Advancing Continuous Diffusion Language Models 🌊

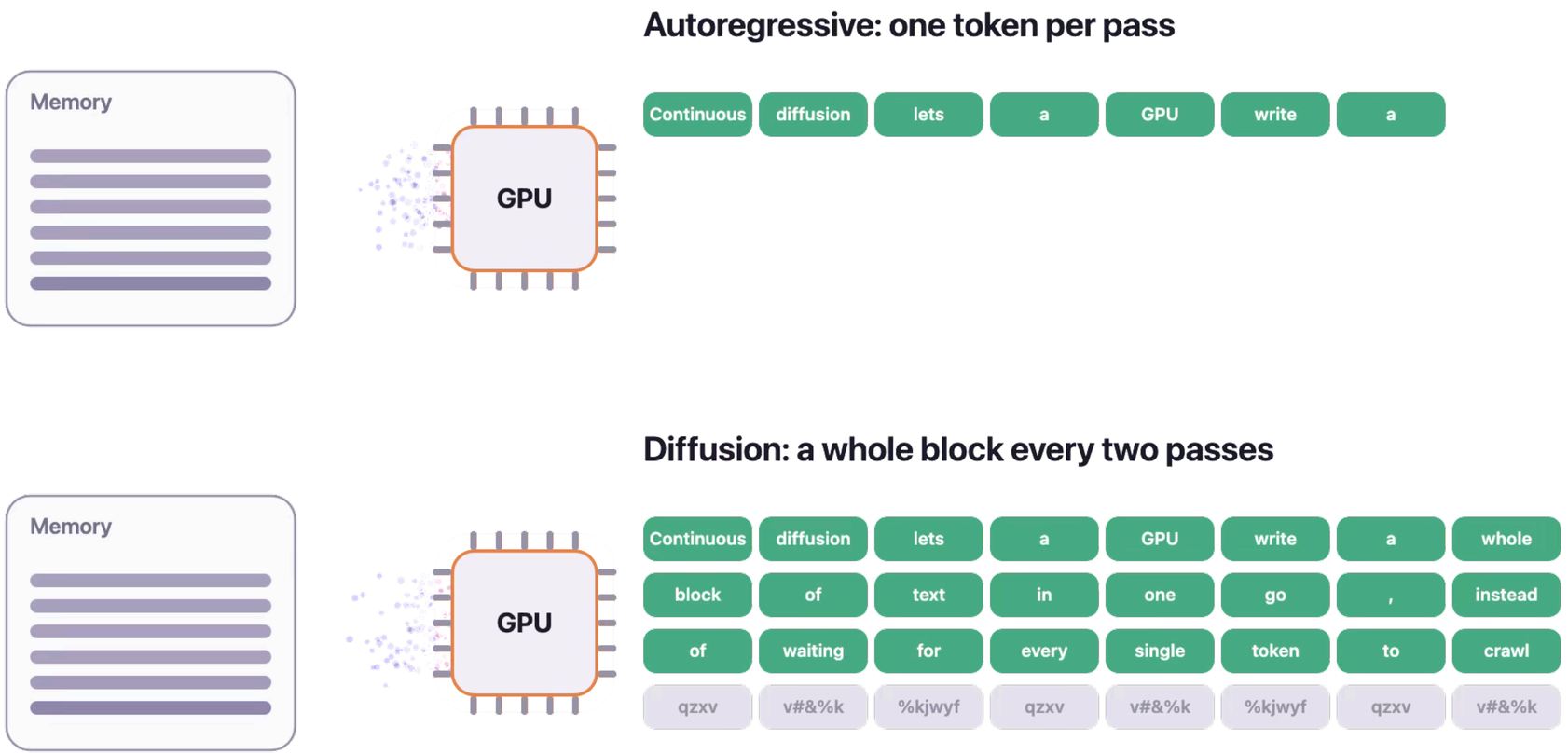
State of Affairs

Oscar Davis, University of Oxford, 19/11/2026

Motivation

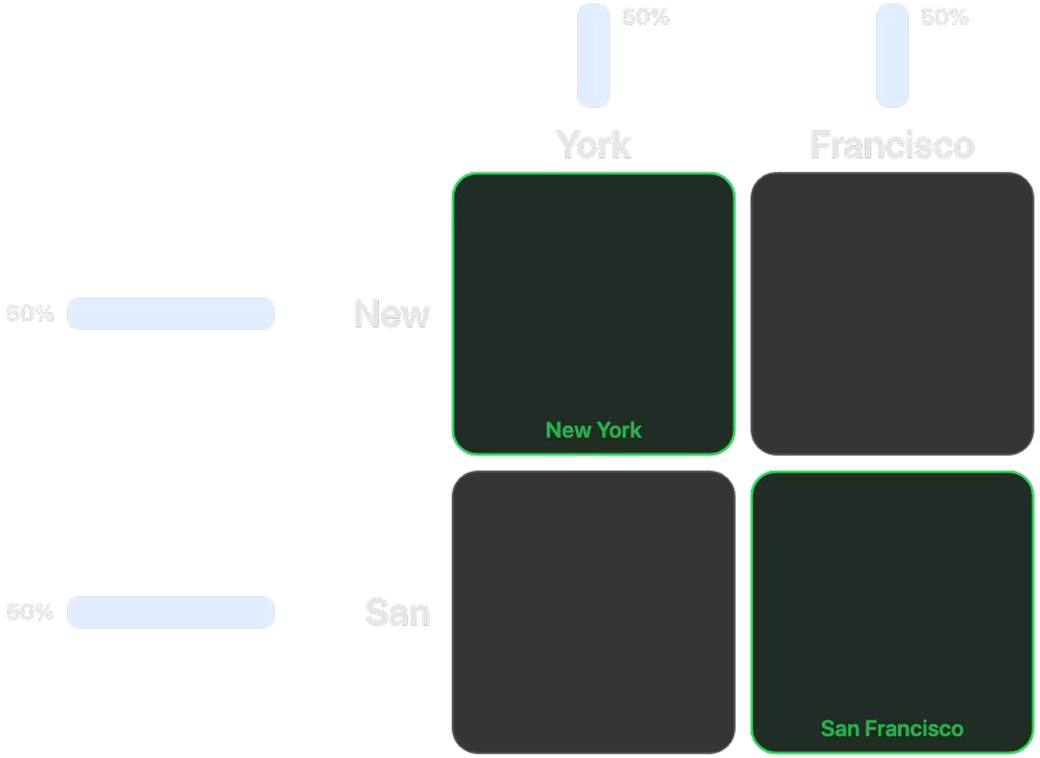
Local accelerated inference

Batch size 1



Discrete Diffusion?

Name a random city: [mask] [mask]





Steering, tilting, fine-tuning

Sampling from desired distributions

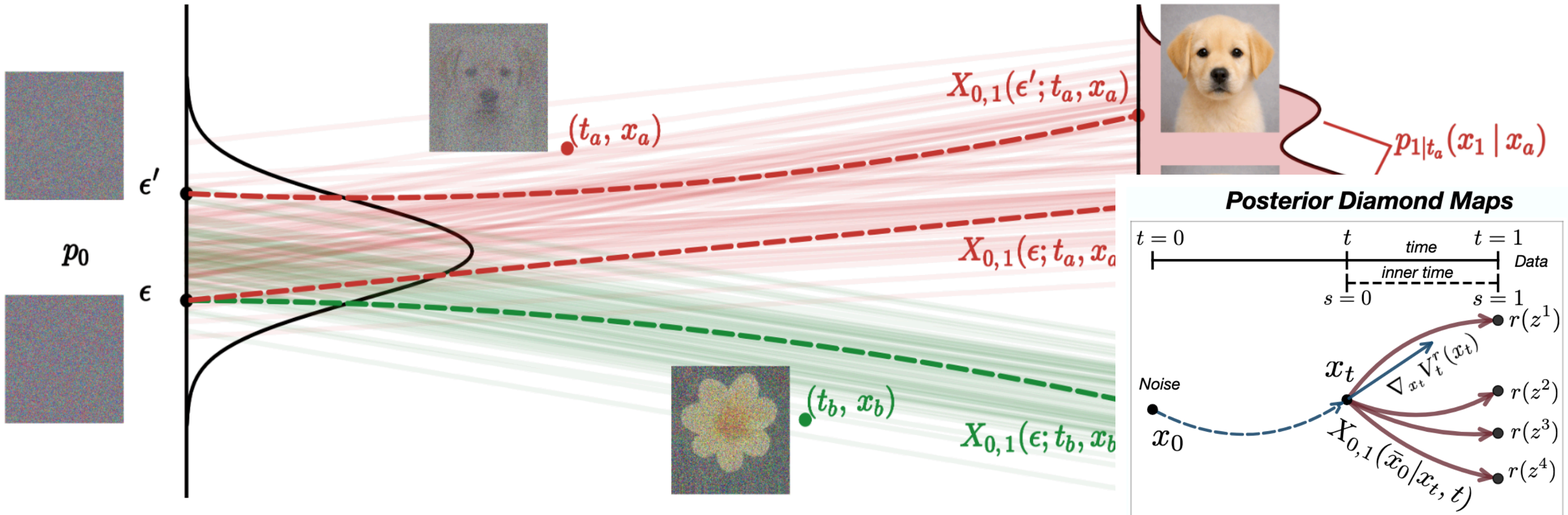


Figure 2 *An MFM X conditions on an intermediate time–state pair (t, x) (visualised as interpolant and learns a shared conditional flow $X_{s,u}(\cdot\,; t, x)$ that maps base noise $\epsilon \sim p_0$ (visualised as clean images) via $X_{0,1}(\epsilon; t, x)$. For a given (t, x) pair, varying the initial no from the same posterior $p_{1|t}(\cdot|x)$. Conversely, for the same initial noise ϵ, conditioning (t_a, x_a) and (t_b, x_b) yields one sample from each of two different posteriors $p_{1|t_a}(\cdot|x_a)$ a*

Meta Flow Maps, Potaptchik et al. (2026)

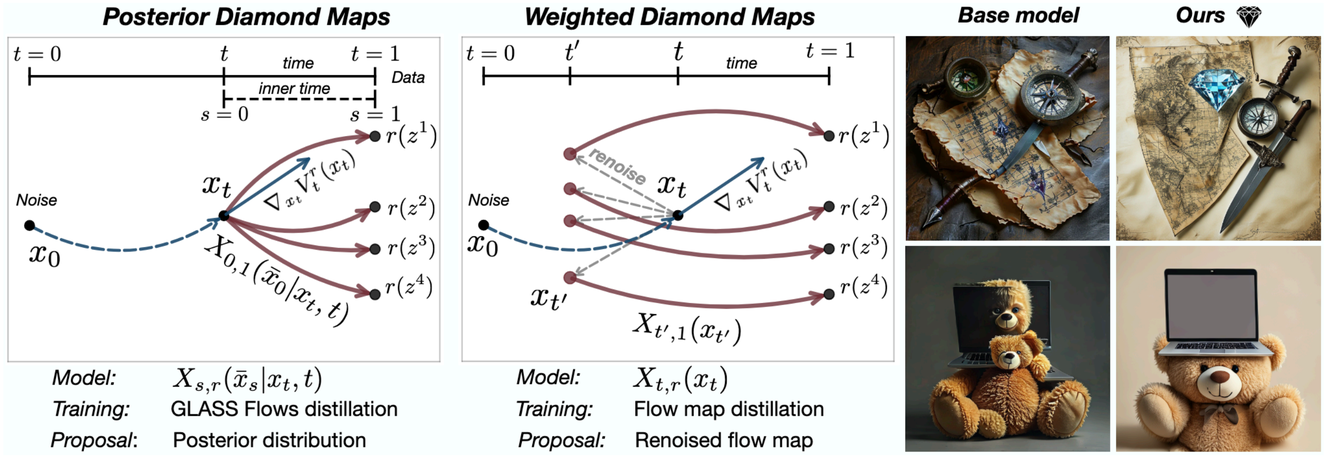


Figure 1. **Overview of Diamond Maps.** Diamond Maps are stochastic flow maps that allow to perform one-step "look-aheads" of a flow trajectory (blue) to potential endpoints at time 1 to evaluate a reward r. This allows for efficient exploration, search, and guidance. We propose 2 Diamond Map Designs. *Posterior Diamond Maps* (left) distill GLASS Flows into a flow map $X_{s,s'}(\bar{x}|x_t, t)$ designed to sample exact samples from the posterior. *Weighted Diamond Maps* (middle) allow to use standard flow maps by making them stochastic via a simple renoising procedure. Right: Improved image alignment with Diamond Maps. Prompts: "A diamond, a folded treasure map, a compass, and a dagger". "A laptop on top of a teddy bear".

