}, \qquad \hat R_t^{\theta,\ell}(i,j) = \frac{\nu_t}{|V|}\cdot\frac{q_t^{\theta,\ell}(j)}{q_t^{\theta,\ell}(i)}. \tag{18}$$

The prior term of the ELBO vanishes, since $\gamma_0 = 0$ makes $q(X_0 \mid \mathbf{x})$ exactly the uniform prior, and the remainder is the integrated KL rate between the two jump processes (Campbell et al., 2022), which after dropping θ-independent terms would become

$$-\log p_\theta(\mathbf{x}) \le \frac{1}{|V|}\int_0^1 \nu_t\, \mathbb{E}_{X_t\sim q(\cdot\,|\,\mathbf{x})}\left[\sum_{\ell=1}^{L}\sum_{j\neq X_t^\ell}\left(\frac{q_t^{\theta,\ell}(j)}{q_t^{\theta,\ell}(X_t^\ell)} - \frac{q_t^\ell(j)}{q_t^\ell(X_t^\ell)}\log\frac{q_t^{\theta,\ell}(j)}{q_t^{\theta,\ell}(X_t^\ell)}\right)\right]\mathrm{d}t + \text{const.} \tag{19}$$

Unlike (14), the total outflow $\sum_{j\neq X_t^\ell}\hat R_t^{\theta,\ell}(X_t^\ell, j)$ *does* depend on θ, so the first term does not drop and the objective is not a cross-entropy: the model is scored on the ratios $q_t^{\theta,\ell}(j)/q_t^{\theta,\ell}(X_t^\ell)$ rather than on the probability it assigns to the true token, and every position contributes at every t, not only the corrupted ones. See Schiff et al. (2024); Gourevitch et al. (2026) for practical estimators of (19), and Lou et al. (2024), which sidesteps it by learning the ratios $p_{\mathrm{data}}(\mathbf{y})/p_{\mathrm{data}}(\mathbf{x})$ directly with a score-entropy loss.

Relation to the continuous case. USDMs are the discrete family closest to the flows of Section 2: Sahoo et al. (2025) show that a Gaussian diffusion pushed through an argmax matches the marginals of a uniform state diffusion. Discrete flow matching (Gat et al., 2024; Campbell et al., 2024) reformulates both families as probability paths with a prescribed interpolant, in a formalism much closer to that of Section 2.

C.2 The VDM noise schedule

Plaid (Gulrajani & Hashimoto, 2023; Yang et al., 2026) is a variational diffusion model (Kingma et al., 2021) on token embeddings, so the weight in (4) is the derivative of a learnt signal-to-noise ratio. We recall here where it comes from, and why its endpoints are treated differently from its shape.

Following Section 2, Plaid corrupts the embedded sequence as $I_t = \alpha_t X_0 + \beta_t \mathsf{emb}(X_1)$, with $X_0 \sim \mathcal{N}(\mathbf{0}, \mathrm{I})$ and emb applied position-wise, and sets

$$\mathrm{SNR}(t) \coloneqq \frac{\beta_t^2}{\alpha_t^2}. \tag{20}$$

VDM takes the path variance-preserving, $\alpha_t^2 + \beta_t^2 = 1$, so that $\beta_t^2 = \mathrm{sigmoid}(\log \mathrm{SNR}(t))$ and the whole path is determined by $\log \mathrm{SNR}$ alone which it parametrises with a monotone network in t. As shown in (4), the continuous time bound then splits into a prior term, a reconstruction term, and a diffusion term with the diffusion term being

$$\mathcal{L}_{\mathrm{diff}}(\theta) = \frac{1}{2}\,\mathbb{E}_{t\sim\mathcal{U}(0,1),\,X_0}\left[\mathrm{SNR}'(t)\;\|\mathbf{e}_\theta(I_t,t) - \mathsf{emb}(\mathbf{x})\|^2\right], \tag{21}$$

with $\mathrm{SNR}' > 0$. The schedule enters (21) twice: through the weight $\mathrm{SNR}'(t)$ and through I_t. The second dependence carries no extra freedom, since $\beta_t^2 = \mathrm{sigmoid}(\log \mathrm{SNR}(t))$ and $\alpha_t^2 = 1 - \beta_t^2$ are functions of $\mathrm{SNR}(t)$ alone. Let us write

$$J_w \coloneqq \sqrt{\frac{1}{1+w}}\, X_0 + \sqrt{\frac{w}{1+w}}\, \mathsf{emb}(\mathbf{x}) \tag{22}$$

for the corrupted sequence at SNR level w, we have $I_t = J_{\mathrm{SNR}(t)}$: two schedules that reach the same SNR give the same noisy input, whatever time they reach it at. The denoiser's second argument behaves the same way as VDM conditions it on $\log \mathrm{SNR}(t)$ rather than on t; we write $\mathbf{e}_\theta(\cdot, w)$ for it below. Substituting $w = \mathrm{SNR}(t)$ so that $\mathrm{d}w = \mathrm{SNR}'(t)\,\mathrm{d}t$,

$$\mathcal{L}_{\mathrm{diff}}(\theta) = \frac{1}{2}\int_{\mathrm{SNR}(0)}^{\mathrm{SNR}(1)} \mathbb{E}_{X_0} \left\|\mathbf{e}_\theta(J_w, w) - \mathsf{emb}(\mathbf{x})\right\|^2 \mathrm{d}w, \tag{23}$$

so, at a fixed denoiser, the bound depends on the schedule through its two endpoints $\mathrm{SNR}(0)$ and $\mathrm{SNR}(1)$ and nothing else. VDM uses both halves of this statement: the endpoints are learnt by the ELBO, and the shape in between is learnt by minimising the variance of the Monte Carlo estimate of (21). Plaid inherits both, which is what we mean by learnt log-SNR endpoints in Table 2.

D Additional experimental results

D.1 Hyperparameters and experimental setup

Hyperparameters. We provide all our hyperparameters in Table 4. Throughout, we use the standard diffusion transformer (DiT Peebles & Xie, 2023) adapted by Lou et al. (2024) for discrete diffusion. For one-hot flows, we also use a simple linear projection in $\mathbb{R}^{|V|\times h}$, where h is the hidden dimension of the transformer, following Roos et al. (2026); Lee et al. (2026). Models are evaluated with the EMA weights, except for in the large Nemotron case. We evaluate using the final checkpoint. OWT and TinyGSM checkpoints are trained using $8\times$ H100. Nemotron 1.7B checkpoints are trained with $256\times$ H100. All models are trained with AdamW (Loshchilov & Hutter, 2019); the hyperparameters for which are provided in Table 4 too.

L2 Weighting. For one-hot flows (including base one-hot, CANDI, DoubleCat, SemiCat), we weigh the loss as in Davis et al. (2026), following Potaptchik et al. (2026b); Geng et al. (2024), inversely proportional to the square root of the L_2 distance of the prediction to the target. Precisely, we divide the loss by $\mathrm{sg}(\Delta)$ for $\Delta = \sqrt{\left\|\pi_t^{\theta,\ell}(I_t) - X_1^\ell\right\| + \epsilon}$, with $\epsilon = 10^{-2}$.

