LOGIT-KL FLOW MATCHING: NON-AUTOREGRESSIVE TEXT GENERATION VIA SAMPLING-HYBRID INFERENCE

Anonymous authors

Paper under double-blind review

ABSTRACT

Non-autoregressive (NAR) language models offer notable efficiency in text generation by circumventing the sequential bottleneck of autoregressive decoding. However, accurately modeling dependencies in discrete sequences remains challenging in this paradigm. In this work, we advance the field of NAR generation by applying conditional flow matching (CFM) methods grounded in geometrically principled interpolation, specifically leveraging Kullback-Leibler (KL) divergence geodesics, which correspond to linear interpolation in logit space. We rigorously establish that maximizing conditional likelihood in this setting precisely recovers the flow matching velocity field, supplying the theoretical justification for this approach in sequence modeling. To address practical performance gaps of *basic* inference, we propose a novel empirical *sampling* strategy that iteratively denoises and re-noises, along with a *hybrid* scheme that integrates our *sampling* method with *basic* procedure. Across unconditional and conditional text and code infilling, the approach improves perplexity and downstream metrics over prior NAR baselines under matched settings.

1 Introduction

Non-autoregressive (NAR) language models have emerged as efficient alternatives to traditional autoregressive models in NLP by generating all tokens simultaneously. However, capturing complex dependencies in discrete textual data remains challenging without sequential modeling.

We investigate conditional flow matching (CFM) methods for text generation, building on recent advances such as Discrete Flow Matching (DFM) Gat et al. (2024), Dirichlet Flow Matching Stärk et al. (2024), and Fisher-Flow Davis et al. (2024), which represent tokens as one-hot vectors in a V-1-dimensional simplex. These methods interpolate a sequence of distributions ρ_t from an initial ρ_0 to a data distribution ρ_1 ; for text, the latter is sampled as discrete sequences in the simplex. Prior work identifies issues with naive linear interpolation in simplex space Stärk et al. (2024). We propose instead using KL-geodesics, equivalent to linear interpolation in logit space, to better capture the underlying geometry.

Our CFM framework leverages this interpolation, training with a denoiser maximizing the conditional likelihood $p_{\theta}(x_1 \mid x_t)$, enabling tractable approximation of the joint distribution. While theoretical guarantees previously existed only for single-token predictions, we show that maximizing this conditional likelihood still exactly recovers the flow matching velocity field in logit space for sequence modeling, lending theoretical support to our approach.

Standard inference procedures with this framework yield suboptimal results, so we introduce a novel sampling strategy: given a state x_t , we sample x_1 from $p(x_1 \mid x_t)$ and re-noise it to x_{t+h} , iterating this process. Despite the lack of full theoretical analysis, this method yields stronger empirical results. We further propose a hybrid inference scheme blending our basic and sampling strategies, yielding improved performance on tasks such as text generation, conditional question answering, and code infilling (see Figure 1).

Our contributions are:

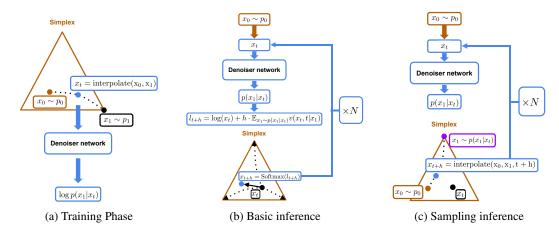


Figure 1: Overview of the Proposed Approach: Training: Sample $x_0 \sim p_0$ (uniform distribution on simplex), $x_1 \sim p_1$ (target distribution represented by samples); interpolate to obtain x_t . The denoiser network predicts $\log p_{\theta}(x_1|x_t)$, trained via log-probability maximization. Inference: For basic inference, numerically solve an ODE with vector field: $\mathbb{E}_{x_1 \sim p_{\theta}(x_1|x_t)}[v(x_t, t \mid x_1)]$ using Euler method with N steps and a step size of h = 1/N. Alternatively, in sampling inference, interpolate between $x_0 \sim p_0$ and $x_1 \sim p(x_1|x_t)$ at each step.

- Using KL-geodesic (logit-space linear) interpolation for flow matching in discrete sequences.
- Theoretical analysis showing conditional likelihood maximization exactly recovers the flow matching velocity field for logit-space interpolation.
- A novel sampling and hybrid inference strategy with strong empirical results.
- Empirical improvements: at least 27% lower perplexity for unconditional generation (Fine-Fine-Web), and at least 17%, 26% BLEU boosts for conditional tasks (Lamini Instruction, WMT 14 de-en); plus 56% and 14% gains in Pass@1 and Pass@10 for code infilling where 10% of the code lines were omitted. Prior methods are trained and evaluated under the same setup to ensure a fair comparison.

2 BACKGROUND

Flow matching Lipman et al. (2023) constructs a deterministic transport from a simple *base* distribution ρ_0 (e.g., $\mathcal{N}(0,I)$) to an unknown *data* distribution ρ_1 given by samples. It introduces a time-dependent density $\rho(x,t)$ and velocity field v(x,t) for $t\in[0,1]$ that satisfy the mass-conservation (Liouville) equation

$$\partial_t \rho(x,t) = -\nabla \cdot (\rho(x,t) v(x,t)), \tag{1}$$

with boundary conditions $\rho(\cdot,0)=\rho_0$ and $\rho(\cdot,1)=\rho_1$. Once v is known, samples are generated by integrating the characteristic ODE

$$\frac{dx_t}{dt} = v(x_t, t), \qquad x_{t=0} \sim \rho_0, \tag{2}$$

and taking $x_{t=1}$ as a draw from ρ_1 .