Diamond Maps, Holderrieth et al. (2026)

Quick recap on Diffusion/Flow Matching

The setting 🛠️

Diffusion, flow matching, stochastic interpolants, etc.

- Modelling a distribution, $p_{\text{data}} \equiv p_1$, through an ODE:

$$\dot{x}_t = v_t(x_t) \text{ with } x_0 \sim p_0 = \mathcal{N}(0, \mathrm{I})$$

- We want $x_1 \sim p_1$.

$$I_t(X_0, X_1) = (1 - \gamma_t)X_0 + \gamma_t X_1.$$

with $I_0 = X_0 \sim p_0$ and $I_1 = X_1 \sim p_1$.

- For $v_t(x) := \mathbb{E}[\dot{I}_t \mid I_t = x]$, we have that $x_1 \sim p_1$.

Training 🏋️

Simple objective

- Training then reduces to

$$\mathscr{L}(\theta) = \mathbb{E} \left\| v_t^{\theta}(I_t) - \dot{I}_t \right\|^2$$

with $X_0 \sim p_0, X_1 \sim p_1, t \sim \mathscr{U}(0,1)$.

Training 🏋️

Endpoint training

- We know that $\mathbb{E}\left[\dot{I}_t \mid I_t = x\right] = \dot{\gamma}_t \dfrac{\mathbb{E}\left[X_1 \mid I_t = x\right] - x}{1 - \gamma_t}$.

- So, train for denoising:

$$\mathscr{L}(\theta) = \mathbb{E}\|X_1 - \pi_t^\theta(x)\|^2, \text{ with } v_t^\theta(x) = \dot{\gamma}_t \frac{\pi_t^\theta(x) - x}{1 - \gamma_t}.$$

Inference

Simple Euler integration

- Draw $x_0 \sim p_0$.
- For $T = 1/\tau$ steps,

$$x_{t+\tau} = x_t + \tau v_t^\theta(x_t).$$

References 📋

To dive deeper

- Flow Matching for Generative Modeling, Lipman et al. (2022).
- Stochastic Interpolants: A Unifying Framework for Flows and Diffusions, Albergo et al. (2023).

Continuous Diffusion Language Models

The idea is not new! 💡

A tiny bit of history

- The earliest mention in *recent* diffusion models is in DDIM (2020), as far as I know:

> A NON-MARKOVIAN FORWARD PROCESSES FOR A DISCRETE CASE
>
> In this section, we describe a non-Markovian forward processes for discrete data and corresponding variational objectives. Since the focus of this paper is to accelerate reverse models corresponding to the Gaussian diffusion, we leave empirical evaluations as future work.

- It is also mentioned in Sohl-Dickenstein et al. (2015)!
- Also thoroughly worked out by Dieleman et al. (2022; CDCD).

So why is it only taking off now?!

Categorical Flow Maps (2026) 🔥

What brought back the clout

- What if we did the simplest thing possible?
- And then finally tried to get some continuous advantage to work?
  - (*e.g.*, acceleration)

Concurrent: Flow Map Language Models by Lee et al. (2026); followed by Discrete Flow Maps by Potaptchik et al. (2026).

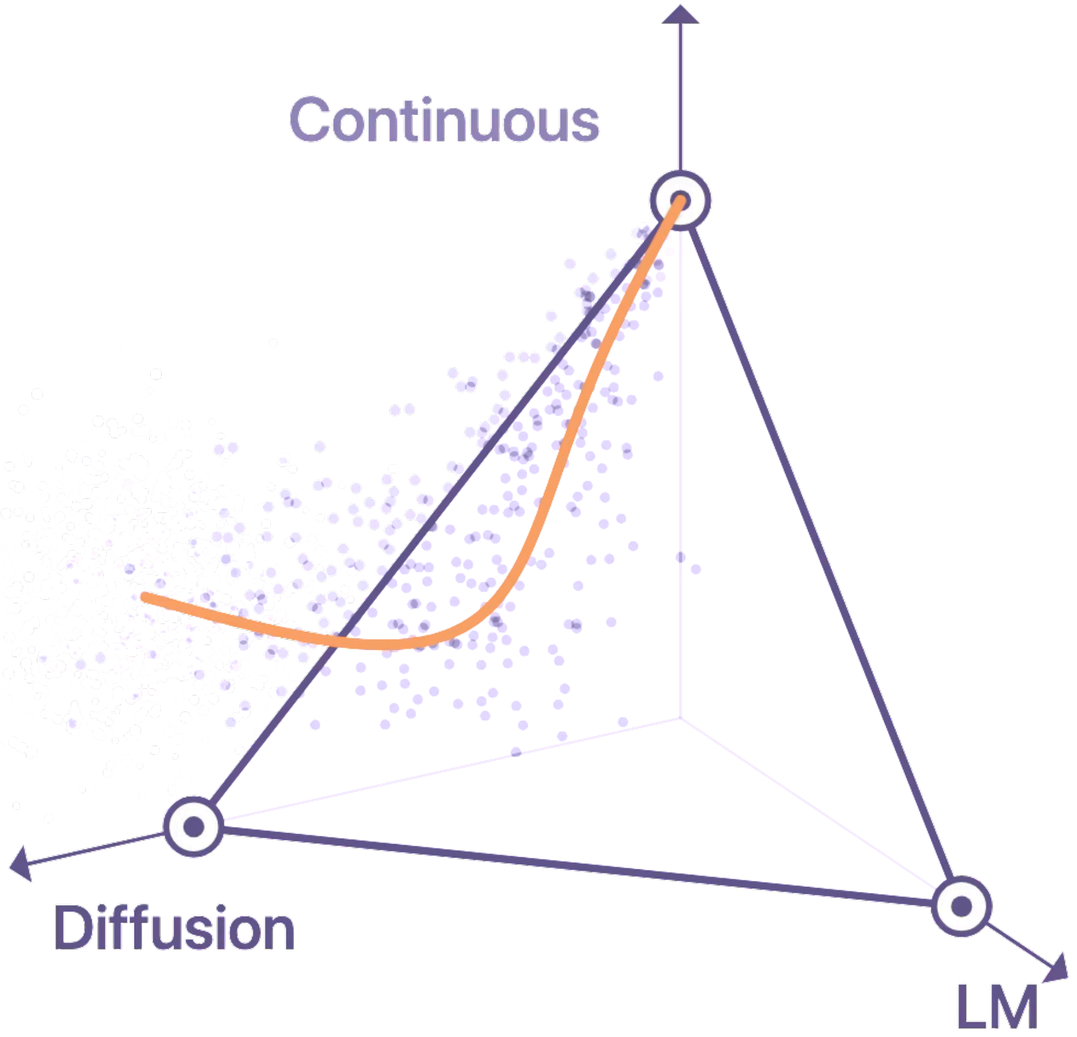
Variational Flow Matching

Eijkelboom et al. (2024)

- Let's one-hot encode our data, $X_1 \in \mathbb{R}^{L \times |V|}$.
- And do flow matching as before!
- Data is discrete, so:

$$\mathscr{L}(\theta) = -\mathbb{E} \sum_{\ell=1}^{L} \log \langle \pi_t^{\theta,\ell}(I_t), X_1^{\ell} \rangle.$$

… That's it!



How to accelerate a flow model?

Flow Map Matching by Boffi et al. (2024, 2025)

- We call the *flow map* the function, $X : [0,1]^2 \times \mathbb{R}^d \to \mathbb{R}^d$, such that

$$X_{s,t}(x_s) = x_t, \text{ for any solution of the flow, } (x_t)_{t\in[0,1]}.$$

- One could then sample by taking $X_{0,1}(x_0) = x_1$.

- How can we learn this function?

Flow Maps

Vector field parametrisation

- We parametrise

$$X_{s,t}(x) = x + (t - s)v_{s,t}(x).$$

- Then it follows that

$$v_{t,t}(x) = v_t(x).$$

Flow Maps

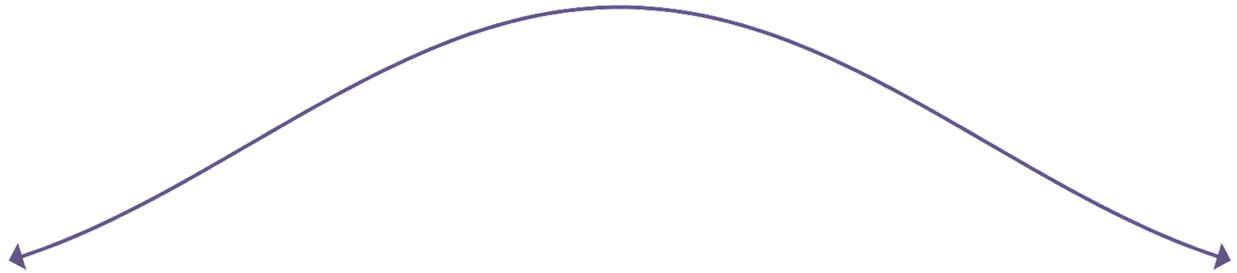
Progressive self-distillation

Since

$$X_{s,t}(x_s) = X_{u,t}(X_{s,u}(x_s)),$$

then

$$v_{s,t}(x_s) = (1-\alpha)v_{s,u}(x) + \alpha v_{u,t}(X_{s,u}(x_s)).$$



Flow Maps

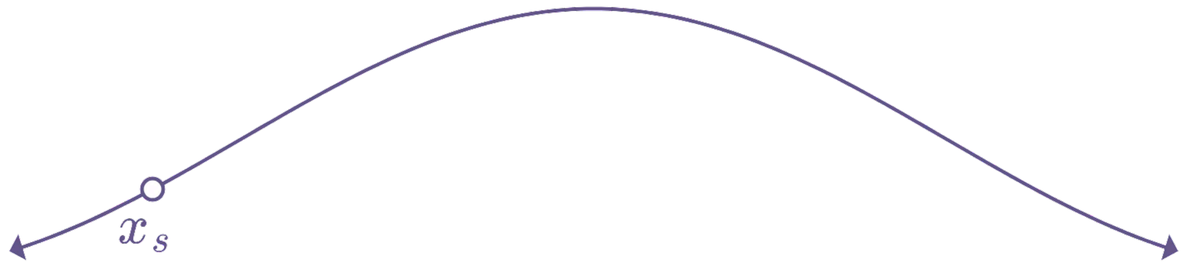
Lagrangian self-distillation

Since

$$X_{s,t}(x_s) = x_t$$

Taking derivatives,

$$\partial_t X_{s,t}(x) = v_{t,t}(X_{s,t}(x)).$$



$$\partial_t X_{s,t}(x_s) = v_{t,t}\big(X_{s,t}(x_s)\big)$$

Flow Maps

Summary – Learning

- Suffices then to enforce the usual flow matching loss for $s = t$ (diagonal):

$$\mathscr{L}_{\text{diag}}(\theta) = \mathbb{E} \left\| v^{\theta}_{t,t}(I_t) - \dot{I}_t \right\|^2$$

- Alongside a self-distillation objective (sg is stop-grad.):

$$\mathscr{L}_{\text{sd}}(\theta) = \mathbb{E} \left\| \partial_t X^{\theta}_{s,t}(I_s) - \text{sg}\left(v^{\theta}_{t,t}(X^{\theta}_{s,t}(I_s)) \right) \right\|^2$$

Flow Maps

Summary – Sampling

- Sampling can then be done over any grid $0 = \tau_1 < \ldots < \tau_N = 1$.
- Starting from $x_0 \sim p_0$,

$$x_{\tau_{k+1}} \leftarrow X^{\theta}_{\tau_k, \tau_{k+1}}(x_{\tau_k}).$$

So can’t we just apply it?

Back to Categorical Flow Maps

- We were learning a denoiser.
- Learning the vector field does not work well!
  - $|V| \approx 50$k, so $x_1 - x_0$ is dominated in norm by x_0.
- Two-time vector field vs. a one-time denoiser…

Endpoint Flow Maps

The fix!

- Instead of having a two-time vector field, introduce a two-time denoiser!
- Formally, $\pi : [0,1]^2 \times \mathbb{R}^d \to \mathbb{R}^d$ is such that

$$v_{s,t}(x) = \frac{\pi_{s,t}(x) - x}{1 - s}.$$

Recall that $v_t(x) = \dfrac{\pi_t(x) - x}{1 - t}$.