OWT setup. We use the typical OWT experimental setup: 1024 long sequences, tokenised by the tokeniser of GPT-2 (Radford et al., 2019).

TinyGSM. The TinyGSM dataset (Liu et al., 2023) contains 12.3M grade school mathematics questions paired with Python code traces that generate the correct solution.

D.2 DoubleCat ablations

The ablations over the choice of prior and the choice of schedule for DoubleCat are available in Tables 5 and 6. We also provide the full ablation of DoubleCat and one-hot flow with self-conditioning on and off in Table 7.

D.3 OpenWebText Pareto Frontiers

We provide additional OWT Pareto frontiers in Figures 5 to 7.

D.4 Corruption as data augmentation

We show, here, that corruption seen as data augmentation, instead of a proper change in the interpolant, lowers the performance of the resulting model on the unconditional generation task of OWT by substantially diminishing its diversity (entropy). To be precise, for $U^\ell \sim \mathcal{U}(V)$ i.i.d., we use interpolants instead pointing towards

$$\tilde{X}_1^\ell = \zeta_t X_1^\ell + (1-\zeta_t)U^\ell, \tag{24}$$

with various choices of $\zeta_t \sim \mathrm{Ber}(\rho_t)$, constant and scheduled. Results are provided in Table 8. Any score comparable in Gen. PPL and entropy is found for very low constant choices of ρ_t; lower (better) Gen. PPL scores imply entropy collapse.

	OpenWebText	TinyGSM	Nemotron (1.7B)
Architecture (DiT)			
Hidden size	768	768	2304
adaLN cond. dim	128	128	128
Layers	12	12	18
Attention heads	12	12	18
FFN (SwiGLU) multiplier	4	4	3
Dropout	0	0	0
Sequence length L	1024	512	2048
Vocabulary $\|V\|$	50 257 (GPT-2)	151 680 (Qwen2)	100 352 (TikToken)
Parameters, trunk	121M	121M	1 281M
Parameters, in + out proj.	77M	233M	462M
Parameters, total	199M	354M	1 743M
Optimisation (AdamW)			
Optimiser	(β_1=0.9, β_2=0.999, ϵ=10^{-8})		(β_1=0.9, β_2=0.95, ϵ=10^{-8})
Weight decay	0.1		0.1
Gradient clipping	1.0		1.0
Schedule	Cosine decay w/ linear warmup		Cosine decay w/ linear warmup
Warmup steps	3 000		2 000
Peak / final LR	3×10^{-4} / 10^{-5}		3×10^{-4} / 3×10^{-5}
Total steps	1 000 000	200 000	400 457
Global batch size	512	256	2 560
Precision	bfloat16 mixed		bfloat16 mixed
EMA decay	0.9999		none

Table 4: Hyperparameters of our models on OpenWebText, TinyGSM and Nemotron.

	Accuracy (%)				Best (%)	
Y_0^ℓ	$T=1$	$T=0.5$	$T=0.2$	top-k=1	Runs (↑)	Name err. (↓)
Barycentre	4.70	6.52	6.29	7.51	**55.57**	**43.90**
Uniform	**6.14**	**9.17**	**10.84**	**11.30**	53.53	46.10
Uniform vertex	5.31	7.05	7.81	8.49	51.48	47.99

Table 5: Ablation on the simplex prior Y_0^ℓ for DoubleCat, at NFE 256. Accuracy is reported per sampler; the last two columns report the best value attained over those samplers. "Barycentre" is $\delta_{\frac{\mathbf{1}}{|V|}}$, "Uniform" is uniform over the simplex, $\mathcal{U}(\Delta^{|V|})$, and "uniform vertex" is uniform over the vertices of the simplex (*i.e.*, the one-hot encoding of $\mathcal{U}(V)$). The inner schedule is set to $\kappa_t \equiv t^2$. We retain the uniform distribution which consistently outperforms the other choices on accuracy.

	Accuracy (%)				Best (%)	
κ_t	$T=1$	$T=0.5$	$T=0.2$	top-k=1	Runs (↑)	Name err. (↓)
$\sqrt{t}$	**9.70**	**13.50**	**14.18**	**14.78**	**63.61**	**36.16**
t	6.97	9.55	10.61	10.84	55.19	44.43
t^2	6.14	9.17	10.84	11.30	53.53	46.10
t^3	5.00	7.35	7.05	8.79	51.18	48.45

Table 6: Ablation on the simplex schedule κ_t for DoubleCat, at NFE 256, with $Y_0^\ell \sim \mathcal{U}(\Delta^{|V|})$ for all runs. Accuracy is reported per sampler; the last two columns report the best value attained over those samplers.

Method	Self-cond.	**Accuracy** (%, ↑) $T=1$	$T=0.5$	$T=0.2$	top-k=1	**Best** (%) Runs (↑)	Name err. (↓)
One-hot flow	Off	3.28	4.42	3.89	3.64	50.91	48.07
	On	**6.44**	**6.75**	**8.34**	**6.97**	**61.79**	**37.60**
DoubleCat	Off	11.45	15.85	16.00	15.92	69.67	30.02
	On	**16.83**	**20.77**	**20.70**	**19.79**	**75.21**	**24.34**

Table 7: Self-conditioning effect on one-hot flows and DoubleCat for schedule $\kappa_t \equiv \sqrt{t}$.

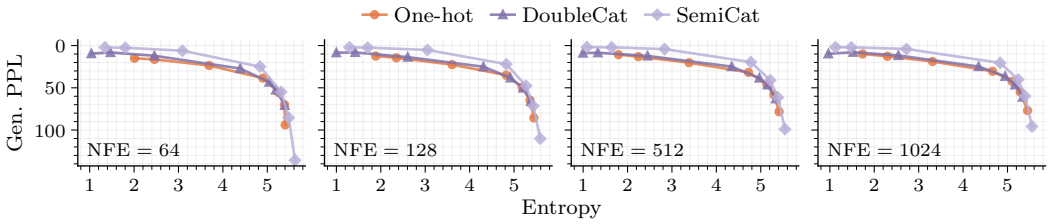


Figure 5: Gen. PPL/entropy Pareto frontiers as measured by GPT-2 for one-hot flows trained on OpenWebText with sequence lengths of 1024 tokens, using the GPT-2 tokeniser.