Learning the velocity field (conditional flow matching). Because v is unknown, it is approximated with a neural network $v_{\theta}(x,t)$ using interpolation between initial and target data samples. Let $\gamma_t(x_0,x_1)$ be any interpolation with $\gamma_0=x_0$ and $\gamma_1=x_1$; draw $t\sim \mathcal{U}[0,1], x_0\sim \rho_0$, and $x_1\sim \rho_1$, and set $x_t=\gamma_t(x_0,x_1)$. Define the vector field $v(x_t,t\,|\,x_0,x_1)\coloneqq\frac{d}{dt}\gamma_t(x_0,x_1)$. The conditional flow matching objective is

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,x_0,x_1} \Big[\| v_{\theta}(\gamma_t(x_0,x_1),t) - v(\gamma_t(x_0,x_1),t \,|\, x_0,x_1) \|_2^2 \Big]. \tag{3}$$

At inference, we integrate the ODE with v_{θ} from t=0 to t=1 to obtain samples.

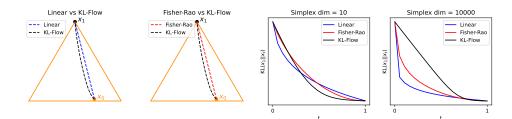


Figure 2: Qualitative and quantitative comparison of three distinct classes of geodesics on the probability simplex: Linear, Fisher–Rao, and KL–Flow. The first two panels juxtapose the trajectories of the Linear and Fisher–Rao interpolations against the KL–Flow interpolation. The rightmost panels depict the temporal evolution of the divergence $\mathrm{KL}(x_1\|x_t)$ for two different simplex dimensions, $|\mathcal{V}|=10$ and $|\mathcal{V}|=10\,000$.

Table 1: Perplexity (lower is better) obtained by 150M-parameter language models trained on the *FineFineWeb* corpus under the Linear, Fisher-Rao, KL-Flow geodesics.

Geodesic	Llama 2	GPT 3	GPT 2
Linear	1344	15418	13881
Fisher-Rao	192	298	379
KL-Flow	41	53	62

3 CONDITIONAL FLOW MATCHING FOR DISCRETE SEQUENCES

In language modelling, the terminal random variable x_1 is a one-hot vector (a vertex of the (V-1)-simplex). Following Stärk et al. (2024), we take the initial distribution ρ_0 to be the uniform (Dirichlet(1)) measure on the simplex, so $x_0 \sim \rho_0$ is a strictly positive probabilistic token mixture. A central design choice in flow matching is the *interpolation* between x_0 and x_1 .

KL geodesic on the simplex. While linear interpolation in probability space is possible, its drawbacks for discrete data have been documented (Stärk et al., 2024); Fisher–Rao geodesics have also been proposed (Davis et al., 2024). We instead use the geodesic induced by the Kullback–Leibler (KL) divergence—the canonical information–theoretic discrepancy on the simplex.

Definition 3.1 (KL geodesic). For $t \in [0, 1]$, the KL-geodesic joining x_0 and x_1 is

$$x_t = \frac{x_0^{1-t} x_1^t}{\sum_{i=1}^V x_{0,i}^{1-t} x_{1,i}^t} \equiv C_t x_0^{1-t} x_1^t, \tag{4}$$

where C_t normalizes x_t onto the simplex.

It is linear in logits, $l_t = (1-t)\log x_0 + t\log x_1$ with $x_t = \operatorname{Softmax}(l_t)$. Moreover, KL-geodesics preserve a usable learning signal: as shown in Fig. 2, $\operatorname{KL}(x_1\|x_t)$ decays substantially more slowly along the KL path—especially for large vocabularies ($|\mathcal{V}| = 10,000$)—whereas Linear and Fisher-Rao paths collapse $\operatorname{KL}(x_1\|x_t)$ near zero too early (for t close to 0), effectively turning the transport into a one-shot step and depriving the model of informative gradients over most of the time horizon. Empirically, Table 1 shows that training with Linear or Fisher-Rao objectives yields markedly worse perplexity, consistent with this geometric analysis.

Logit parameterization Write $l_0 = \log x_0$ and $l_1 = \log x_1$, and define the logit-linear interpolation $l_t = (1-t)l_0 + t \, l_1$, so that $x_t = \operatorname{Softmax}(l_t)$. Because \log is undefined at zero, we use a standard β -smoothed target for the one-hot x_1 ,

$$x_1 = (1 - \beta) \delta_i + \frac{\beta}{V} \mathbf{1}, \quad \beta \in (0, 1),$$

where δ_i is the canonical basis vector of the realized token and 1 is the all–ones vector. Equivalently, in logit space we could write linear ODE:

$$\frac{dl_t}{dt} = l_1 - l_0, (5)$$

so the KL path is a straight line in logits whose image under Softmax remains intrinsic to the simplex.