Learning the two-time denoiser

The same self-distillation objectives hold!

- For instance, the Lagrangian self-distillation is still valid.

$\longrightarrow$ We use that!

Categorical Flow Maps

ICML 2026

Categorical Flow Map

Real 1-step samples

$X_{0,1}(x_0) = x_1$

Uniform Discrete Diff.

Illustrative sample

$x_s \mid x_t, x_0 \sim \mathrm{Cat}(\ldots)$

```
ObroUQ4fAbnFbmOInLYaYRvj8vfg1MYTH9xIOM2K
ScosfogrOwxnq7OMkTkx OJR1Woctqn73tPy5Cqp
JqK4zn GfcgNXAey H92aTQHqOhBC83qFl52sOMj
WHXHaub630td37fEeVWiDIq2AnHTmt OBGhovLon
eNo52eoQni6JDXYlgBADTQ gzv2plCArp6B2Id 0
961kEnzdx qXCA8 JTjs9KIVdKF1j8flLQoh4pLN
AKG6mTpyQstVeC  enq633xjOSM Pa...
```

Categorical Flow Maps

Recap of the algorithm

- Training losses:

$$\mathscr{L}_{\text{diag}}(\theta) = -\,\mathbb{E} \sum_{\ell=1}^{L} \log\langle \pi_t^{\theta,\ell}(I_t), X_1^{\ell} \rangle.$$

$$\mathscr{L}_{\text{sd}}(\theta) = \mathbb{E} \left\| \partial_t X_{s,t}^{\theta}(I_s) - \text{sg}\left(v_{t,t}^{\theta}(X_{s,t}^{\theta}(I_s)) \right) \right\|^2$$

- Combined as

$$\mathscr{L}(\theta) = \mathscr{L}_{\text{diag}}(\theta) + \mathscr{L}_{\text{sd}}(\theta).$$

Some experiments

Quick glance – graphs

Method	NFE	QM9			ZINC		
		Valid ↑	Unique ↑	FCD ↓	Valid ↑	Unique ↑	FCD ↓
Multi-Step Diffusion / Flow (reference)							
GDSS (Jo et al., 2022)	1,000	95.7	98.5	2.90	97.0	99.6	14.7
GruM (Jo et al., 2024)	1,000	99.7	–	**0.11**	98.7	–	2.26
CatFlow (Eijkelboom et al., 2024)	100	**99.8**	**99.9**	0.44	**99.2**	**100.0**	13.2
DeFoG (Qin et al., 2025)	500	99.4	96.3	0.12	**99.2**	99.9	**1.45**
DeFoG (Qin et al., 2025)	50	99.2	96.2	0.26	96.7	99.9	2.12
Few-Step							
Set2GraphVAE (Vignac & Frossard, 2022)	1	59.9	93.8	–	–	–	–
MoFlow (Zang & Wang, 2020)	1	91.4	98.7	4.46	63.1	99.9	20.9
PairFlow (*) (Park et al., 2025)	1	44.3	**99.5**	–	11.2	**100.0**	–
PairFlow (*) (Park et al., 2025)	2	66.9	**99.4**	–	42.6	**100.0**	–
Naive Flow Map (Boffi et al., 2025)	1	84.6	93.0	2.31	59.5	99.9	21.7
Naive Flow Map (Boffi et al., 2025)	2	87.7	97.7	0.69	51.6	**100.0**	16.1
CFM CSD (ours)	1	90.0	91.8	**2.14**	73.5	99.9	**19.3**
CFM CSD (ours)	2	91.1	97.6	**0.49**	67.0	99.9	12.1
CFM ECLD (ours)	1	**95.8**	89.2	2.16	**93.5**	99.5	20.0
CFM ECLD (ours)	2	**97.0**	96.6	0.52	**82.4**	99.9	**12.2**

Table 1. Molecular generation results on QM9 and ZINC. Top shaded rows indicate multi-step reference methods; our goal is to match their quality with far fewer sampling steps. A dash indicates that the metric was not reported by the corresponding source. In each sub-category (and for each NFE for few-step models), we make bold the best result. (*) PairFlow percentages derived from $N = 1{,}024$ samples. MoFlow results from Jo et al. (2024).

Some experiments

Quick glance – LM1B

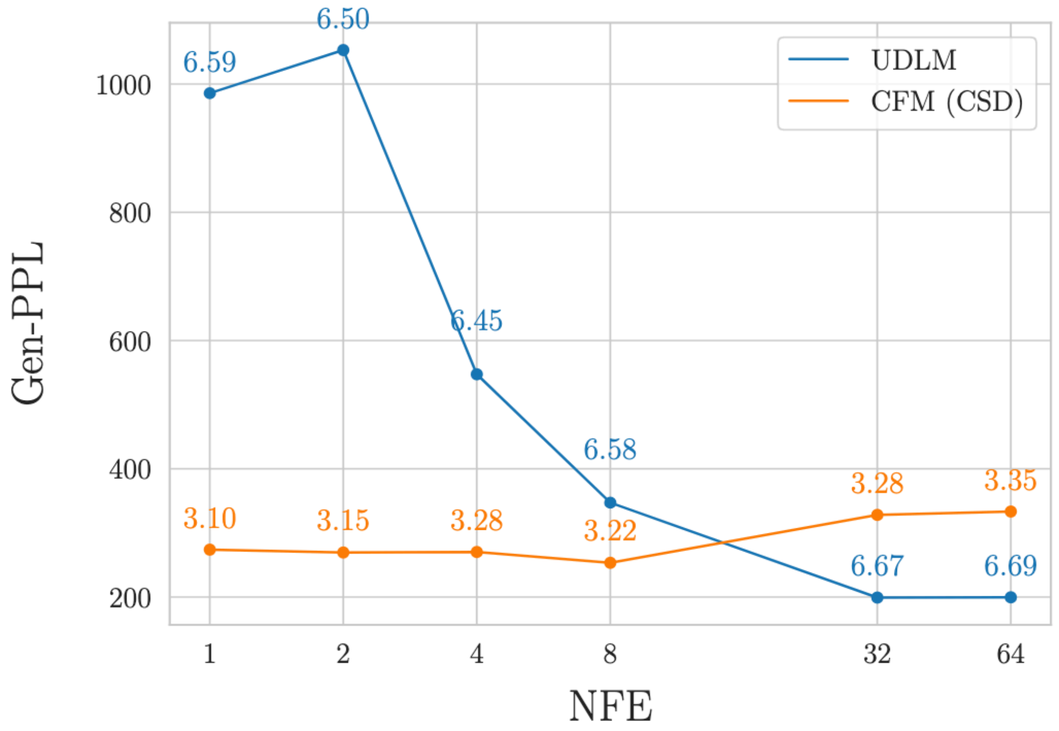


Figure 5. LM1B: NFE against Gen-PPL as measured by GPT-2. The numbers above are the corresponding entropies on the token space of the GPT model.

Acceleration enables steering 🎛️

Another exciting use case

- Note that our ODE,

$$\mathrm{d}X_t = v_t(X_t)\mathrm{d}t,$$

- has a stochastic counterpart, with the same marginals:

$$\mathrm{d}X_t = \left[v_t(X_t) + \frac{\sigma_t^2}{2}s_t(X_t)\right]\mathrm{d}t + \sigma_t \mathrm{d}W_t.$$

- The above process, we can steer towards a reward!

Acceleration enables steering 🎛️

Another exciting use case

- For a reward r, we want to sample from a distribution:

$$\tilde{p}(x) \propto p_1(x)\exp(r(x)).$$

- We can adjust it by setting the score to be

$$\tilde{s}_t(x) = s_t(x) + t\nabla_x r(X_{t,1}(x)).$$

- The algorithm was taken from *Test-time scaling of diffusions with flow maps* by Sabour et al. (2025).

Toy example of steering

Proof of concept only

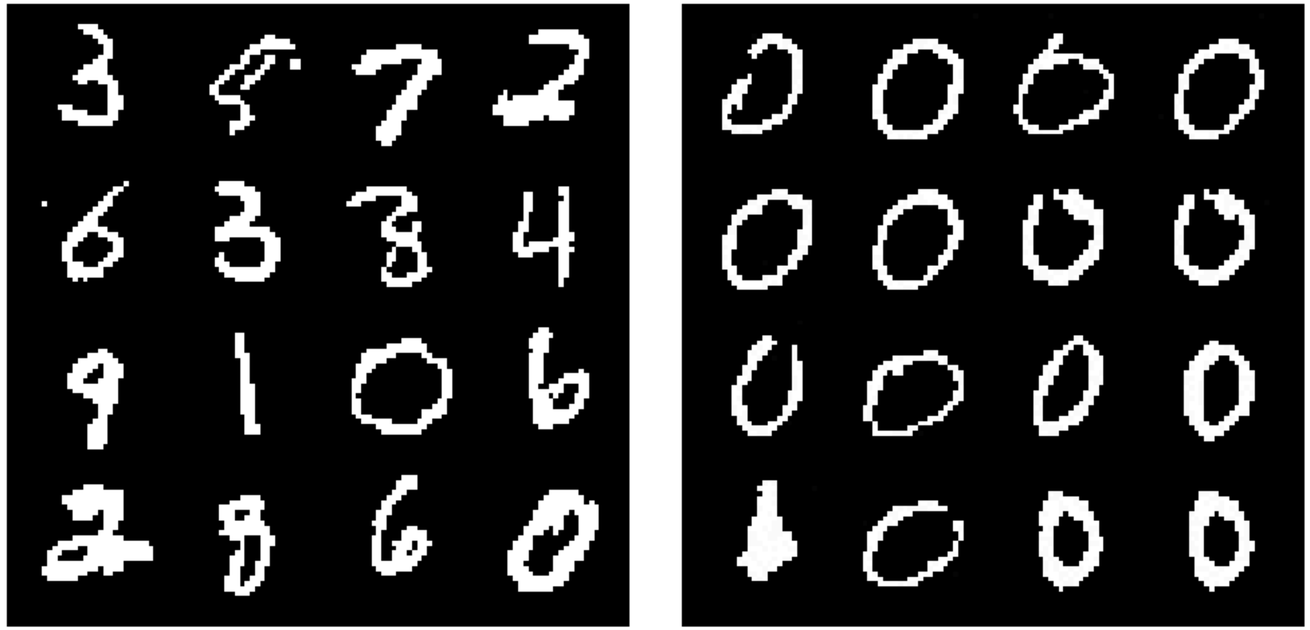


Figure 3. Samples from our categorical flow map on binary MNIST: (left) unconditional (right) and tilted by the classifier reward towards zeroes.

Categorical Flow Maps

Quick checkpoint

- We have shown that we can train one-step samplers.
- We have shown that we can use it to have better estimates of our endpoints.
- But the quality wasn’t there yet. Can we do better?
- Can we scale this up?