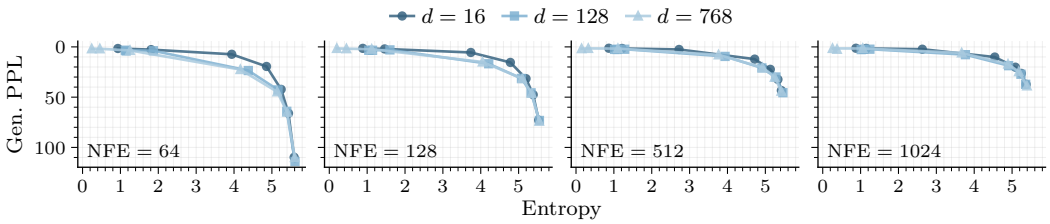


Figure 6: Gen. PPL as measured by GPT-2 for models trained on OpenWebText with sequence lengths of 1024 tokens, using the GPT-2 tokeniser. All Plaid runs for different embedding dimensionalities, d.

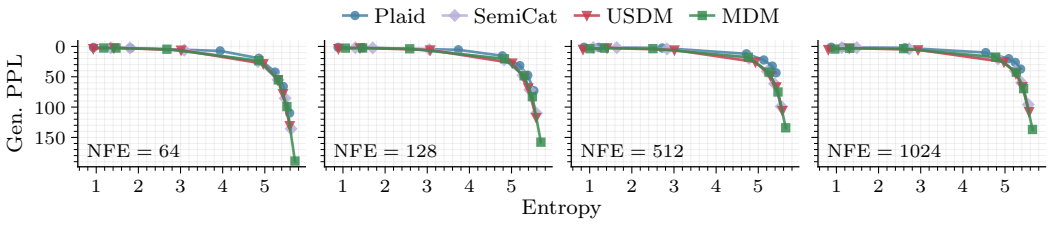


Figure 7: Gen. PPL as measured by GPT-2 for models trained on OpenWebText with sequence lengths of 1024 tokens, using the GPT-2 tokeniser. Plaid is for $d = 16$.

Flip probability ρ_t	Gen. PPL (↓) (entropy) NFE $= 50$	NFE $= 100$	NFE $= 200$
Baseline, 1M steps, NFE $= 128$			
One-hot flow		85.64 (5.45)	
DoubleCat		66.02 (5.38)	
SemiCat		117.00 (5.62)	
no L_2 weighting			
$\rho_t \equiv 0.01$	127.70 (5.44)	118.74 (5.52)	115.59 (5.51)
$\rho_t \equiv 0.1$	91.58 (5.09)	81.71 (5.19)	79.62 (5.12)
$\rho_t \equiv 0.25$	68.97 (4.72)	61.64 (4.83)	65.67 (4.88)
$\rho_t \equiv 0.5$	48.94 (4.13)	44.23 (4.16)	47.68 (4.18)
$\rho_t = t$	22.88 (3.12)	22.07 (3.30)	23.13 (3.18)
$\rho_t = \gamma_t$	26.80 (2.86)	22.41 (2.86)	24.29 (2.99)
with L_2 weighting			
$\rho_t \equiv 0.01$	100.44 (5.42)	86.72 (5.43)	85.33 (5.45)
$\rho_t \equiv 0.1$	75.00 (5.14)	65.62 (5.19)	63.50 (5.18)
$\rho_t \equiv 0.25$	57.64 (4.84)	53.68 (4.89)	51.50 (4.89)
$\rho_t \equiv 0.5$	44.43 (4.28)	39.28 (4.30)	41.99 (4.31)
$\rho_t = t$	27.99 (3.37)	28.55 (3.46)	29.26 (3.36)
$\rho_t = \gamma_t$	24.60 (3.00)	21.63 (2.98)	23.82 (3.13)

Table 8: OWT Gen. PPL (GPT-2), token entropy in parentheses, at 200k steps, temperature 1. The corruption flips each X_1^ℓ independently, with probability ρ_t, to a token drawn uniformly from V; it enters the interpolant only, leaving the loss target clean. Trained with γ being the usual mixture schedule.

α	**Accuracy** (%, ↑) $T = 1$	$T = 0.5$	$T = 0.2$	top-k=1	**Best** (%) Runs (↑)	Name err. (↓)
0	7.20	9.55	9.63	8.72	69.22	29.26
0.25	13.50	18.35	17.89	17.97	79.38	19.86
0.5	**19.26**	23.88	25.17	25.93	84.91	14.71
0.75	15.47	**25.25**	**29.57**	**29.26**	**87.11**	**12.81**
1	14.56	23.86	27.74	27.37	86.73	**12.81**

Table 9: TinyGSM accuracy at NFE $= 256$, for α-Cat at corruption levels $\alpha \in \{0, 0.25, 0.5, 0.75, 1\}$.

D.5 General hybrid corruption processes

We also explore the design of hybrid methods based on continuous flows. Inspired by AUDM (Gourevitch et al., 2026), we introduce α-Cat, a generalisation of CANDI (Pynadath et al., 2026), where the model is instead exposed to the following interpolants: letting $\tau^\ell \sim \mathcal{U}(0,1)$ i.i.d., as before, and $U^\ell \mid X_1^\ell = \mathbf{x}^\ell \sim \mathrm{Cat}(\alpha \frac{\mathbf{1}}{|V|} + (1-\alpha) e_{\mathbf{x}^\ell})$ i.i.d., we define the interpolants to be, for each ℓ, for $\tau^\ell > t$,

$$I_t^\ell(X_0, X_1) = (1-\gamma_t) X_0^\ell + \gamma_t e_{U^\ell}, \tag{25}$$

and, for $\tau^\ell \leq t$,

$$I_t^\ell(X_0, X_1) = X_1^\ell. \tag{26}$$

The model is thus effectively exposed to various levels α of additional discrete corruption, recovering CANDI at $\alpha = 0$. Simply put, at each position, the model flows towards either a random vertex with probability α or to the true one otherwise, until its reveal time τ^ℓ. We report the TinyGSM results in Table 9. As we can see, with increasing α, we recover better performance until $\alpha = 0.75$ as it recovers the effect of SemiCat. Nonetheless, for $\alpha = 1$, the model learns a leave-one-out denoiser; that is to say, it learns $\mathbb{E}[X_1^\ell \mid I_t^{-\ell} = x_t^{-\ell}]$, which ignores position ℓ to predict token ℓ, like discrete diffusion methods (Gourevitch et al., 2026). This is natural, since $\mathrm{I}(I_t^\ell; X_1^\ell) = 0$ for $t < \tau^\ell$ and $\alpha = 1$, where I denotes mutual information. Variants with $\alpha > 0$ thus partially defy the purpose of hybrid methods, as they rely on discretely revealed positions.