3.1 Denoising Objective

Single-token case. Consider the special case in which the sequence length equals one. The entire input is then represented by a single vector whose dimensionality matches the vocabulary size. As introduced in Definition 3.1 the KL-geodesic reproduces the conditional flow-matching objective equation 3:

$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t,x_0,x_1} \| v_{\theta}(x_t, t) - (l_1 - l_0) \|^2,$$
(6)

where $x_t = \operatorname{Softmax}(l_t)$ denotes the intermediate point obtained by applying the softmax map to the logit vector $l_t = (1-t)\,l_0 + t\,l_1$. The quantities l_0 and l_1 are, respectively, the logits generating the initial state x_0 and the target state x_1 after projection onto the probability simplex. Both the conditional vector field $v(x_t,t\mid x_0,x_1)=l_1-l_0$ and its learnable counterpart $v_\theta(x_t,t)$ admit the following reparametrisation in terms of l_t :

$$l_1 - l_0 = \frac{l_1 - l_t}{1 - t}, \qquad v_\theta(x_t, t) = \frac{\hat{v}_\theta(x_t, t) - l_t}{1 - t},$$
 (7)

Substituting the identities in equation 7 into the loss equation 6 transforms the original objective into a denoising-style regression problem in which the model must recover the clean target logit $l_1 = \log x_1$ from the corrupted observation x_t :

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,x_0,x_1} \left\| \hat{v}_{\theta}(x_t,t) - l_1 \right\|^2. \tag{8}$$

Proposition 3.2. Let $\mathcal{L}_{CFM}(\theta)$ be defined as in equation 8. For every $t \in (0,1)$ and every x_t the function

$$\hat{v}_{\theta}^{\star}(x_t, t) = \mathbb{E}_{x_1 \sim p(x_1|x_t)} l_1 \tag{9}$$

is the (almost surely) unique minimiser of the loss equation 8.

Corollary 3.3. Suppose we approximate the true conditional $p(x_1 \mid x_t)$ with a parametric model $p_{\theta}(x_1 \mid x_t)$. Then an estimate of the vector field compatible with equation 7 is

$$v(x_t, t) = \frac{1}{1 - t} \Big(\mathbb{E}_{x_1 \sim p_\theta(x_1 | x_t)} l_1 - l_t \Big). \tag{10}$$

The subscript θ is omitted in $v(x_t, t)$ to emphasise that learning proceeds through the conditional density $p_{\theta}(x_1 \mid x_t)$, rather than through direct parametrisation of the vector field itself.

Sequences of length S We now extend the analysis from the single-token setting to sequences that contain exactly S tokens. As a prior over sequences we assume S independent Dirichlet distributions, each defined on the (V-1)-simplex associated with the vocabulary of size V. In contrast, the "clean" or target distribution p_1 is supported on the vertices of the Cartesian product of simplices. Following the prescriptions in Stärk et al. (2024); Gat et al. (2024), we interpolate each token independently along the KL-geodesic. Consequently, the logit representation becomes an $S \times V$ matrix l_t whose k-th row $l_t^{(k)}$ corresponds to token k.

Fixing an index $k \in \{1, \dots, S\}$ and specialising Equation equation 9 to the present context yields

$$\hat{v}_{\theta}^{(k)}(x_t, t) = \mathbb{E}_{x_1 \sim p(x_1|x_t)} l_1^{(k)}, \tag{11}$$

where $l_1^{(k)}$ denotes the logits that would generate the clean token $x_1^{(k)}$

Proposition 3.4. For the KL–geodesic described above, the expression in equation 11 factorises over individual tokens, and the optimal vector field for the k-th coordinate can be written as

$$\hat{v}_{\theta}^{(k)}(x_t, t) = \mathbb{E}_{x_1^{(k)} \sim p(x_1^{(k)}|x_t)} l_1^{(k)}, \tag{12}$$

where $p(x_1^{(k)} \mid x_t)$ is the marginal conditional distribution associated with the k-th token.

Consequently, under the KL-geodesic, computing the optimal velocity field reduces to evaluating the exact marginal posteriors $p(x_1^{(k)} \mid x_t)$ for each token k independently. In practice we approximate these posteriors with a parametric model $p_{\theta}(x_1^{(k)} \mid x_t)$. We draw $x_1 \sim p_1$ (from the data distribution) and $t \sim \mathcal{U}(0,1)$, set $x_0 \sim p_0$, and form $x_t = \operatorname{Softmax} \left((1-t) \log x_0 + t \log x_1 \right)$. The model outputs token-wise conditionals $p_{\theta}(x_1^{(k)} \mid x_t)$, for which we minimize the sequence-level NLL:

$$\mathcal{L} = -\mathbb{E}_{t,x_1 \sim p(x_1), x_t \sim p(x_t | x_1)} \sum_{k=1}^{S} \log p_{\theta}(x_1^{(k)} | x_t), \tag{13}$$

A practical realisation of the conditional model $p_{\theta}(x_1^{(k)} \mid x_t)$ can be obtained by adapting a Transformer architecture: the standard causal attention is replaced with bidirectional attention so that the representation of each token has access to the entire sequence x_t , and an additional conditioning mechanism is introduced to incorporate the continuous time variable t.

4 INFERENCE: ITERATIVE SAMPLING SCHEME

We present three complementary inference procedures under the KL–geodesic interpolation introduced earlier: a *deterministic KL–flow integrator*, a *stochastic iterative sampler*, and a *hybrid* routine that combines both. Unless stated otherwise, logits evolve along the logit–linear path

$$l_t = (1-t) l_0 + t l_1, x_t = Softmax(l_t).$$

4.1 DETERMINISTIC INFERENCE VIA KL-FLOW

Within classical flow matching, samples are generated by numerically integrating the ODE associated with the KL-geodesic. For the interpolation in Definition 3.1, the logit vector obeys the linear ODE

$$\frac{dl_t}{dt} = \frac{l_1 - l_t}{1 - t}. ag{14}$$

Algorithm 1 implements an explicit scheme (Euler with step size h=1/N) that advances t from 0 to 1. In experiments we refer to this baseline as **KL-flow (basic)**.