Scaling Categorical Flow Maps

Davis, Filippova, Ablin, Turrisi, Shidani, Cuturi, Béthune

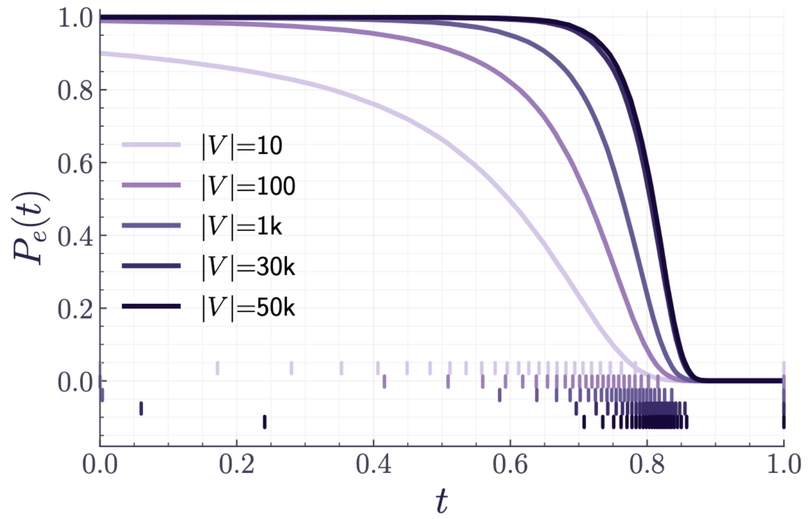
What was missing in the recipe? 🍳

Presentations of FMLM and DFM tricks

- Consider the decoding error rate (Pynadath et al., 2025; Sahoo et al. 2025):

$$P_e(t) := \mathbb{P}(\text{argmax } X_1 \neq \text{argmax } I_t).$$

- This quantity has a steep phase transition:



Lee et al. (2026)

Make uniform progress in the DER

Decoding error rate

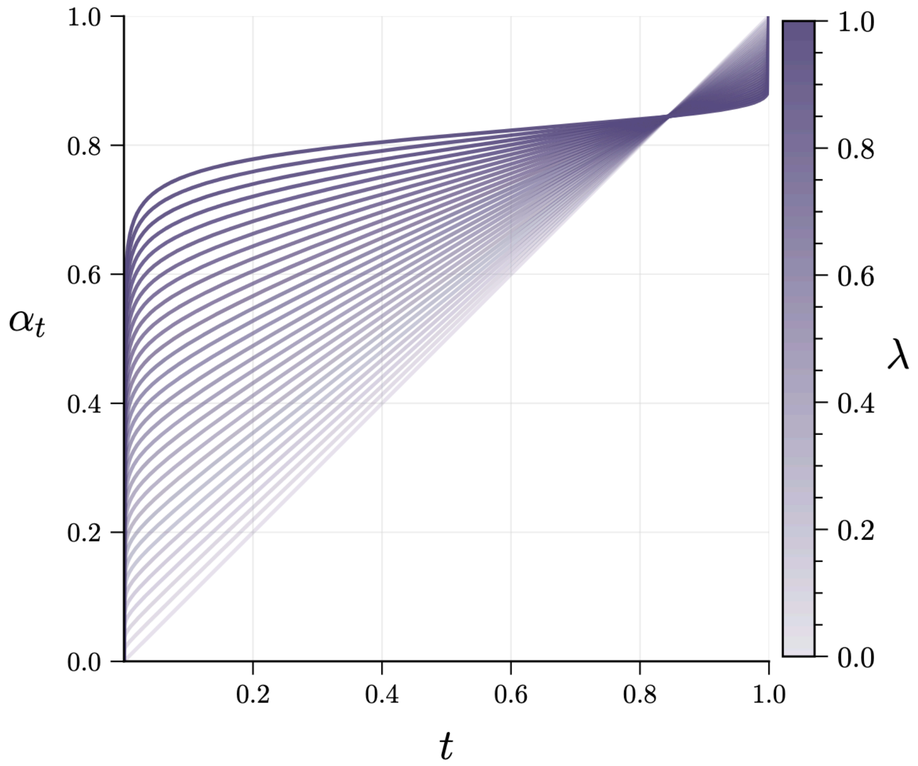


Figure 3 Mixture schedules for varying $\lambda \in [0, 1]$ at vocabulary size 100k.

Adaptive loss weight

From Geng et al. (2024)

- Given prediction $p(x)$, target $q(x)$,
- Let the error be $\|\Delta(x)\| = \|p(x) - q(x)\|^2$,
- Then multiply the loss by

$$w(x) = \mathrm{sg}\left((\Delta(x) + c)^{-r}\right).$$

Progressive self-distillation

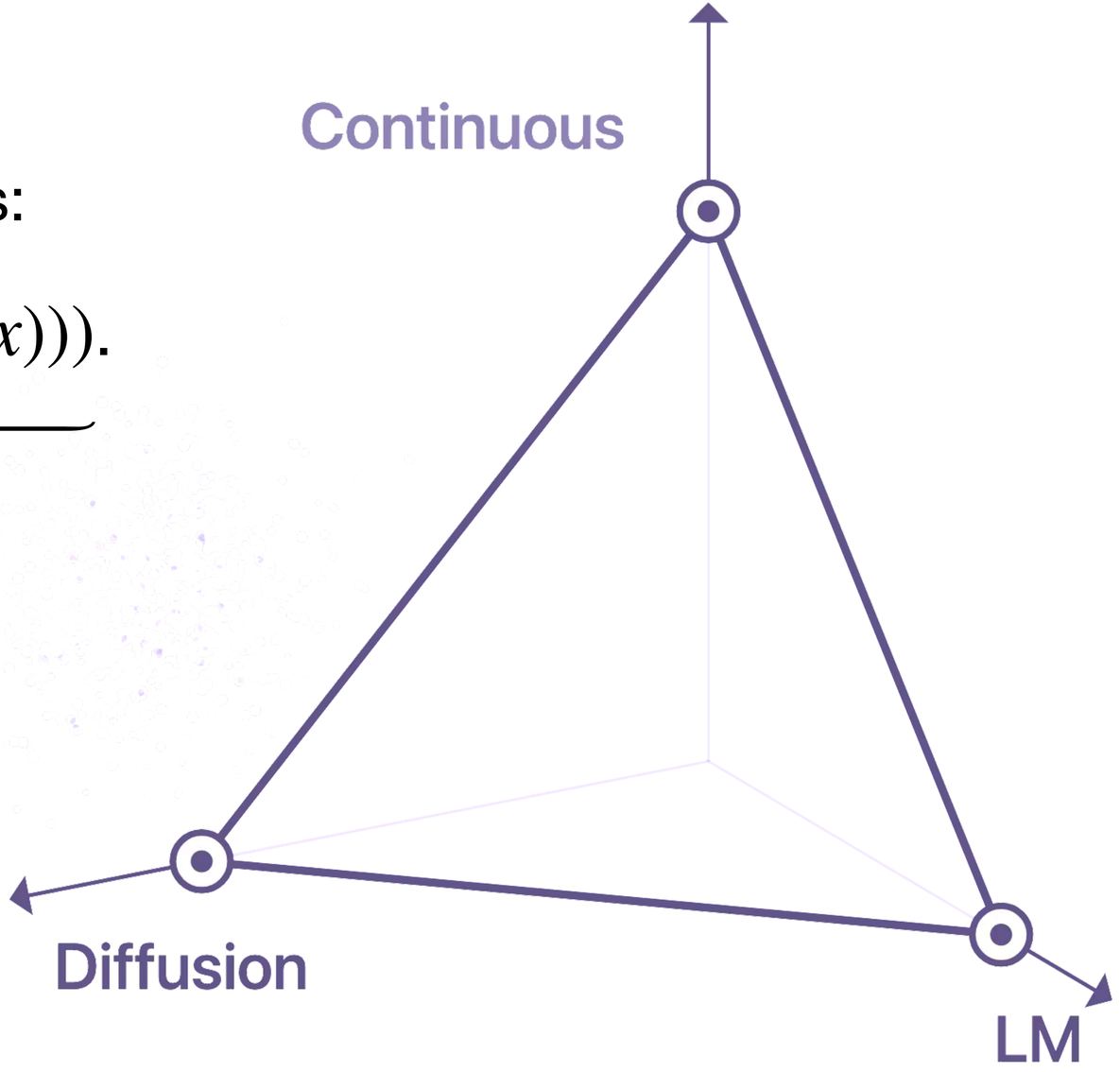
Works best, here

- For the endpoints, the identity becomes:

$$\pi^{\theta}_{s,t}(x) = \underbrace{(1-\alpha')\pi^{\theta}_{s,u}(x) + \alpha'\pi^{\theta}_{u,t}(X^{\theta}_{s,u}(x)))}_{:=T^{\theta}_{\text{psd}}(I_s)}.$$

- The loss therefore is:

$$\mathscr{L}_{\text{PSD}}(\theta) = \mathbb{E}\text{D}\left(\pi^{\theta}_{s,t}(I_s), T^{\theta}_{\text{psd}}(I_s)\right).$$



Sweeps at 400M 🧹

Extensive grid search

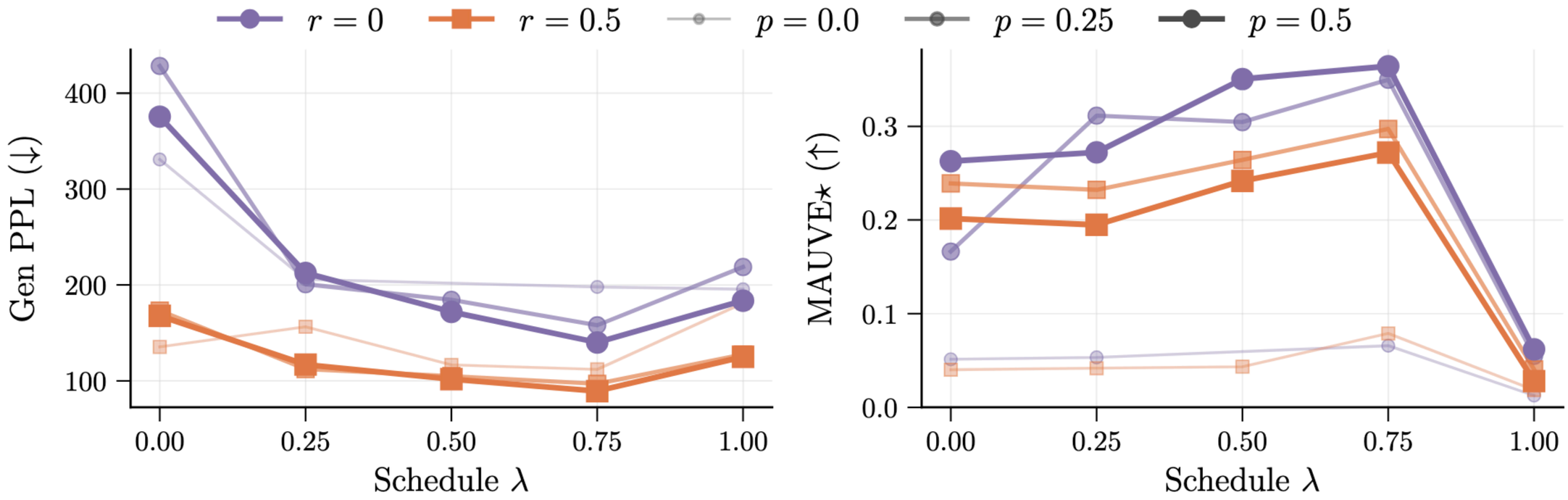


Figure 4 Ablation over schedule λ, adaptive loss reweighting r, and clean prefix probability p: Generative Perplexity and MAUVE⋆ score as a function of λ.

Sweeps at 400M 🧹 (ctd.)

Extensive grid search

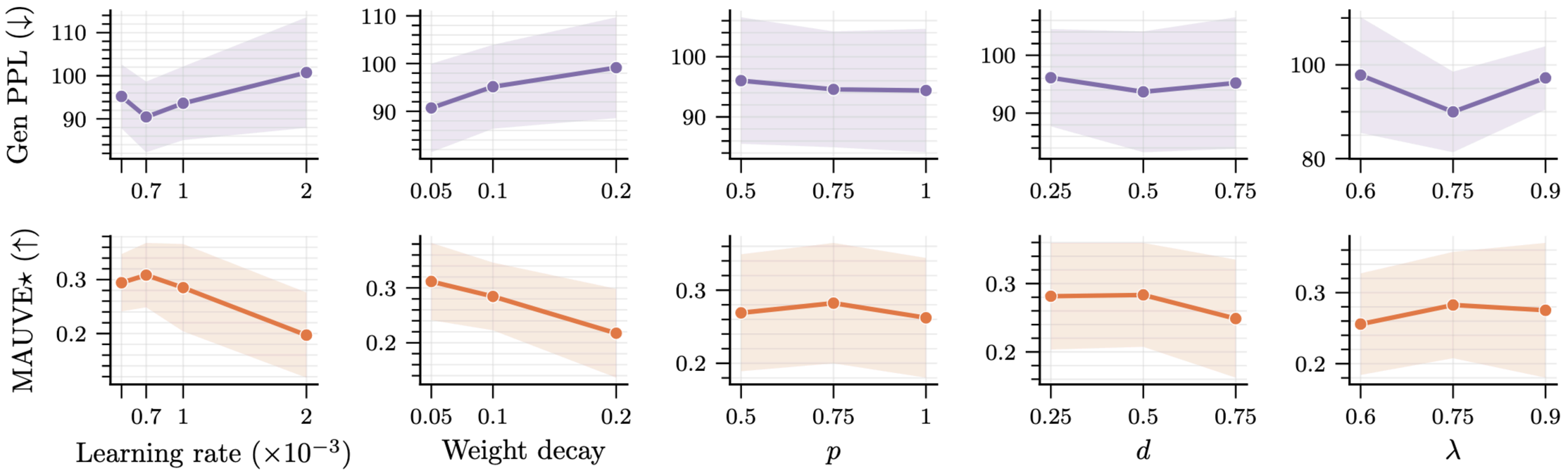


Figure 5 Marginal effect of each swept hyperparameter on Gen PPL (top) and MAUVE⋆ (bottom), averaged over all other variables (324 configurations). Shaded regions denote ±1 standard deviation. Learning rate and weight decay are the primary drivers; p, d, and λ are robust in the narrowed range.