	Accuracy (%, ↑)					
Student	NFE 1	2	4	8	16	32
FMLM+ (Agarwal et al., 2026)	0.2	1.2	5.7	12.4	17.8	19.9
SemiCat teacher	–	–	–	5.5	12.1	20.1
$T = 1$	0.2	0.5	1.6	5.4	13.0	19.0
$T = 0.5$	0.7	1.2	2.8	9.8	16.5	24.0
$T = 0.3$	0.5	0.8	3.9	9.2	16.3	22.7
$T = 0.2$	0.6	1.8	4.8	10.9	16.3	22.5
$T = 0.05$	**1.1**	1.8	3.6	9.0	17.1	**24.2**
+ boundary pairs 3%	**1.1**	**2.5**	**5.6**	9.6	15.9	20.0
+ boundary pairs 25%	0.6	1.1	1.9	4.9	10.0	15.5
+ gap curriculum 2k	**1.1**	2.4	5.4	**11.7**	**19.1**	22.1
+ gap curriculum 2k, 3%	0.6	2.0	3.9	6.7	13.8	20.0
+ gap curriculum 5k, 3%	0.7	1.5	3.7	7.8	12.4	20.4

Table 10: TinyGSM accuracy for D-MMD distillation of SemiCat, at teacher temperature T. Students train for 20k steps at constant learning rate 2×10^{-5}, with early stopping on GSM8K test set accuracy at 32 NFE.

D.6 Acceleration via D-MMD

We apply D-MMD distillation (Hoogeboom et al., 2026; Salimans et al., 2024) to our pre-trained SemiCat model. We tune the temperature of the teacher T, and attempt some simple techniques such as sampling boundary pairs $(s, t) = (0, 1)$ with a probability p, and a gap curriculum updated every n steps, both inspired by Lee et al. (2026). We find that the resulting model is competitive with FMLM+ (Agarwal et al., 2026), and tends to have higher accuracy in the 1-step and 32-step case, while FMLM+ retains a consistent and balanced accuracy for different numbers of steps. Our results are obtained with simple naive sampling, and we hypothesise that more advanced samplers could further improve the performance (*e.g.*, confidence-based).

E Embedded spaces

E.1 Reproducing SemiCat's statistics on an embedded space

Let us formally write out the interpolation incurred by SemiCat. Given i.i.d. times $\tau^\ell \sim \mathcal{U}(0, 1)$, at each position, we have, for $0 \leq t < \tau^\ell$,

$$I_t^\ell(X_0, X_1) = (1 - \gamma_t)X_0^\ell + \gamma_t Y_t^\ell, \tag{27}$$

and, for $t \geq \tau^\ell$,

$$I_t^\ell(X_0, X_1) = X_1^\ell. \tag{28}$$

Thus, conditioning on the same times, we have either, for unrevealed times,

$$WI_t^\ell(X_0, X_1) \sim \mathcal{N}\left(\gamma_t W Y_t^\ell, (1 - \gamma_t)^2 W^\top W\right). \tag{29}$$

Moreover, since

$$Y_t^\ell = (1 - \kappa_t)Y_0^\ell + \kappa_t X_1^\ell, \tag{30}$$

for a Dirichlet prior, the mean conditional on X_1^ℓ is

$$(1 - \kappa_t)W\frac{\mathbf{1}}{|V|} + \kappa_t W X_1^\ell, \tag{31}$$

the first term corresponding to the barycentre of the embeddings. For times after reveal, $I_t^\ell(X_0, X_1) = WX_1^\ell = \mathsf{emb}(X_1^\ell)$. The only difficulty to obtain the same statistics directly in the embedded space is to obtain the same covariance structure of the Gram matrix. To do so efficiently, we can compute $W^\top W$, which we can then decompose by Cholesky as $W^\top W = LL^\top$, and thus apply it to a Gaussian draw in lower dimension, LZ, for $Z \sim \mathcal{N}(0, \mathrm{I})$, thus obtaining a covariance of $LL^\top = W^\top W$. Our trained models obtain the same final performance as normal SemiCat training, but it is less stable due to the PSD requirement of Cholesky.

E.2 EMBEDDING-DEPENDENT SCHEDULES

The decoding error rate for one-hot encoded data has the well-known form from Pynadath et al. (2026); Sahoo et al. (2025). We give here the building block needed to generalise it to arbitrary embeddings: the pairwise nearest-neighbour comparison. The full DER then follows from the joint behaviour of these $|V| - 1$ comparisons, which we do not treat here as they are not independent.

Proposition 1 (Embedding-dependent pairwise decoding scores)**.** Given an embedding matrix $E \in \mathbb{R}^{|V| \times d}$, and the interpolant $I_t^\ell = (1-\gamma_t)X_0^\ell + \gamma_t \mathsf{emb}(X_1^\ell)$ between a standard Gaussian and the token's embedding, define the nearest-neighbour decoding scores $S_t(v) \coloneqq \left\|\mathsf{emb}(v) - I_t^\ell\right\|^2$. Then, for any pair of tokens $i \neq j$, writing $\Delta_{ij} \coloneqq \mathsf{emb}(j) - \mathsf{emb}(i)$,

$$\big(S_t(i) - S_t(j)\big) \sim \mathcal{N}\left(m_{ij}(t),\, 4(1-\gamma_t)^2 \left\|\Delta_{ij}\right\|^2\right), \tag{32}$$

and hence, since decoding is an argmin over squared distances, token i is preferred over token j with probability

$$\forall 0 \leq t \leq 1, \qquad \mathbb{P}\left(S_t(i) < S_t(j)\right) = 1 - \Phi\left(\frac{m_{ij}(t)}{2(1-\gamma_t)\left\|\Delta_{ij}\right\|}\right), \tag{33}$$

where Φ is the c.d.f. of the standard normal distribution and the mean score gap is

$$m_{ij}(t) \coloneqq 2\gamma_t \left\langle \Delta_{ij}, \mathsf{emb}(X_1^\ell)\right\rangle + \left\|\mathsf{emb}(i)\right\|^2 - \left\|\mathsf{emb}(j)\right\|^2. \tag{34}$$

Proof. Let $\ell \in \{1, \ldots, L\}$. We have that

$$I_t^\ell \sim \mathcal{N}\left(\gamma_t \mathsf{emb}(X_1^\ell), (1-\gamma_t)^2 \mathrm{I}\right). \tag{35}$$

For all $v \in V$, define the scores

$$S_t(v) = \left\|\mathsf{emb}(v) - I_t^\ell\right\|^2 = \left\|\mathsf{emb}(v)\right\|^2 + \left\|I_t^\ell\right\|^2 - 2\left\langle \mathsf{emb}(v), I_t^\ell\right\rangle. \tag{36}$$

For the nearest neighbour decoding to yield the true final token, X_1^ℓ, let us compare the pairwise errors, for indices $i \neq j$:

$$S_t(i) - S_t(j) = \left\|\mathsf{emb}(i)\right\|^2 - \left\|\mathsf{emb}(j)\right\|^2 - 2\left\langle \mathsf{emb}(i), I_t^\ell\right\rangle + 2\left\langle \mathsf{emb}(j), I_t^\ell\right\rangle. \tag{37}$$