4.2 STOCHASTIC INFERENCE BY DIRECT SIMULATION

An alternative is to *simulate* the one–step transport induced by a small time increment h>0. Conditioning on the current iterate x_t , the next iterate admits the Markov factorization

$$p(x_{t+h} | x_t) = \int p(x_{t+h} | x_1) p(x_1 | x_t) dx_1.$$
 (15)

The exact posterior $p(x_1 \mid x_t)$ is intractable at the sequence level. The optimization of objective from equation 13 gives the product of tokenwise marginals produced by the denoiser:

$$p_{\theta}(x_1 | x_t) = \prod_{k=1}^{S} p_{\theta}(x_1^{(k)} | x_t).$$

Because the KL-geodesic interpolation also factorizes across tokens we obtain the tractable kernel

$$p_{\theta}(x_{t+h} \mid x_t) = \prod_{k=1}^{S} p_{\theta}(x_{t+h}^{(k)} \mid x_t) = \prod_{k=1}^{S} \int p(x_{t+h}^{(k)} \mid x_1^{(k)}) p_{\theta}(x_1^{(k)} \mid x_t) dx_1^{(k)}.$$
 (16)

Iterating these kernels defines an implicit model distribution over terminal states,

$$p_{\theta}(x_1) = p(x_0) p_{\theta}(x_h \mid x_0) \cdots p_{\theta}(x_1 \mid x_{1-h}). \tag{17}$$

This construction underpins the sampling routine summarized below; see Algorithm 2.

Corollary 4.1 (Iterative sampler). To draw $x_1 \sim p_{\theta}(x_1)$, initialize $x_0 \sim p_0$ and iterate for $t = 0, h, 2h, \ldots$:

- (i) For each token k = 1, ..., S, sample $x_1^{(k)} \sim p_{\theta}(x_1^{(k)} | x_t)$.
- (ii) For each token k, advance along the KL-geodesic by sampling $x_{t+h}^{(k)} \sim p(x_{t+h}^{(k)} \mid x_1^{(k)})$.
- (iii) Set $t \leftarrow t + h$ and repeat while t < 1.

This **KL-flow** (sampling) procedure requires one forward pass of the denoiser model $p_{\theta}(x_1^{(k)} \mid x_t)$ per iteration and thus matches the complexity of the ODE solver.

4.3 LIMITATIONS AND HYBRID SOLVER

The denoiser trained with the sequence–level NLL equation 13 furnishes only token–wise marginals $p_{\theta}(x_1^{(k)} \mid x_t)$. Treating these as conditionally independent yields $p(x_1 \mid x_t) \approx \prod_k p_{\theta}(x_1^{(k)} \mid x_t)$. This surrogate is exact at t=1 but may degrade as t decreases due to emerging inter–token dependencies. To balance the stability of early–time deterministic transport, we adopt a **KL-flow** (**hybrid**) procedure: integrate the ODE of Algorithm 1 from t=0 up to a threshold t^* , then switch to the sampler of Algorithm 2 for the remaining horizon. Empirically, this combination improves perplexity/entropy trade–offs relative to either component alone (see Appendix D for selection of t^* and top-k settings).

5 RELATED WORK

Non-autoregressive text generation methods can be divided into those operating in continuous latent spaces Li et al. (2023); Ye et al. (2023); Gong et al. (2022); Strudel et al. (2022) and those working directly with discrete token representations, as considered in this work. Among the latter, Campbell et al. (2024) proposed Discrete Flow Models, which combine Continuous-Time Markov Chains and normalising flows to model both discrete and continuous variables, achieving state-of-the-art results on protein generation. Gat et al. (2024) introduced Discrete Flow Matching, defining sample paths between distributions via learned posterior approximations such as probability denoisers. Stärk et al. (2024) extended this line by proposing Dirichlet Flow Matching, limiting paths to Dirichlet mixtures for tractable density calculations. Davis et al. (2024) developed Fisher-Flow, utilising the Fisher–Rao Riemannian metric to transport mass between categorical distributions along hypersphere geodesics. Alternatively, Lou et al. (2024) presented a diffusion-based approach, generalising score matching to discrete spaces for the construction of discrete diffusion models. These advances collectively demonstrate the strength of flow matching and diffusion methods for discrete generative modelling (see Appendix F for further discussion).

6 EXPERIMENTS

We evaluated KL-Flow on diverse text generation tasks, spanning unconditional language modeling, conditional sequence generation, and code infilling. All models used a bidirectional Transformer backbone (adapted from modded- $NanoGPT^1$), with continuous time embeddings as in DiT Peebles & Xie (2023) and logit interpolation fixed at $\beta=0.01$; top-k sampling (k=1) was used for sampling inference scheme (see Appendix D). We employed two model sizes: a 150M-parameter configuration for TinyStories and a 1.5B-parameter setup for other data domains, following the architectural and hyperparameter details of the original repository. Further hyperparameters and ablation results are provided in Appendix E. KL-Flow was compared with DFM Gat et al. (2024), GPT-2 Jordan et al. (2024), and SEDD Lou et al. (2024). All models taken for comparison were trained from scratch in the same setup and on the same data subset as our proposed KL-Flow model to force comparison validity. All training was conducted on 4 NVIDIA H100 GPUs (80GB each).