Sweeps at 1.7B 🧹

For self-distillation

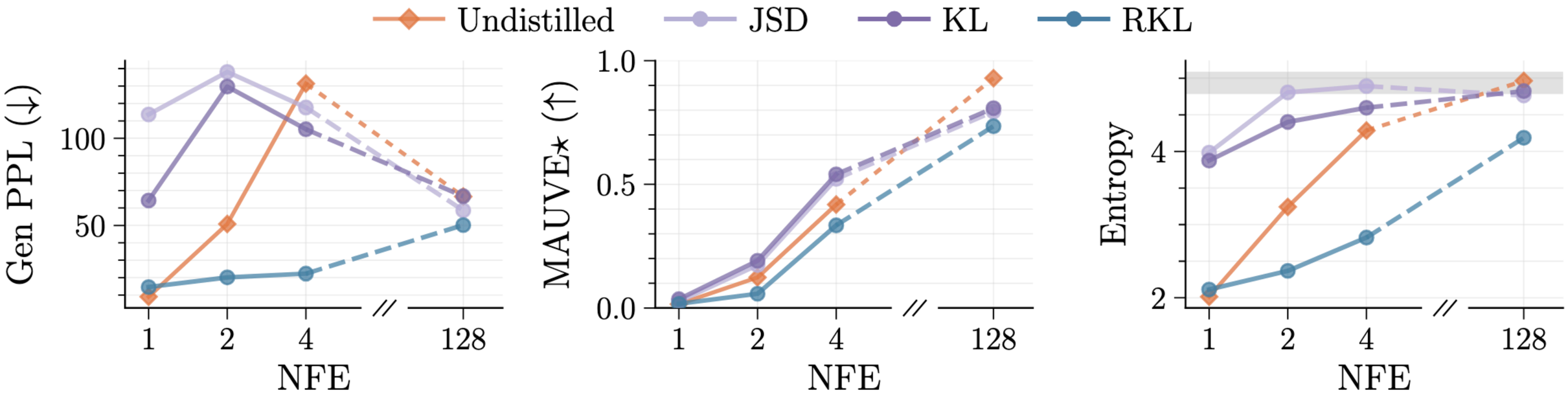


Figure 6 Ablation on PSD self-distillation, 10k steps: marginal effect of the divergence on MAUVE⋆ and Gen PPL, averaged over the time-sampling distribution.

Sweeps at 1.7B 🧹 (ctd.)

For self-distillation

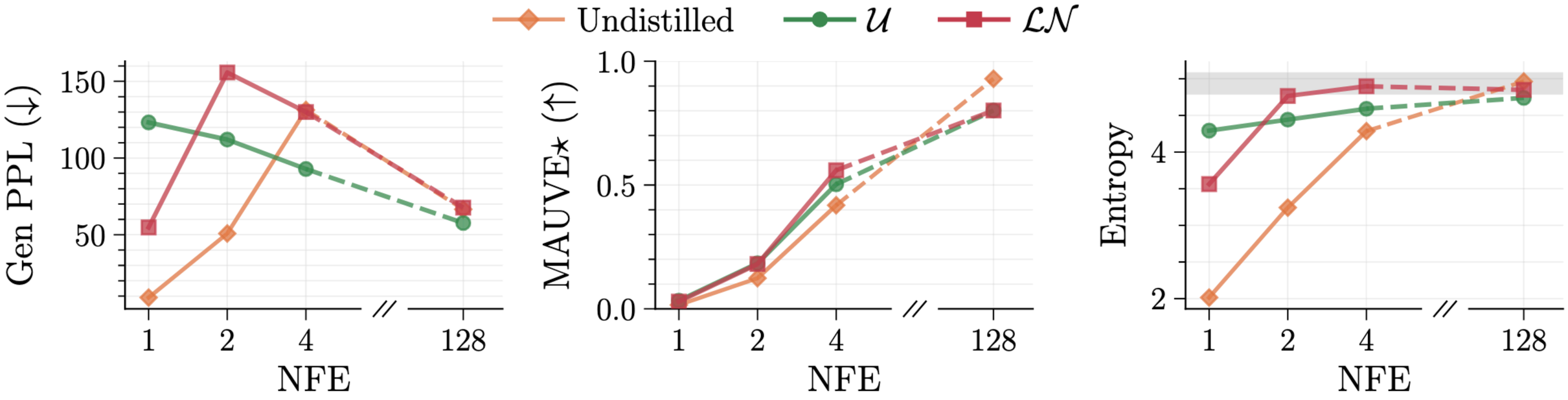


Figure 7 Ablation on PSD self-distillation, 10k steps: marginal effect of the (s, t) time-sampling distribution on MAUVE⋆ and Gen PPL, averaged over the divergence without the reverse KL data.

Generative PPL. & Entropy

Self-distillation works and is competitive!

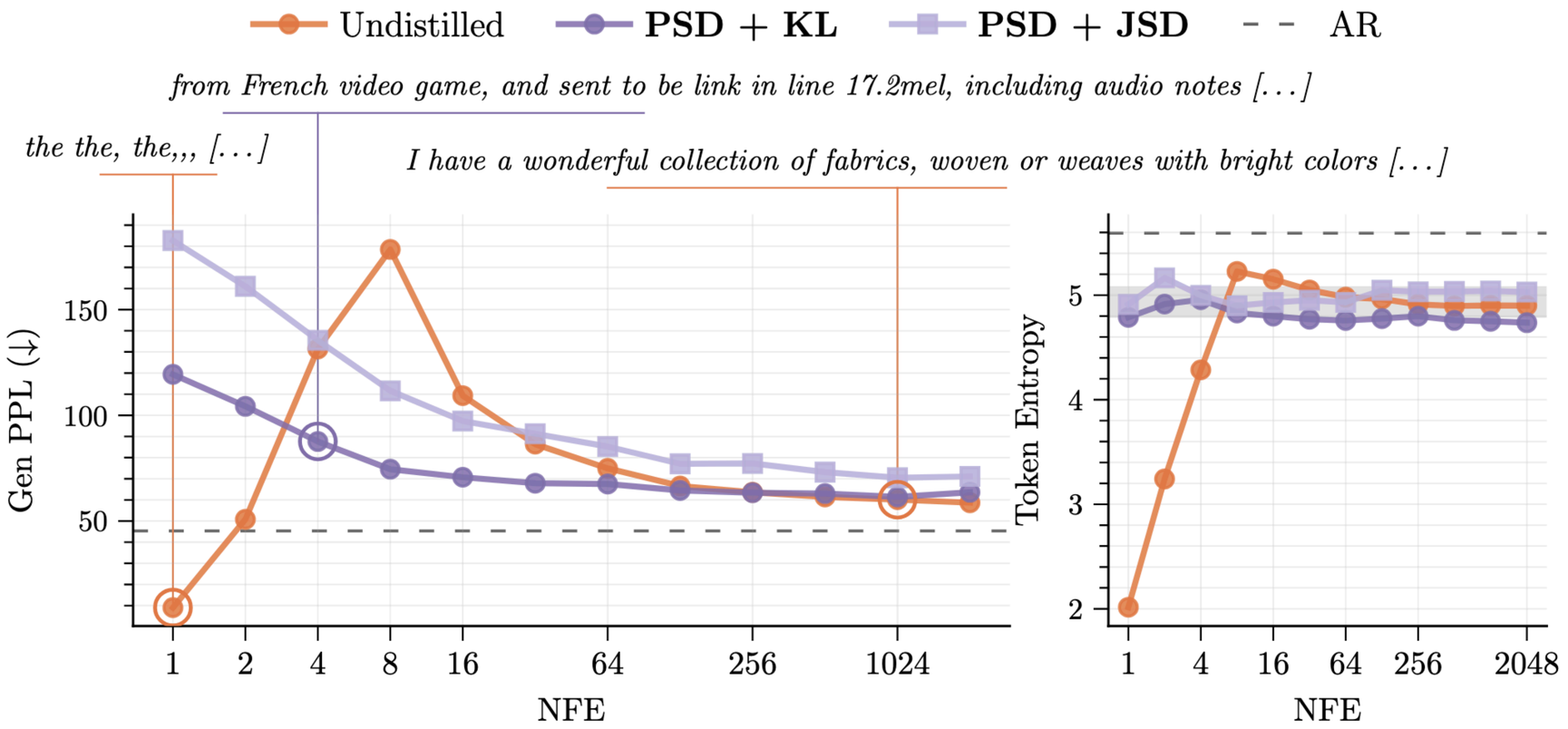


Figure 1 (Left) Gen PPL vs. NFE. (Right) Token entropy vs. NFE. The gray band indicates the token entropy of the data distribution. Above the circled points are provided short qualitative samples that reflect the generation quality at that NFE regime. Self-distillation enables high-quality, diverse generation at significantly fewer function evaluations.

Likelihood benchmarks

Not far from discrete diffusion

Table 1 Zero-shot accuracy (%) on likelihood-based benchmarks. †Models from Sahoo et al. (2026a) (1.7B params, same data and compute budget). *External prior work. CFM likelihood is estimated via MC ELBO. Best non-AR results per column are in **bold**.

	Method	ARC-E	BoolQ	OBQA	PIQA	RACE	SIQA
	Chance	24.7	50.4	26.6	51.6	24.2	32.2
Prior work	SMDM-1B*	37.4	61.5	27.0	60.3	29.3	37.9
Prior work	LLaDa-8B-Base* (Nie et al., 2025)	–	–	–	74.4	–	–
Prior work	AR† *(autoregressive)*	72.7	71.9	40.4	78.1	36.2	41.9
Prior work	AR *(autoregressive, ours)*	73.1	63.6	40.2	76.9	36.3	43.5
Prior work	Duo† *(uniform diffusion)*	**53.4**	59.6	**33.0**	**62.7**	**35.0**	39.0
Prior work	MDLM† *(masked diffusion)*	50.5	**62.8**	32.0	62.2	34.7	**39.2**
Prior work	Eso-LM† *(interpolating)*	46.0	53.4	29.6	55.6	26.1	36.1
	CFM *(ours)*	47.1	42.5	24.4	57.2	28.3	37.9

Many more details

For practitioners

- To be found in the appendix!
- Extensive qualitative samples.

Which new “recipes” work best? Why?

On the Design Space of CDLMs

Davis, Roos, Bizeul, Shidani, Turrisi, Béthune, Ablin, Cuturi

Goals 🎯

of the work

- For NeurIPS, tens of works had come out already – what's working?
- New odd "artefacts" (never needed in continuous):
  - Self-conditioning;
  - Stochastic sampling…
- And one-hot can't reason!

TinyGSM as a benchmark

Less hackable than Gen. PPL.

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

Self-conditioning

Quick reminder

- Introduced for language in Chen et al. (2022; Analog bits).
- Take a forward pass of your model on the input:

$$\hat{X}_1 = \mathrm{sg}\left(\pi_t^{\theta}(I_t, 0)\right).$$

- Train with that as an extra input (25% of the time):

$$\mathscr{L}_{\mathsf{diag}}(\theta) = -\,\mathbb{E}\sum_{\ell=1}^{L} \log\langle \pi_t^{\theta,\ell}(I_t, \hat{X}_1), X_1^{\ell}\rangle.$$

Trajectory ambiguity ?