Expanding the dot product with embedding i (or identically for j), by linearity,

$$\left\langle \mathsf{emb}(j), I_t^\ell\right\rangle = (1-\gamma_t)\left\langle \mathsf{emb}(j), X_0^\ell\right\rangle + \gamma_t \left\langle \mathsf{emb}(j), \mathsf{emb}(X_1^\ell)\right\rangle, \tag{38}$$

we can then rewrite the difference of scores, for $\Delta_{ij} \coloneqq \mathsf{emb}(j) - \mathsf{emb}(i)$, as

$$S_t(i) - S_t(j) = 2\gamma_t \left\langle \Delta_{ij}, \mathsf{emb}(X_1^\ell)\right\rangle + \left\|\mathsf{emb}(i)\right\|^2 - \left\|\mathsf{emb}(j)\right\|^2 + 2(1-\gamma_t)\left\langle \Delta_{ij}, X_0^\ell\right\rangle. \tag{39}$$

Since $X_0^\ell \sim \mathcal{N}(0, \mathrm{I})$, the last term follows $\mathcal{N}\left(0, 4(1-\gamma_t)^2\left\|\Delta_{ij}\right\|^2\right)$. Hence,

$$\big(S_t(i) - S_t(j)\big) \sim \mathcal{N}\left(m_{ij}(t), 4(1-\gamma_t)^2\left\|\Delta_{ij}\right\|^2\right), \tag{40}$$

where

$$\forall 0 \leq t \leq 1, \qquad m_{ij}(t) \coloneqq 2\gamma_t \left\langle \Delta_{ij}, \mathsf{emb}(X_1^\ell)\right\rangle + \left\|\mathsf{emb}(i)\right\|^2 - \left\|\mathsf{emb}(j)\right\|^2. \tag{41}$$

So the probability of the event of interest can be expressed as

$$\mathbb{P}\left(S_t(i) < S_t(j)\right) = \Phi\left(\frac{-m_{ij}(t)}{2(1-\gamma_t)\left\|\Delta_{ij}\right\|}\right) = 1 - \Phi\left(\frac{m_{ij}(t)}{2(1-\gamma_t)\left\|\Delta_{ij}\right\|}\right). \tag{42}$$

□

Recovering the one-hot DER. Informally, consider unit-norm embeddings with pairwise inner products $\rho_{ij} \coloneqq \langle \mathsf{emb}(i), \mathsf{emb}(j) \rangle$, $i \neq j$. If cross-entropy spreads the embeddings roughly uniformly over the sphere, then, since two independent uniform points on the sphere have an inner product of mean 0 and variance $1/d$, the ρ_{ij} are of order $1/\sqrt{d}$, and thus vanish as d grows. Taking $i = X_1^\ell$ to be the true token, the norm terms of Proposition 1 cancel, $\|\Delta_{ij}\|^2 = 2(1-\rho_{ij})$ and $m_{ij}(t) = -2\gamma_t(1-\rho_{ij})$, so that

$$\mathbb{P}\left(S_t(i) < S_t(j)\right) = \Phi\left(\frac{\gamma_t}{1-\gamma_t}\sqrt{\frac{1-\rho_{ij}}{2}}\right) \xrightarrow[\rho_{ij}\longrightarrow 0]{} \Phi\left(\frac{\gamma_t}{\sqrt{2}(1-\gamma_t)}\right), \tag{43}$$

the one-hot value. In the same limit, the comparisons also recover the correlation structure of the one-hot case, up to $O(\rho_{ij})$ corrections, so the full DER tends to the one-hot expression of Pynadath et al. (2026); Sahoo et al. (2025). An embedding spread out by cross-entropy therefore approximately inherits the one-hot DER.

E.3 Cross-entropy, mean squared error and the embedding geometry

To gain some intuition, we compare, informally and approximately, how the cross-entropy and the MSE terms of Plaid (Gulrajani & Hashimoto, 2023) move the input embeddings $W \in \mathbb{R}^{|V|\times d}$, with rows $\mathsf{emb}(v)$. Fix a position ℓ with true token $i = \mathbf{x}^\ell$, write $\sigma_t = 1-\gamma_t$ and $I_t^\ell = \sigma_t X_0^\ell + \gamma_t \mathsf{emb}(i)$. *We assume the denoiser is close to Bayes-optimal*, so that its prediction is the posterior

$$\pi_t^{\theta,\ell,v}(I_t) \propto p(v)\exp\left(-\left\|I_t^\ell - \gamma_t \mathsf{emb}(v)\right\|^2 / 2\sigma_t^2\right), \tag{44}$$

for some prior p over tokens. Denote by $\mu \coloneqq \pi_t^{\theta,\ell} W = \sum_v \pi_t^{\theta,\ell,v}\mathsf{emb}(v)$ the expected embedding, and observe that $\mu - \mathsf{emb}(i) = \sum_{v\neq i}\pi_t^{\theta,\ell,v}\Delta_{iv}$ points *towards* the tokens carrying the error mass.

The cross-entropy sees the embeddings only through the interpolant, through $\mathsf{emb}(i)$, and recall that X_1^ℓ is the one-hot random variable for the token at position ℓ (*i.e.*, $X_1^{\ell,\mathbf{x}^\ell} = 1$ and $X_1^{\ell,v} = 0$ for $v \neq \mathbf{x}^\ell$). Hence, omitting the arguments for the denoiser at time t, π_t^θ, for conciseness,

$$\begin{aligned}
\nabla_{\mathsf{emb}(i)}\left(-\log \pi_t^{\theta,\ell,i}\right) &= \gamma_t \nabla_{I_t^\ell}\left(-\log \pi_t^{\theta,\ell,i}\right) && (45)\\
&= \gamma_t \nabla_{I_t^\ell}\Bigg(\frac{\left\|I_t^\ell - \gamma_t\mathsf{emb}(i)\right\|^2}{2\sigma_t^2} \\
&\qquad + \log\sum_v p(v)\exp\left(-\frac{\left\|I_t^\ell - \gamma_t\mathsf{emb}(v)\right\|^2}{2\sigma_t^2}\right)\Bigg) && (46)\\
&= \frac{\gamma_t}{\sigma_t^2}\left(\left(I_t^\ell - \gamma_t\mathsf{emb}(i)\right) - \sum_v \pi_t^{\theta,\ell,v}\cdot\left(I_t^\ell - \gamma_t\mathsf{emb}(v)\right)\right) && (47)\\
&= \frac{\gamma_t}{\sigma_t^2}\left(\sum_v X_1^{\ell,v}\left(I_t^\ell - \gamma_t\mathsf{emb}(v)\right) - \sum_v \pi_t^{\theta,\ell,v}\cdot\left(I_t^\ell - \gamma_t\mathsf{emb}(v)\right)\right) && (48)\\
&= \frac{\gamma_t}{\sigma_t^2}\sum_v\left(X_1^{\ell,v} - \pi_t^{\theta,\ell,v}\right)\left(I_t^\ell - \gamma_t\mathsf{emb}(v)\right) && (49)\\
&= \frac{\gamma_t}{\sigma_t^2}\left(I_t^\ell \underbrace{\sum_v\left(X_1^{\ell,v} - \pi_t^{\theta,\ell,v}\right)}_{=0} - \gamma_t\sum_v\left(X_1^{\ell,v} - \pi_t^{\theta,\ell,v}\right)\mathsf{emb}(v)\right) && (50)
\end{aligned}$$

$$\nabla_{\mathsf{emb}(i)}\left(-\log \pi_t^{\theta,\ell,i}\right) = \frac{\gamma_t^2}{\sigma_t^2}\left(\mu - \mathsf{emb}(i)\right). \tag{51}$$