6.1 Datasets

Unconditional generation. The TinyStories dataset Eldan & Li (2023) consists of synthetically generated short narratives authored by GPT-3.5 and GPT-4. All models were trained on 4B tokens with the maximum sequence length capped at 512.

https://github.com/KellerJordan/modded-nanogpt

Table 2: Comparison of unconditional text generation models trained on the Tiny Stories dataset. The results of the best-performing models are indicated in **bold**, while the instances where our approach matches or exceeds the performance of alternative Non-Autoregressive (NAR) methods are highlighted in blue.

Method	Grammar ↑	Creativity ↑	Consistency ↑	Plot ↑	Perplexity ↓
GPT 2	5.3	6.4	4.9	4.9	15.4
DFM	3.5	5.7	3.6	3.5	20.8
SEDD	4.2	6.1	4.0	3.8	20.7
KL-Flow	4.4	6.1	4.0	3.7	19.0

To verify the scalability of KL-Flow, we further considered 10 B tokens sampled from the *Fine-FineWeb* dataset M-A-P et al. (2024), which contains deduplicated and quality-filtered English web documents. Each training instance was truncated or padded to a uniform length of 1024 tokens. Models trained on this source served as the initialization (pre-training) for all subsequent conditional-generation experiments.

Conditional text generation was evaluated on two sequences-to-sequence datasets. (i) The *Lamini Instruction* benchmark Wu et al. (2023). (ii) The WMT14 German–English translation dataset Bojar et al. (2014). In both cases the concatenation of the prompt and the ground-truth response was restricted to 512 tokens. Total training exposure was fixed at 4B tokens.

For the code infilling task we curated an open-source Python corpus². Only files comprising fewer than $1\,024$ tokens were retained. During training, for each example a uniformly random proportion between $10\,\%$ and $90\,\%$ of the lines was masked, and the model was instructed to reconstruct the elided span. Generalization was quantified on the MBPP benchmark Austin et al. (2021b).

To ensure the validity of comparisons, all baseline models were trained on the identical data subsets, using the same dataset shuffles and number of tokens to train on.

6.2 EVALUATION TECHNIQUES

The quality of unconditional text generation was evaluated using generative perplexity—measured by scoring generated samples with large language models (GPT-2 Radford et al. (2019), GPT-3 Brown et al. (2020), and Llama-2 Touvron et al. (2023))—and diversity was assessed via empirical entropy (values above 5 indicated substantial lexical variety). For the Tiny Stories dataset, we additionally reported grammar, creativity, consistency, and plot coherence, as in Eldan & Li (2023). When scoring with external LMs (GPT-2/3, Llama-2), we use their tokenizers for perplexity evaluation.

Sequence-to-sequence outputs were measured using ROUGE-L (longest common subsequence overlap) Lin (2004), BERTScore (semantic similarity via contextual embeddings) Zhang et al. (2020), and BLEU (clipped n-gram precision with brevity penalty, $n \leq 4$) Papineni et al. (2002).

Code infilling was evaluated by Pass@k (fraction of synthesized functions passing all unit tests out of k samples) and Compiles@k (fraction of code snippets compiling/executing without syntax errors), for $k \in \{1, 10\}$.

6.3 Unconditional Language Modeling

The experimental evaluation of the proposed framework was carried out with the *KL-Flow* (*hybrid*) inference strategy that was introduced in Section 4. The numerical evidence summarised in Table 2 demonstrates that KL-Flow consistently surpasses all alternative non-autoregressive baselines across the majority of metrics, although the traditional autoregressive GPT-2 model retains an overall lead on this relatively simple dataset. In contrast, the FineFineWeb dataset imposes a significantly higher level of linguistic and semantic difficulty. Table 3 reports perplexity values measured for a range of numbers of function evaluations (NFE). Before analysing comparative performance, we verified that every model under consideration preserves sufficient output variability by computing the empirical entropy of produced token distributions; all entropy scores exceeded the threshold of 5, thereby con-

²https://huggingface.co/datasets/jtatman/python-code-dataset-500k

Table 3: Generative perplexity on unconditional text generation compared to prior work. Models were trained on FineFineWeb dataset. The best results are highlighted in **bold**.

Method	NFE	Llama 2	GPT 3	GPT 2
Data	-	9.2	15.8	31.4
GPT 2	1024	48.7	84.9	97.2
DFM	256/512/1024	150.6/107.3/75.0	312.8/198.9/125.9	381.4/245.8/157.2
SEDD	256/512/1024	70.8/57.7/47.6	123.8/95.7/74.8	145.8/114.2/90.2
KL-flow (150M)	256/512/1024	61.0/47.1/35.1	101.7/75.8/54.1	117.3/88.1/62.9
KL-flow (1.5B)	256/512/1024	51.5/ 41.7/32.7	81.1/63.7/48.4	96.6/76.2/58.5

Table 4: Evaluation of conditional text generation on test set compared to prior works. The best results are highlighted in **bold**.