Intuition

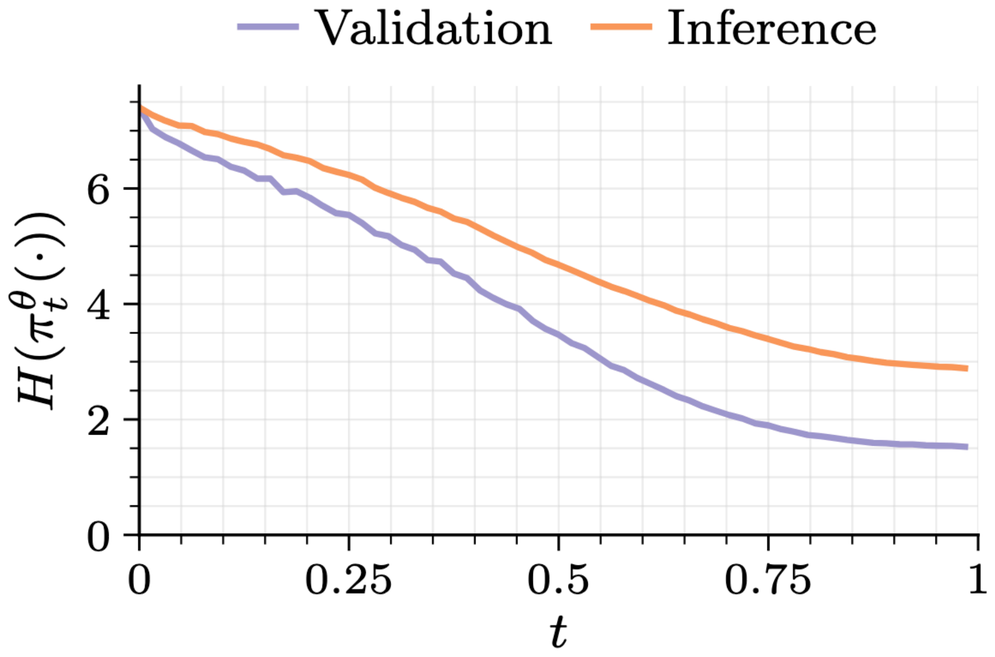
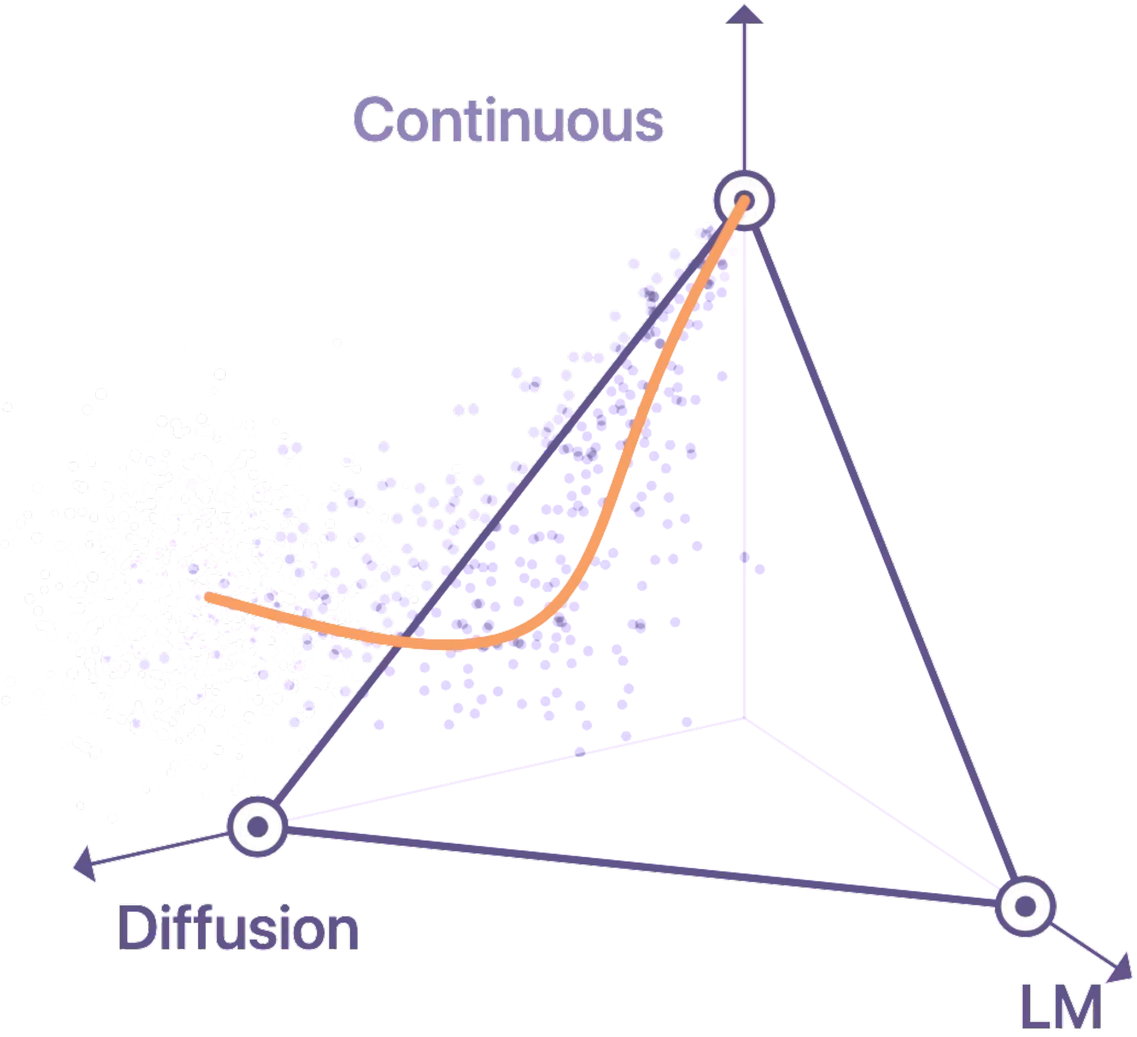


Figure 2 Entropy of a one-hot flow's denoiser, π^θ, along validation set and inference trajectories.



Trajectory ambiguity ?

Intuition

- Your inference path looks like:

$$\hat{I}_t = (1 - \gamma_t)X_0 + \gamma_t \hat{Y}_t.$$

- Your training follows only “clean” paths!

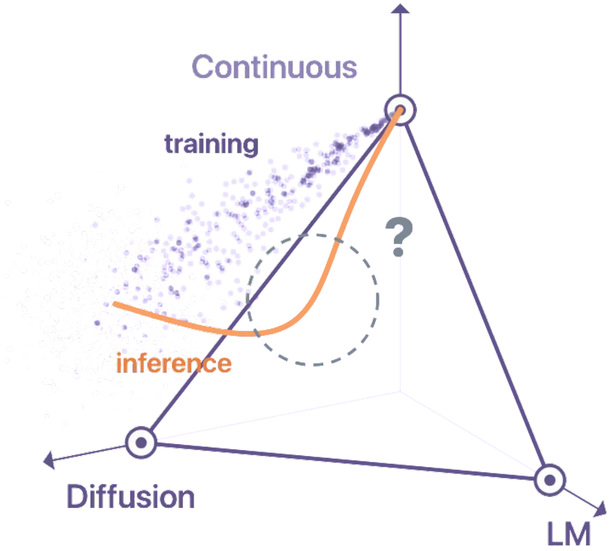
DoubleCat 🐈

Two-level interplant for additional ambiguity

$$\begin{cases} Y_t^\ell &= (1-\kappa_t)Y_0^\ell + \kappa_t X_1^\ell \\ X_t^\ell &= (1-\gamma_t)X_0^\ell + \gamma_t Y_t^\ell \end{cases}$$

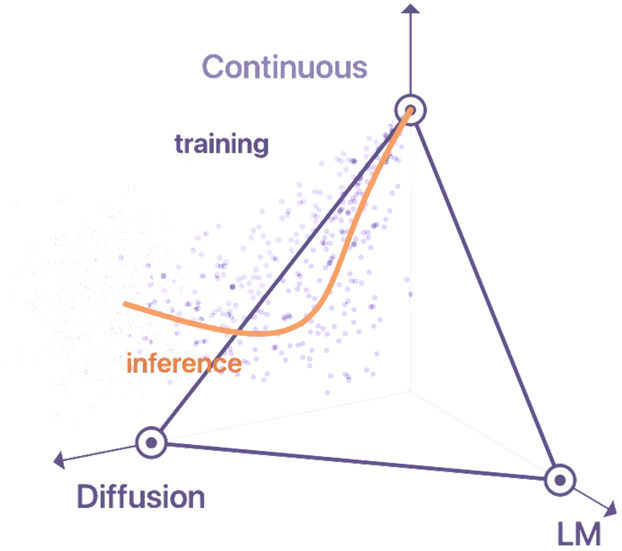
One-hot flow

inference goes where training rarely does



DoubleCat

training bends like inference



DoubleCat 🐈

Two-level interplant for additional ambiguity

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**

Why does it matter so much? 🧐

When it didn’t for continuous (tentative)

- Let the true denoiser be $D_t^\star(x) = \mathbb{E}[X_1 \mid I_t = x]$.
- The trace of its Jacobian is $\frac{\gamma_t}{(1-\gamma_t)^2}(1 - \|D_t^{\star,\ell}(x)\|^2)$:
  - In uncertain regions, this blows up; but those are rarely trained upon.
- This quantity remains stable for regular enough continuous densities.

Can we do any better?

Making SemiCat

- Tokens are hard to condition on for such large Gaussians (CANDI; Pynadath et al. 2026).
- Let’s introduce discrete reveals to fix this!

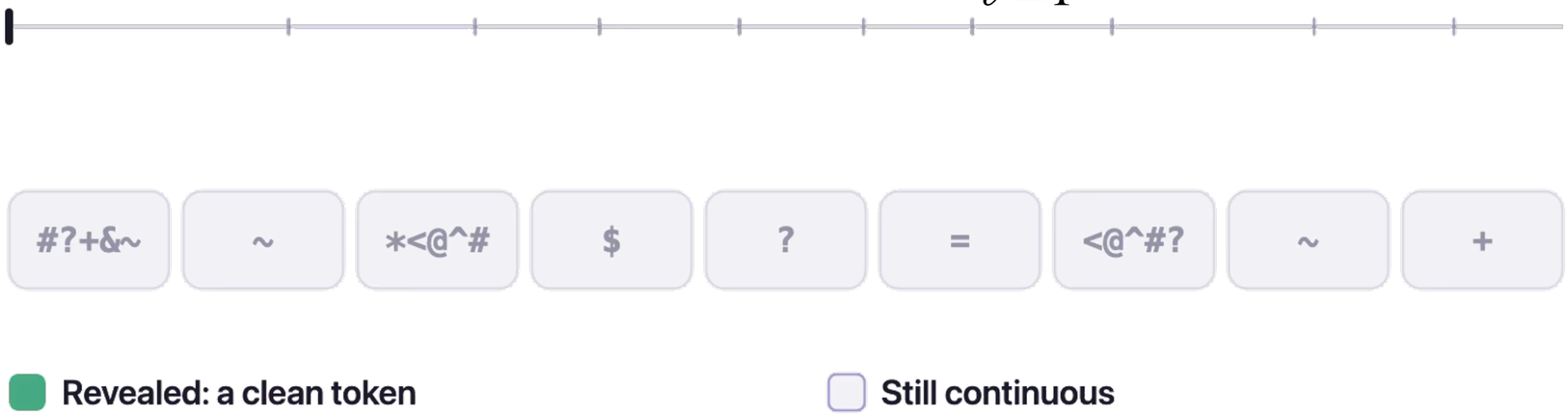
SemiCat

CANDI + DoubleCat (masking/revealing)

- The interpolant now reads, $\tau^\ell \sim \mathcal{U}(0,1)$ iid,

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

- Train on unrevealed positions $\mathscr{L}_{\mathrm{sc}}(\theta) = -\,\mathbb{E} \sum_{\ell=1}^{L} \mathbf{1}(t < \tau^\ell) \log \langle X_1^\ell, \pi_t^{\theta,\ell}(I_t^{\mathrm{sc}}) \rangle.$



SemiCat

Discrete diffusion-level, 50% *without self-conditioning*

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
with standard hyperparameters†	21.08	35.41	37.68	38.67	90.75	9.10
+ InfoNoise (Raya et al., 2026)	**21.46**	**36.24**	41.09	38.06	86.58	13.04
+ Algorithm 2 ($\rho/N = 0.75$, $\lambda = 0.25$, $\eta_{\max} = 0.25$)	–	–	**49.73**	**49.94**	**95.36**	**4.38**

Further reading 📋

- Why Gaussian Diffusion Models Fail on Discrete Data and How to Prevent It? by Shabalin et al. (2026).

How to design embeddings? 