For (47), we used the Bayes optimality of the denoiser. Writing $a_v \coloneqq \left\|I_t^\ell - \gamma_t\mathsf{emb}(v)\right\|^2/2\sigma_t^2$, so that $\nabla_{I_t^\ell} a_v = (I_t^\ell - \gamma_t\mathsf{emb}(v))/\sigma_t^2$,

$$\nabla_{I_t^\ell}\log\sum_v p(v)e^{-a_v} = -\sum_v \frac{p(v)e^{-a_v}}{\sum_u p(u)e^{-a_u}}\nabla_{I_t^\ell}a_v = -\frac{1}{\sigma_t^2}\sum_v \pi_t^{\theta,\ell,v}\cdot\left(I_t^\ell - \gamma_t\mathsf{emb}(v)\right), \tag{52}$$

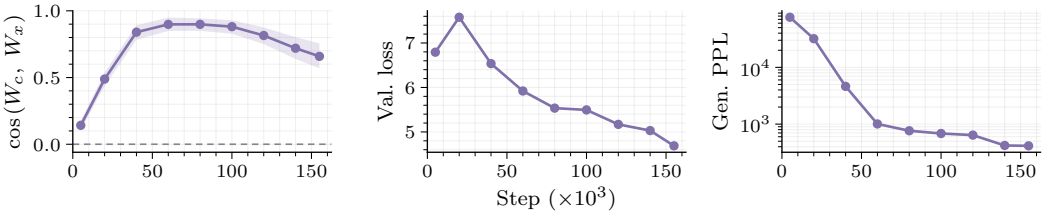


Figure 8: Statistics across training steps, for a Plaid model trained on OWT. **(Left)** The evolution of cosine similarity between W_c (the part of the input matrix for self-conditioning) and W_x (the part that multiplies the input). **(Middle)** The validation loss. **(Right)** The Gen. PPL.

with the last equality coming from the definition of the Bayes optimal denoiser. The wrong embeddings, $\mathsf{emb}(v)$ for $v \neq i$, receive no gradient from the cross-entropy through the input, since they do not enter I_t^ℓ. On the other hand, for the MSE term, $\frac{1}{2} w_t \left\| (\pi_t^{\theta,\ell} - X_1^\ell) W \right\|^2$, with weight $w_t \geq 0$ for all t, and *ignoring the dependency of* $\pi_t^{\theta,\ell}$ *on* $\mathsf{emb}(i)$ *through* I_t^ℓ,

$$\nabla_{\mathsf{emb}(v)} \frac{1}{2} w_t \left\| (\pi_t^{\theta,\ell} - X_1^\ell) W \right\|^2 = w_t \left(\pi_t^{\theta,\ell,v} - X_1^{\ell,v} \right) (\mu - \mathsf{emb}(i)) \,. \tag{53}$$

The cross-entropy gradient and the simplified MSE gradient act on $\mathsf{emb}(i)$ along the same direction, $\mu - \mathsf{emb}(i)$, but with opposite signs. A descent step on the cross-entropy moves the true token *away* from the predicted mean—*i.e.*, away from the tokens it is confused with—weighted by how much it is confused with them, and without preference for any other structure, thus spreading the embeddings apart. A descent step on the MSE instead moves $\mathsf{emb}(i)$ *towards* μ with weight $1 - \pi_t^{\theta,\ell,i}$, and every confused token $\mathsf{emb}(v)$ towards $\mathsf{emb}(i)$ with weight $\pi_t^{\theta,\ell,v}$, drawing them together; on its own, it is trivially minimised by the collapse $W = 0$.

Again, *the above is only informal*: a trained network merely approximates the posterior, and independence of the denoiser on $\mathsf{emb}(i)$ is a strong simplification, for no stop-gradient operator is typically applied there.

E.4 Plaid self-conditioning incorporation

With a self-conditioning input signal $\hat{x}_1$, alongside the usual input x_t, the Plaid architecture forwards the two inputs to the rest of the diffusion transformer as

$$h_0^\ell = \begin{bmatrix} W_c & W_x \end{bmatrix} \begin{pmatrix} \hat{x}_1 \\ x_t \end{pmatrix} . \tag{54}$$

It thus doubles the size of the effective input matrix (*i.e.*, from $d \times h$ to $2d \times h$).

E.5 Embedded self-conditioning: beyond Euler steps

Another observation which is consistent with our understanding from Section 4.3 is in Figure 8. Observe that one can split the input projection matrix in two parts: the one that multiplies the usual input, W_x, and the one that multiplies the self-conditioning, W_c (see Appendix E.4 for additional detail). We can then measure the cosine similarity between their vectorised forms. For the first 80k steps, the model converges to representations that are highly aligned (with a cosine similarity of 0.9), and the input projection thus amounts approximately to a linear combination followed by a common projection thereon. After that, the loss, performance and the embedding alignment of the model plateau for roughly 30k iterations, and then continue improving anew when the cosine similarity drops down to 0.6, alluding to the fact that the model learns a more complex function than a mere Euler-like step.

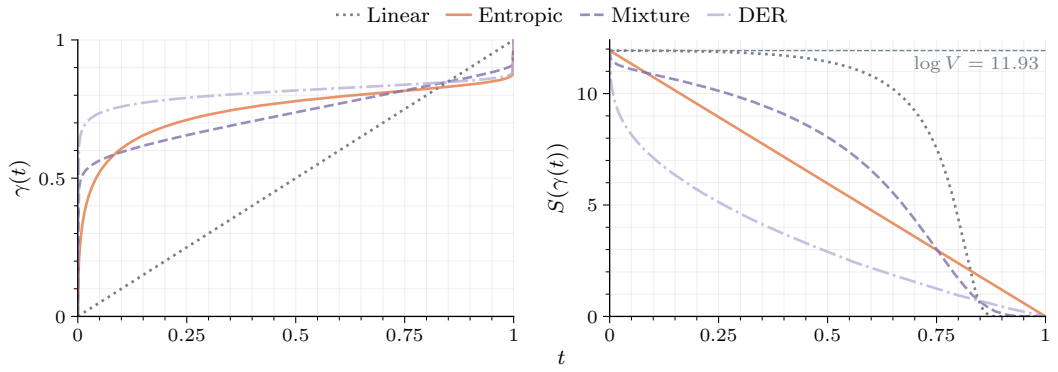


Figure 9: The entropic schedule against the base DER, mixture and linear schedules. **(Left)** The schedules themselves. **(Right)** The expected remaining entropy, $S(\gamma_t) = \mathbb{E}_{X_0^\ell}[H(X_1^\ell \mid I_t^\ell)]$.