Dataset	Method	BLEU Score		ROUGE-L		BERT	Score
		Top-5	Avg	Top-5	Avg	Top-5	Avg
	GPT 2	7.8	3.1	28.9	18.2	63.8	56.4
Lamini Instruction	DFM	8.1	3.6	30.0	19.2	61.6	53.6
Lammi instruction	SEDD	5.4	2.1	25.9	15.8	61.0	53.7
	KL-flow (hybrid)	9.5	4.3	34.5	23.9	67.9	61.1
	KL-flow (sampling)	7.7	4.1	31.3	21.5	66.6	60.1
	GPT 2	19.7	9.8	48.3	36.7	78.1	71.0
WMT14 De-En	DFM	21.3	11.2	50.0	38.8	77.1	69.6
WM114 De-Ell	SEDD	14.6	6.5	44.9	34.5	74.2	68.2
	KL-flow (hybrid)	23.8	13.7	53.5	44.7	82.1	77.7
	KL-flow (sampling)	27.0	18.1	56.9	49.4	84.5	81.2

firming generation diversity. When the NFE parameter is kept at its default value 1024, KL-Flow in the intermediate 150M configuration already establishes a clear advantage over both diffusion-based and flow-based non-autoregressive competitors. Reducing the computational budget by a factor of two (NFE equal to 512) does not alter this observation: KL-Flow maintains a comfortable margin. Even under an aggressive four-fold reduction to 256 evaluations, the model preserves performance that is comparable to or superior to GPT-2, underscoring the method's capacity for substantial generation acceleration without sacrificing linguistic plausibility. Scaling the architecture from 150M to 1.5B parameters further accentuates these gains. In the larger setting, KL-Flow attains the best perplexities across all three reference language models (Llama 2, GPT-3, and GPT-2) and for every NFE level examined.

6.4 CONDITIONAL LANGUAGE MODELING

The empirical evaluation of the conditional generation framework was carried out on two complementary benchmarks, namely the Lamini Instruction and the WMT14 German–English translation datasets. Performance was quantified through the standard metrics BLEU, ROUGE-L, and BERTScore; the corresponding results, reported in Table 4, include both the maximum value obtained among the top 5 decoded responses and the mean over this candidates, thereby providing simultaneous insight into peak quality and output stability. Inspection of the numerical results reveals that the *KL-Flow* consistently surpasses all prior works. When the conditional distribution admits multiple plausible continuations, as in the Lamini Instruction scenario, the *hybrid* inference strategy achieves the highest scores across all metrics. By contrast, in the lower-entropy setting of deterministic machine translation, the purely *sampling* based variant exhibits a clear advantage.

<pre>def move_num(test_str):</pre>	<pre>def move_num(test_str):</pre>	<pre>def move_num(test_str):</pre>	<pre>def move_num(test_str):</pre>
res = ''	res = ''	res = ''	res = ''
dig = ''	en = ''	<pre>for test_str, ele:</pre>	Convert given string
<pre>for ele in test_str:</pre>	<pre>for ele in test_str:</pre>	uid, dig = test_str	<pre>for ele in (test_str)):</pre>
<pre>if ele.isdigit():</pre>	<pre>if ele.isdigit():</pre>	<pre>if ele.isdigit():</pre>	<pre>if ele.isdigit():</pre>
dig += ele	dig += ele	dig += ele	dig += ele
else:	else:	else:	else:
res += ele	res += ele	res += ele	res += ele
res += dig	res += dig	res += dig	res += dig
return (res)	return (res)	return (res)	return (res)
(a) KL-flow	(b) DFM	(c) GPT 2	(d) SEDD

Figure 3: An illustrative example of code infilling. The highlighted lines were generated by the model. Lines highlighted in green indicate correct infilling, while those highlighted in red denote incorrect infilling.

Table 5: Quantitative comparison of several code-infilling approaches on the MBPP benchmark. For each masking ratio the two quality indicators Pass@k and Compiles@k are reported for $k \in \{1, 10\}$. The highest value in every column appears in **bold**.

	Infilling 10%		Infilling 50%				Infilling 90%					
Method	Pas	ss@	Comp	iles@	Pa	ss@	Comp	iles@	Pas	s@	Comp	iles@
	1	10	1	10	1	10	1	10	1	10	1	10
GPT-2	8.8	20.1	54.2	92.8	0.7	3.4	27.5	67.6	0.1	0.6	15.7	56.2
DFM	11.1	25.5	39.7	88.8	2.6	8.0	15.7	59.3	0.1	1.1	7.0	33.2
SEDD	9.2	22.1	51.7	93.7	1.8	6.6	30.3	77.9	0.1	0.3	16.8	60.2
KL-Flow	17.4	29.2	73.7	92.0	4.4	11.2	58.1	87.4	0.2	1.7	60.4	90.8

6.5 Code infilling

The code–infilling problem requires a model to reconstruct those program lines that have been removed, using both the surrounding source context and the natural-language task description. In the present study the network must generate a replacement of arbitrary length, up to 40 tokens. During training and evaluation we conceal a randomly chosen fraction of the original lines; this fraction is drawn uniformly between 10% and 90% of code lines. Figure 3 illustrates infilling example. For completeness we adapted GPT-2 baseline to the same setting. Each masked line is replaced by the specified token and the transformer is trained autoregressively so that, after producing the unmasked part of the program, it appends the content of every hidden line in order.

Table 5 summarises the outcomes for three representative masking regimes: 10%, 50%, and 90% of the code are removed. Across all regimes *KL-Flow* model with *hybrid* inference scheme surpasses prior approaches in both functional correctness and syntactic validity. Detailed curves covering the entire masking spectrum appear in Appendix C.