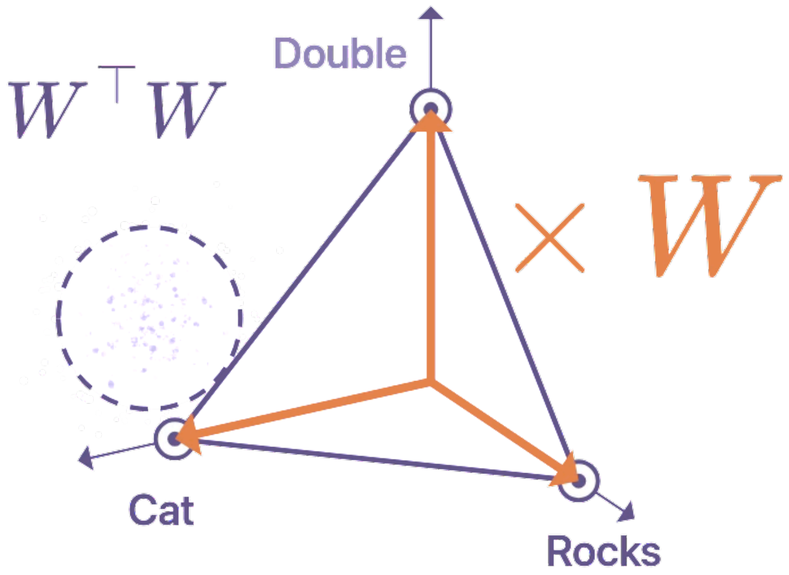
How different is one-hot and embedded?

When done naively

- One-hot flows are really embedded: first layer is $W \in \mathbb{R}^{|V| \times d}$.
- So inputs are distributed as

$$(I_t W)^\ell \sim \mathcal{N}(\gamma_t (X_1^\ell)^\top W, (1-\gamma_t)^2 W^\top W).$$

- Every token embedding is updated at every step!



Embedded flows are one-hot

If trained with cross-entropy

- Let a simple unit-normalised embedded flow be defined by

$$I_t^\ell(X_0, X_1) = (1 - \gamma_t)X_0^\ell + \gamma_t \mathsf{emb}(X_1^\ell).$$

- Train it with cross-entropy, as before.

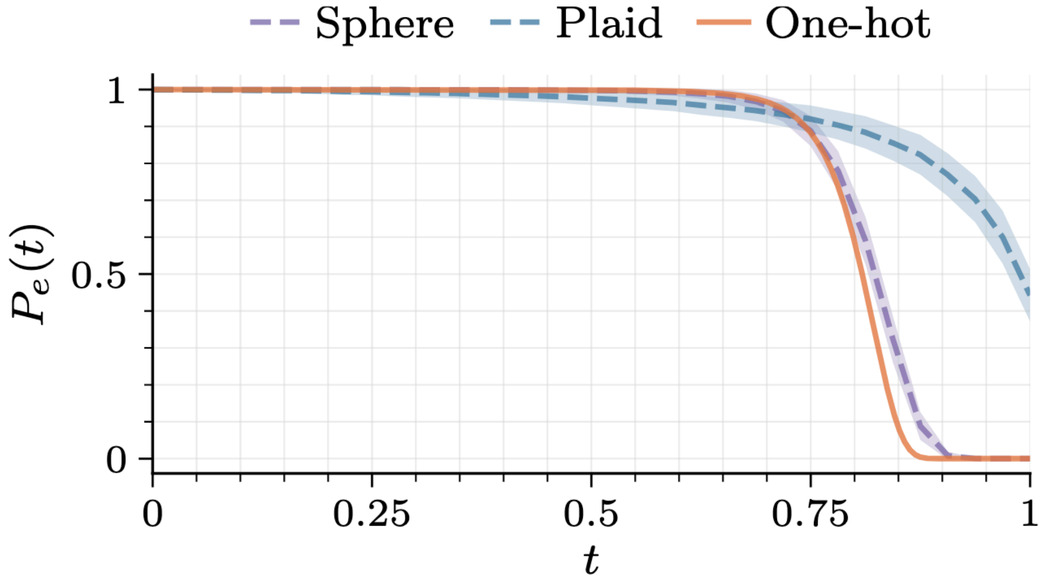
Embedded flows are one-hot

If trained with cross-entropy

- Cross-entropy spreads embeddings maximally apart.
- Therefore the generalised decoding error rate,

$$P_e(t) = \mathbb{P}\left(\operatorname{argmin}_{v \in V} \| e_v - I_t^\ell \|^2 \neq \operatorname{argmax}\ X_1^\ell\right),$$

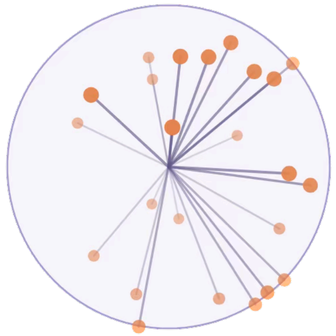
- behaves like that of a one-hot flow!

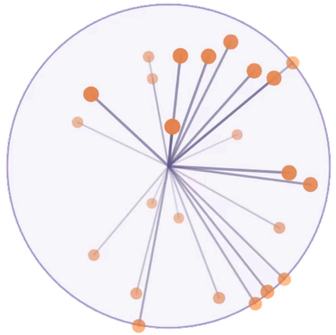


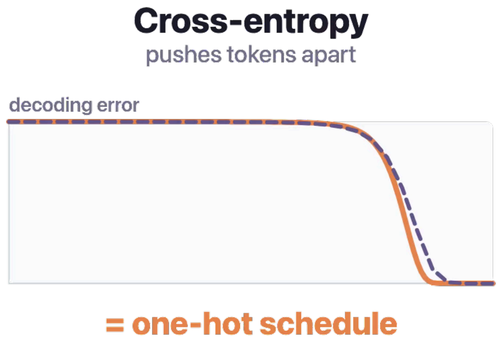
Summary

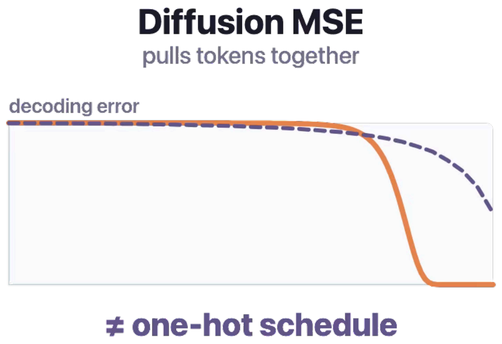
On one-hot vs embedded

- The one-hot(-like) DER requires better schedules.
- Embedded has the advantage of being less consuming memory-wise.





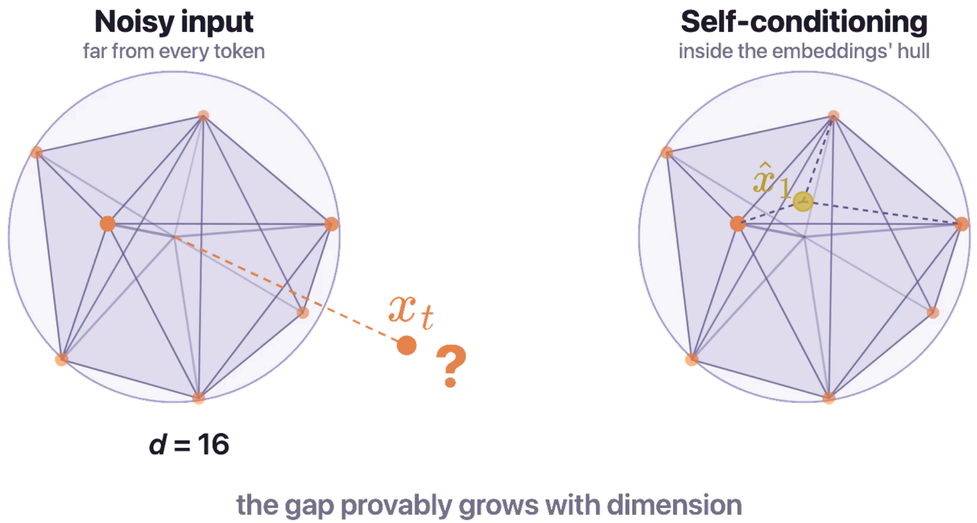




Self-conditioning, again

In embedded spaces

- Self-conditioning helps *also* identify the current token!
- It is a better representation of the current point:
  - It is always on the convex hull.



Verifying at scale

SemiCat, 1.7B, 2.1T tokens

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 equations (2.2) and (3.7)). Best non-AR results at 1.7B parameters per column are in **bold**.

Gen PPL/Entropy Pareto fronts

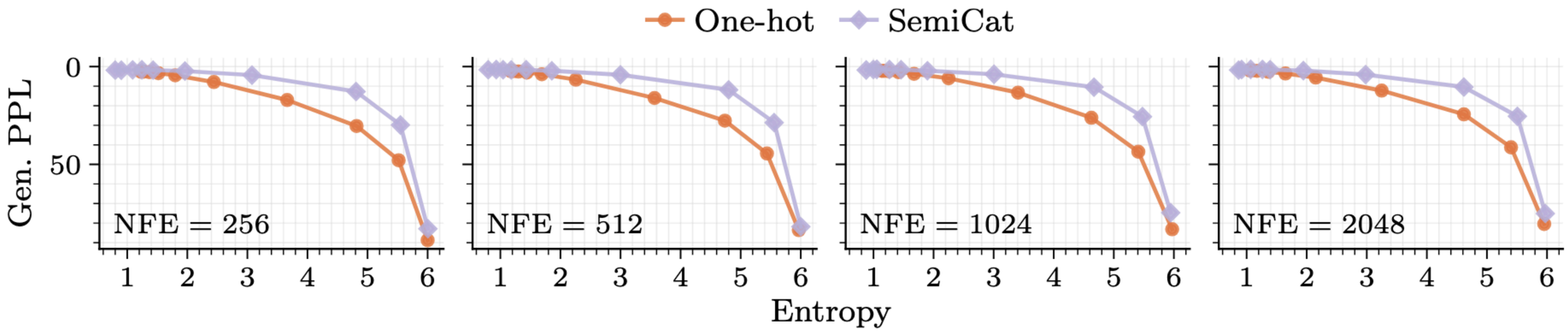


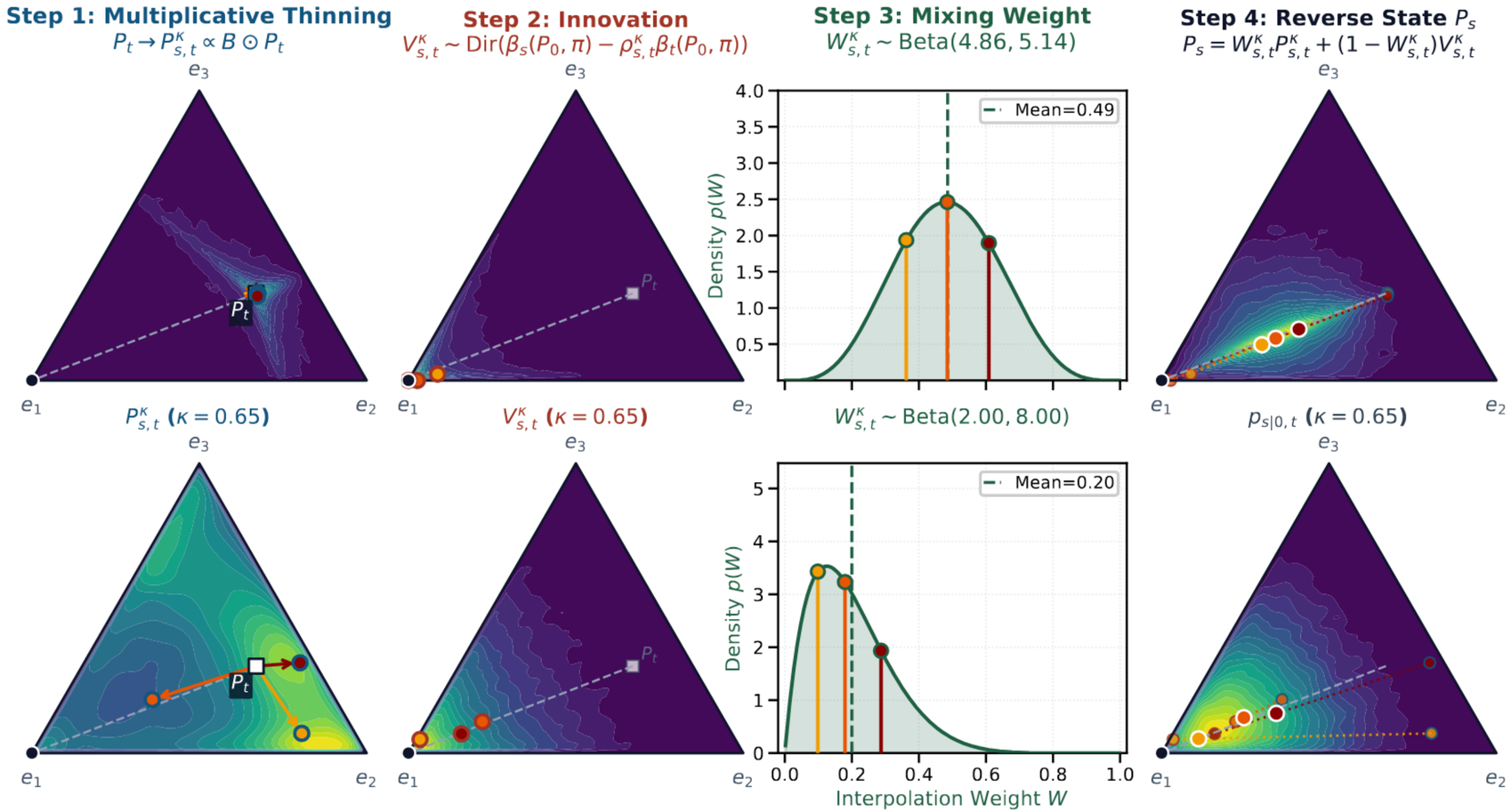
Figure 5 Gen. PPL as measured by Qwen 2.5 7B for models trained on Nemotron with sequence lengths of 2048 tokens, using the TikToken tokeniser.