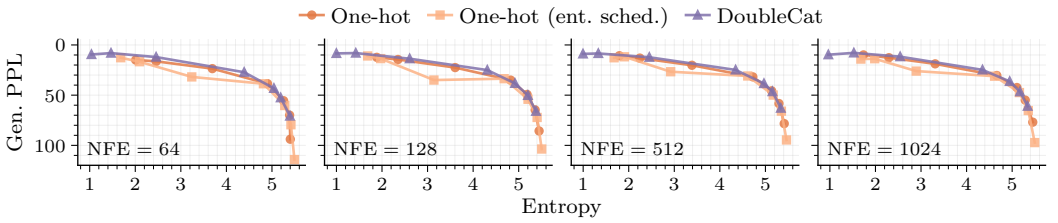


Figure 10: Gen. PPL as measured by GPT-2 for models trained on OpenWebText with sequence lengths of 1024 tokens, using the GPT-2 tokeniser.

F Further schedules

F.1 Entropic schedule

We attempt, here, to understand why the mixture schedule performs better than the raw DER schedule (Lee et al., 2026; Potaptchik et al., 2026b; Davis et al., 2026). For the usual one-hot flow case, the true Bayesian posterior reads

$$\forall 0 \leq t \leq 1, \qquad D_t^\ell = \text{softmax}\left(\frac{\gamma_t I_t^\ell}{(1-\gamma_t)^2}\right). \tag{55}$$

Let $S : [0,1] \longrightarrow \mathbb{R}_+$ be the expected remaining entropy; that is to say,

$$\forall 0 \leq t \leq 1, \qquad S(\gamma_t) \coloneqq \mathbb{E}_{X_0^\ell}\left[H(X_1^\ell \mid I_t^\ell)\right]. \tag{56}$$

Instead of linearising directly the schedule in the probability of $\text{argmax}\ I_t^\ell \neq X_1^\ell$ as the DER schedule does, we propose to choose a schedule to advance linearly in S. Concretely, we are looking for a solution, $\gamma : [0,1] \longrightarrow [0,1]$, to the following:

$$\forall 0 \leq t \leq 1, \qquad S(\gamma_t) = (1-t)\log|V|. \tag{57}$$

The solution has no closed form, so we instead build it with a look up table, which incurs a constant cost once before training, exactly as the DER schedule. Let us call the solution the "entropic" schedule. We plot the DER, mixture and entropic schedules in Figure 9, alongside the resulting S, for each choice of γ. We remark that the mixture schedule in fact closely follows the entropic schedule both in γ and $S \circ \gamma$. Furthermore, we also plot the Gen. PPL/Entropy Pareto fronts of a one-hot flow with the mixture schedule, the entropic schedule and DoubleCat in Figure 10. While it closely matches the curve of the mixture schedule, it exhibits an outlier near 3 nats in entropy, especially at low NFE; however, we report that the resulting performance on TinyGSM is the same as that of the mixture schedule.

We believe that this entropic schedule and the mixture one allude to the fact that making uniform progress in the DER *probability* P_e might be the wrong criterion.

F.2 ASYNCHRONOUS SCHEDULES

While appealing, discrete reveals incur two properties that render the usage of usual continuous diffusion tools difficult: the path is now instead *piecewise* continuous, with discontinuity points that are *stochastic*. For example, the formalism from Flow Map Matching (Boffi et al., 2025a;b) then does not port trivially, because of the deterministic and continuous requirements on the flow map.

To return to a continuous and (conditionally on the endpoints) deterministic process, we note that the revealing procedure can be seen as a limiting case of *asynchronous schedules* with added randomness. Instead of having a single time for the evolution of the interpolant, we can have a distinct time for each position, t^ℓ. In principle, a schedule that depends on the current state or the model's denoising distribution is appealing, as parts of the sequence may benefit from being committed earlier than others. Nonetheless, training such a schedule either requires two forward passes from the denoising model, π^θ, or training uniformly on the $[0,1]^L$ hypercube to produce a model that is agnostic to the specific choice of schedule; but the latter exposes the model to sequences that have uninformative statistics for inference.

We thus decide to solve this tension by resolving different tokens at different speeds, following Wu et al. (2023). Specifically, local times t^ℓ are derived from a global time t^g, *i.e.*, $t^\ell = F(t^g, \ell)$, which is monotonic in ℓ and approximately mean-preserving across positions, so that averaging t^ℓ over ℓ recovers t^g. The results are also available in Table 11, with inner schedule $\kappa_t = t$. Asynchronous times improve accuracy, valid programs and name errors in three of the four matched settings, the exception being the linear schedule with InfoNoise. InfoNoise raises accuracy almost uniformly, but for the mixture schedule it lowers the fraction of valid programs, so that the best accuracy and the fewest name errors are reached by different configurations. All settings remain well below $\mathrm{SemiCat}$ (Table 1); we hypothesise that crafting a better time schedule is crucial here.

Implementation. We follow Wu et al. (2023) and the specific parametrisation by Bansal & Sanghavi (2026). We introduce a global time t^g that we sample as before: according to some density $p(t)$ that is often to be taken uniform, but in the case of InfoNoise can be adapted dynamically during training. During inference we step uniformly in t^g. For each position $\ell = 1, \dots, L$ we introduce a local time $t^\ell = F(t^g, \ell)$ that is a function of the global time and the position. Each position ℓ gets a fixed rank $r_\ell \in (0,1)$, given by $r_\ell = 1 - (\ell - 0.5)/L$. The rank is then pushed through the quantile function of a Beta distribution whose mean is t^g: $t^\ell = Q_t(r_\ell)$, where Q_t is the quantile function of $\mathrm{Beta}(k/(1-t), k/t)$ where k specifies its concentration: as $k \longrightarrow \infty$ we recover a shared time $t^\ell = t^g$. As $k \longrightarrow 0$, Beta tends to a Bernoulli with mean t^g, so position ℓ switches from noise to data exactly when t^g crosses $(\ell - 0.5)/L$: a deterministic left-to-right reveal. In our experiments we use $k = 1$; the resulting local times are shown in Figure 11.

F.3 ADAPTIVE ALLOCATION OF TRAINING DENSITY

Like Deschenaux & Gulcehre (2026); Batzolis et al. (2026) we adapt InfoNoise, the information-guided time allocation of Raya et al. (2026), to the cross-entropy objective used in our model. The adaptation changes only the distribution $\pi(t)$ from which training times are sampled; the loss weighting itself remains fixed. Writing $\bar{\mathcal{L}}(t)$ for the expected loss at time t, the objective thus changes from $\int_0^1 p(t)\bar{\mathcal{L}}(t)\,\mathrm{d}t$ to $\int_0^1 \pi(t)\bar{\mathcal{L}}(t)\,\mathrm{d}t$, where p is the base (uniform) density.