Summary

- Self-conditioning adds ambiguity and identifiability;
  - We need better representations!
- Embedded and one-hot are closer than what you think;
- This is *not* a recommendation to train SemiCat models.
- There are lot of other techniques attempted in the paper.

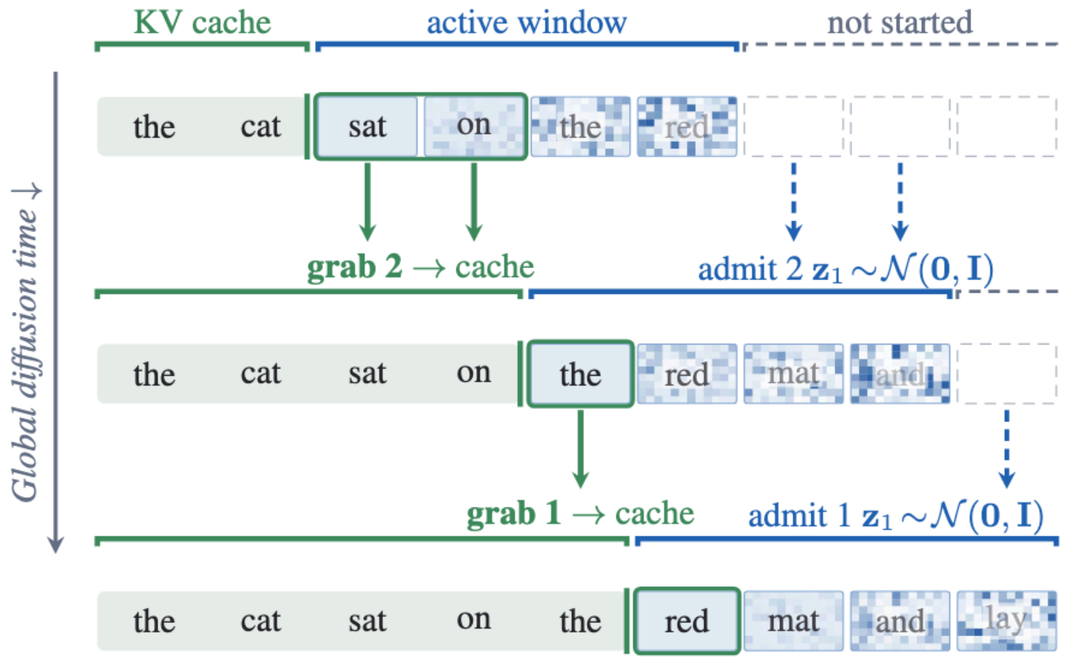
Note on Simplex Diffusion Models 🔺

Deschanaux et al. (2026). No self-conditioning.



Note on Clock Diffusion ⏰

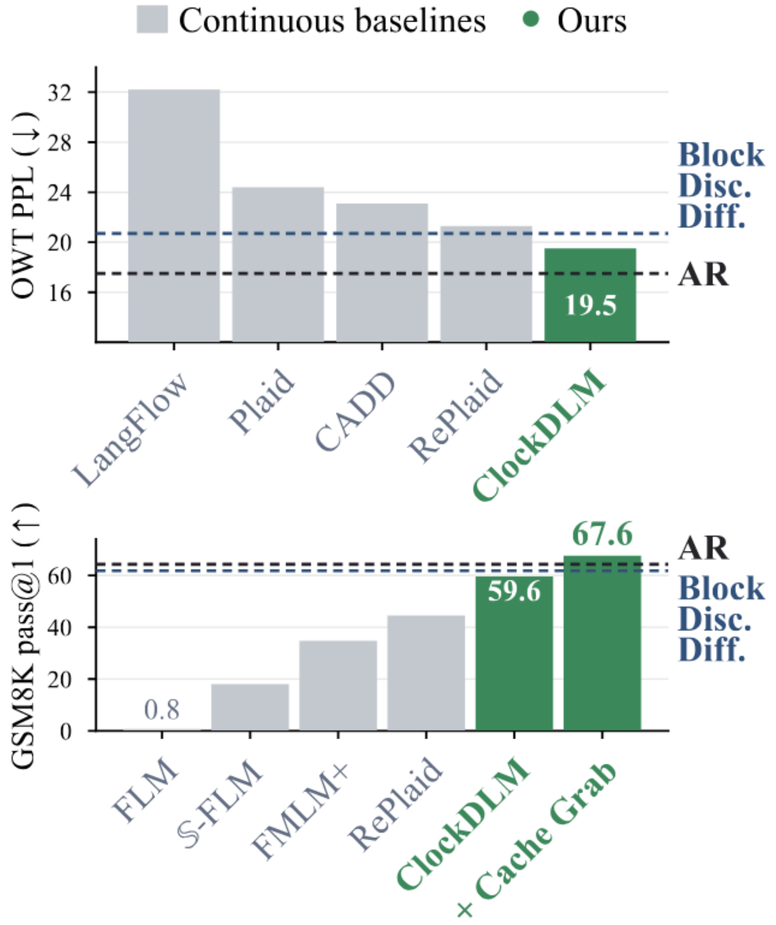
Schiff et al. (2026). Continuous relaxation of reveals.



Clock Diffusion: Continuous language models with

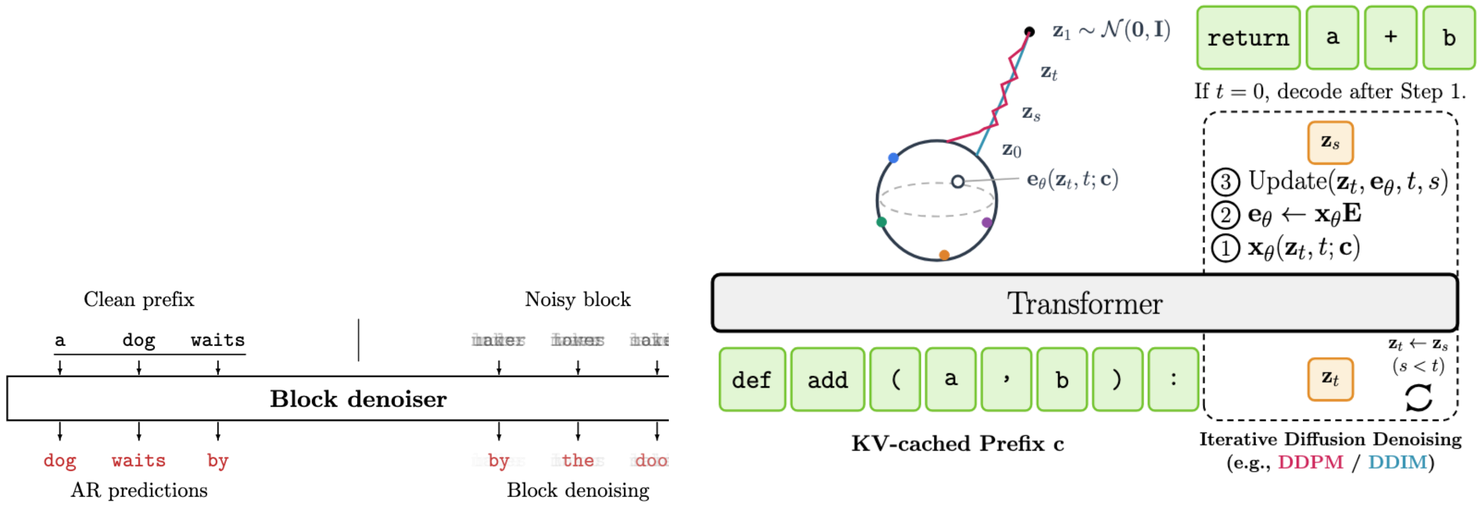
✓ **Arbitrary length** ✓ **Accelerated inference**

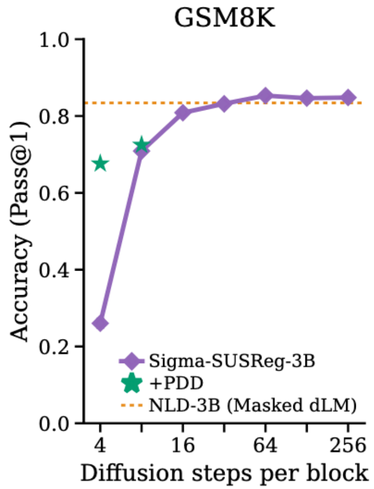
✓ **KV cache** ✓ **SoTA diffusion perplexity**

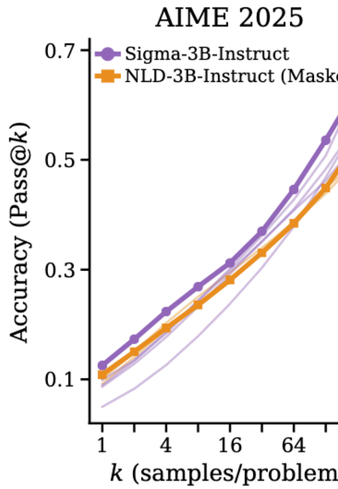


Exciting things happening!

Large Flow Language Models (Chavan et al., 2026)
Large Language Continuous Diffusion Models (Yang et al., 2026)







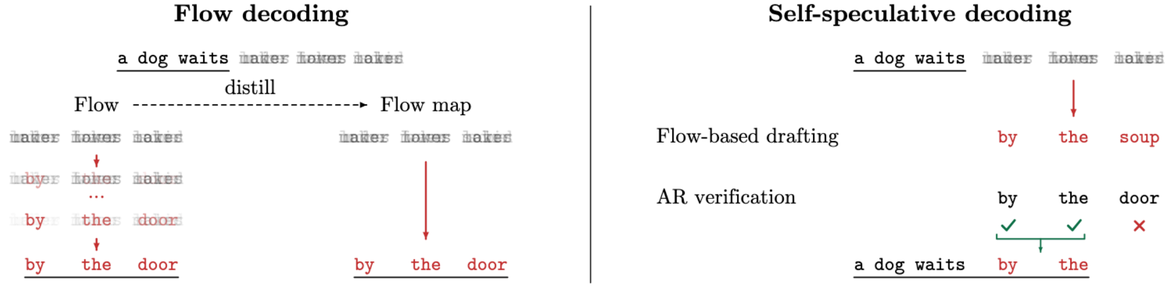


Figure 2: Overview. (Top) Lafa learns a block denoiser that also performs AR prediction on the prefix. (Bottom) Lafa supports flow decoding, which can be further distilled into a flow map for one-step generation. Together, AR and flow support self-speculative decoding, which accepts flow-based drafts from left to right via AR verification.

A lot more to come

- Better acceleration methods;
- Better steering/tilting/guidance;
- Better adaptation.

Thank you for listening! 🙏

- A special thanks to all my collaborators:
  - Academics: Ismail Ceylan, Michael Bronstein, Luca Ambrogioni, Floor Eijkelboom, Daan Roos, JW van de Meent.
  - Apple MLR: Marco Cuturi, Louis Béthune, Pierre Ablin, Alice Bizeul, Amitis Shidani, Anastasiia Filippova, Victor Turrisi.

Contact 🤝

Get in touch!



- Website: https://olsdavis.github.io/
- X: https://x.com/osclsd
- Email: oscar.davis@cs.ox.ac.uk