For a fixed reference batch $\mathcal{B}_{\mathrm{ref}}$ of size $|\mathcal{B}_{\mathrm{ref}}| = 256$ (a single batch), we periodically estimate the cross-entropy across a grid of corruption times $\{t_i\}_{i=1}^M$,

$$\hat{H}(t_i) = \frac{1}{|\mathcal{B}_{\mathrm{ref}}|} \sum_{x \in \mathcal{B}_{\mathrm{ref}}} \mathcal{L}_{\mathrm{CE}}(x, t_i). \tag{58}$$

We measure at $M = 32$ levels: 24 at equal mass of the current density, plus 8 spaced uniformly over the interior of $[0,1]$. Cross-entropy upper-bounds the conditional entropy of the clean token given its corrupted input, with equality for the optimal denoiser, so we use $\hat{H}(t)$ directly as a proxy for the conditional-entropy profile. This differs from applications of InfoNoise to continuous data, where the information profile must first be estimated indirectly from a denoising-error statistic.

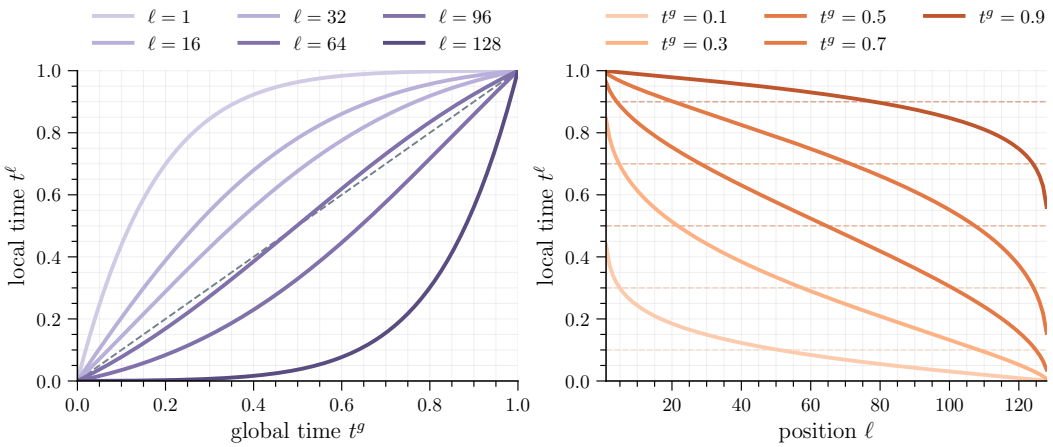


Figure 11: Local per-position times t^ℓ as a function of global time t^g at concentration $k = 1$ and sequence length $L = 128$. **(Left)** Each curve traces one position as global time advances. **(Right)** Each curve is the spatial profile at one global time, monotone decreasing in ℓ: the sequence is revealed left to right. The dashed horizontals mark the per-position mean, which equals t^g at every setting.

The measured values are projected onto a monotone profile and interpolated with a cubic spline, yielding a smooth estimate $\widetilde{H}(t)$. We then construct an information-rate profile from its derivative,

$$r(t) = \left| \frac{\mathrm{d}\widetilde{H}(t)}{\mathrm{d}t} \right|, \tag{59}$$

and allocate training times in proportion to this rate. Thus, corruption regions over which the conditional entropy changes rapidly receive more training samples, while regions in which little information is gained or lost receive proportionally fewer.

To avoid starving regions assigned very low probability, we constrain the resulting density before normalisation. Denoting the unconstrained allocation by $\widetilde{\pi}(t) \propto r(t)$, we use

$$\pi^{\mathrm{new}}(t) = \mathrm{normalise}\left[\mathrm{clip}\left(\widetilde{\pi}(t), \pi_{\min}, \pi_{\max}\right)\right], \tag{60}$$

with $\pi_{\min} = 0.1$ and $\pi_{\max} = 10$. The lower threshold ensures that every corruption level continues to receive training signal, which is particularly important for an online allocation because a region that is never revisited cannot subsequently develop an estimated information rate that would cause it to regain probability mass. The upper threshold prevents the allocation from becoming excessively concentrated.

We represent the sampling distribution on a fixed grid of 256 points. After each update, the density is numerically integrated on this grid using the trapezoidal rule to obtain a cumulative distribution function F^{new}. Rather than replacing the current allocation immediately, we damp successive updates in CDF space,

$$F^{(n)} = (1 - \lambda)F^{(n-1)} + \lambda F^{\mathrm{new}}, \qquad \lambda = 0.5. \tag{61}$$

Averaging normalised cumulative distributions rather than raw loss or entropy values makes the update insensitive to changes in the overall scale of the loss during training and smooths changes in where probability mass is allocated. Using a fixed grid likewise provides a common coordinate system across successive measurements.

We recompute the entropy profile and update the allocation every $n_{\mathrm{dynamic}} = 2500$ training iterations. Between updates, corruption times are sampled from the current tabulated CDF by inverse-transform sampling,

$$u \sim \mathcal{U}(0, 1), \qquad t = \left(F^{(n)}\right)^{-1}(u), \tag{62}$$

where the inverse is obtained by linear interpolation between neighbouring points. The resulting samples determine only which corruption times are presented during subsequent training iterations; all other components of the objective are unchanged.

InfoNoise	Schedule	Asynchronous	**Accuracy** (%, ↑)				**Best** (%)	
			$T = 1$	$T = 0.5$	$T = 0.2$	top-k=1	Runs (↑)	Name err. (↓)
Off	Linear	Off	6.82	9.40	11.14	9.40	63.61	36.01
		On	**7.88**	**10.99**	**12.28**	**10.99**	**66.26**	**33.43**
	Mixture	Off	10.39	14.78	13.65	13.87	68.16	31.54
		On	**12.21**	**15.47**	**16.00**	**14.63**	**73.39**	**26.38**
On	Linear	Off	9.63	17.44	**18.42**	15.69	**67.78**	**31.69**
		On	**11.37**	**18.20**	17.82	**17.44**	66.11	33.43
	Mixture	Off	10.84	16.45	14.48	13.80	62.85	36.85
		On	**15.54**	**19.11**	**19.41**	**18.04**	**68.23**	**31.31**

Table 11: TinyGSM accuracy at NFE $= 256$. For linear and mixture outer schedules γ_t and inner schedule $\kappa_t = t$, we compare a scalar time shared across all positions against an asynchronous left-to-right time per token position, each with and without dynamic allocation via InfoNoise. All settings are $\mathrm{DoubleCat}$ sampled as a one-hot flow. Better of each matched pair in each column is shown in **bold**.