7 CONCLUSIONS AND FUTURE WORK

In this work, we propose using Kullback-Leibler (KL) divergence geodesics—equivalent to linear interpolation in logit space—as a principled approach to flow matching in discrete sequence modeling. Our theoretical analysis shows that the likelihood maximizer precisely matches the exact flow matching velocity, establishing a strong foundation for our method. We also introduce a new empirical *sampling* algorithm which, despite limited theoretical guarantees, consistently outperforms baselines in conditional text modeling on benchmarks such as WMT14 de-en translation and code infilling. Additionally, our *hybrid* inference approach combines both *basic* and *sampling* procedures, achieving strong results in unconditional and conditional generation tasks, including Lamini Instruction dataset. Our findings show that larger models further improve performance, though current progress is limited by computational resources. Therefore, future work should focus on scaling model size and training to unlock further gains.

8 ETHICS STATEMENT

This work does not involve human subjects, personally identifiable information, or any sensitive data. All datasets used are publicly available and widely used in prior research. We are not aware of any ethical issues or potential negative societal impacts related to the methods or results presented in this paper.

9 REPRODUCIBILITY STATEMENT

We have made significant efforts to ensure the reproducibility of our work. Theoretical claims are supported with formal derivations and proofs provided in Sections 3, 4, and Appendix A. The main inference scheme is described in Section 4 and further detailed in Algorithms 1 and 2 in the Appendix. Model architectures, dataset descriptions, training procedures, and hyperparameters are provided in Section 6 and Appendix E. An anonymous implementation of our method, including training and sampling scripts, is also provided in the supplementary submission.

REFERENCES

- Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces, 2021a. URL https://arxiv.org/abs/2107.03006.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models, 2021b. URL https://arxiv.org/abs/2108.07732.
- Ondrej Bojar, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Ales Tamchyna. Findings of the 2014 workshop on statistical machine translation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pp. 12–58, Baltimore, Maryland, USA, 2014. Association for Computational Linguistics. URL https://aclanthology.org/W14-3302.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeff Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL https://arxiv.org/abs/2005.14165.
- 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.
- Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. Maskgit: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 11305–11315, 2022.
- Oscar Davis, Samuel Kessler, Mircea Petrache, Ismail Ilkan Ceylan, Michael M. Bronstein, and Avishek Joey Bose. Fisher flow matching for generative modeling over discrete data, 2024. URL https://arxiv.org/abs/2405.14664.
- 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.

- Ronen Eldan and Yuanzhi Li. Tinystories: How small can language models be and still speak coherent english?, 2023. URL https://arxiv.org/abs/2305.07759.
 - 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.
 - Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019.
 - Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models, 2022. URL https://arxiv.org/abs/2210.08933.
 - Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. Ssd-lm: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL)*, 2022.
 - Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.
 - Keller Jordan, Jeremy Bernstein, Brendan Rappazzo, fernbear, Boza Vlado, You Jiacheng, Franz Cesista, Braden Koszarsky, and Grad62304977. modded-nanogpt: Speedrunning the nanogpt baseline. https://github.com/KellerJordan/modded-nanogpt, 2024.
 - Junyi Li, Wayne Xin Zhao, Jian-Yun Nie, and Ji-Rong Wen. Glyphdiffusion: Text generation as image generation, 2023. URL https://arxiv.org/abs/2304.12519.
 - Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-lm improves controllable text generation, 2022. URL https://arxiv.org/abs/2205.14217.
 - Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pp. 74–81, Barcelona, Spain, 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.
 - Zhenghao Lin, Yeyun Gong, Yelong Shen, Tong Wu, Zhihao Fan, Chen Lin, Nan Duan, and Weizhu Chen. Text generation with diffusion language models: A pre-training approach with continuous paragraph denoise, 2023. URL https://arxiv.org/abs/2212.11685.
 - Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *Proceedings of the Eleventh International Conference on Learning Representations (ICLR)*, 2023. URL https://openreview.net/forum?id=PqvMRDCJT9t.
 - 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.
 - Justin Lovelace, Varsha Kishore, Chao gang Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent diffusion for language generation, 2022. URL https://arxiv.org/abs/2212.09462.
 - M-A-P, Ge Zhang, Xinrun Du, Zhimiao Yu, Zili Wang, Zekun Wang, Shuyue Guo, Tianyu Zheng, Kang Zhu, Jerry Liu, Shawn Yue, Binbin Liu, Zhongyuan Peng, Yifan Yao, Jack Yang, Ziming Li, Bingni Zhang, Minghao Liu, Tianyu Liu, Yang Gao, Wenhu Chen, Xiaohuan Zhou, Qian Liu, Taifeng Wang, and Wenhao Huang. Finefineweb: A comprehensive study on fine-grained domain web corpus. Dataset on Hugging Face, 2024. URL https://huggingface.co/datasets/m-a-p/FineFineWeb. Version v0.1.0.

- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, 2002. Association for Computational Linguistics. URL https://aclanthology.org/P02-1040.
- William S. Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings* of the IEEE/CVF International Conference on Computer Vision (ICCV), pp. 4172–4182, 2023.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. OpenAI Technical Report, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.
- Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Stepunrolled denoising autoencoders for text generation, 2022. URL https://arxiv.org/abs/ 2112.06749.
- Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL https://openreview.net/forum?id=PxTIG12RRHS.
- Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, and Rémi Leblond. Self-conditioned embedding diffusion for text generation, 2022. URL https://arxiv.org/abs/2211.04236.
- Hannes Stärk, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to dna sequence design, 2024. URL https://arxiv.org/abs/2402.05841.
- Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, Alexei V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023. URL https://arxiv.org/abs/2307.09288.
- Minghao Wu, Abdul Waheed, Chiyu Zhang, Muhammad Abdul-Mageed, and Alham Fikri Aji. Lamini-lm: A diverse herd of distilled models from large-scale instructions, 2023. URL https://arxiv.org/abs/2304.14402.
- Jiasheng Ye, Zaixiang Zheng, Yu Bao, Lihua Qian, and Mingxuan Wang. Dinoiser: Diffused conditional sequence learning by manipulating noises, 2023. URL https://arxiv.org/abs/2302.10025.
- Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.
- Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zheng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, Peiyu Liu, Jianyun Nie, and Ji rong Wen. A survey of large language models, 2023. URL https://arxiv.org/abs/2303.18223.

A

A PROOFS OF PROPOSITIONS

Proposition A.1. Let $\mathcal{L}_{CFM}(\theta)$ be defined as in equation 8. For every $t \in (0,1)$ and every x_t the function

$$\hat{v}_{\theta}^{\star}(x_t, t) = \mathbb{E}_{x_1 \sim p(x_1|x_t)} l_1 \tag{18}$$

is the (almost surely) unique minimiser of the loss equation 8.

Proof. Fix an arbitrary pair (x_t, t) . Since equation 8 is quadratic in $\hat{v}_{\theta}(x_t, t)$, its minimiser is obtained by differentiating the integrand with respect to the candidate value and equating the derivative to zero. Concretely,

$$\hat{v}_{\theta}^{\star}(x_t, t) = \frac{1}{p(x_t)} \int l_1 \, p(x_t \mid x_0, x_1) \, p(x_0, x_1) \, \mathrm{d}x_0 \, \mathrm{d}x_1. \tag{19}$$

Assuming an independent coupling $p(x_0, x_1) = p_0(x_0)p_1(x_1)$ and carrying out the integral with respect to x_0 yields

$$\hat{v}_{\theta}^{\star}(x_t, t) = \frac{1}{p(x_t)} \int l_1 \, p(x_t \mid x_1) \, p_1(x_1) \, \mathrm{d}x_1. \tag{20}$$

By Bayes' theorem, $p(x_1 \mid x_t) = \frac{p(x_t \mid x_1) \ p_1(x_1)}{p(x_t)}$. Substituting this identity into equation 20 immediately furnishes equation 9, completing the argument.

Proposition A.2. For the KL–geodesic described above, the expression in equation 11 factorises over individual tokens, and the optimal vector field for the k-th coordinate can be written as

$$\hat{v}_{\theta}^{(k)}(x_t, t) = \mathbb{E}_{x_1^{(k)} \sim p(x_1^{(k)}|x_t)} l_1^{(k)}, \tag{21}$$

where $p(x_1^{(k)} \mid x_t)$ is the marginal conditional distribution associated with the k-th token.

Proof. Because $l_1^{(k)}$ is a deterministic function of $x_1^{(k)}$ alone, one may integrate out all remaining coordinates to obtain

$$\hat{v}_{\theta}^{(k)}(x_t, t) = \int l_1^{(k)} p(x_1 \mid x_t) \, \mathrm{d}x_1 = \int l_1^{(k)} p(x_1^{(k)} \mid x_t) \, \mathrm{d}x_1^{(k)},$$

which coincides with equation 12. While, in principle, the geodesic interpolation could introduce dependencies among tokens through the joint kernel $p(x_t \mid x_0, x_1)$, empirical findings reported in Stärk et al. (2024); Gat et al. (2024) indicate that treating the coordinates independently suffices for practical purposes. Hence, the optimal vector field for each token depends solely on its own marginal posterior. \Box

B ALGORITHMS OF INFERENCE SCHEMES

Here, we present algorithms for basic and sampling inference schemes, see Algorithms 1 and 2.

C ADDITIONAL CODE INFILLING EXPERIMENT

In this section we present full comparison of code infilling task for an arbitrary amount of masked lines. The results were summarized in Figure 4. For most cases the *KL-Flow* outperforms other approaches across all considered metrics. The most noticeable advantage could be seen in Compiles@1 metric, where for any portion of missed code lines the difference from closest competitor is above 10%.

Algorithm 1 Inference scheme (basic)

```
1: Input: Initial distribution p_0; denoiser model p_{\theta}(x_1|x_t); parameter N (number of iterations);
     parameter h (time step size, default 1/N).
 2: Set t = 0
 3: Sample x_t \sim p_0
 4: for i = 1 to N do
          Compute w = p_{\theta}(x_1|x_t)
          Compute smoothed target logits \bar{l}_1 = w \log \left(1 - \beta + \frac{\beta}{V}\right) + (1 - w) \log \left(\frac{\beta}{V}\right)
 6:
          Compute l_t \leftarrow l_t + \frac{h}{1-t}(\bar{l}_1 - l_t)
 7:
          Update x_t \leftarrow \text{Softmax}(l_t)
 8:
 9:
          Update t \leftarrow t + h
10: end for
11: Return x_t
```

Algorithm 2 Inference scheme (sampling)

```
1: Input: Initial distribution p_0; denoiser model p_{\theta}(x_1|x_t); parameter N (number of iterations); parameter h (time step size, default 1/N).

2: Set t = 0

3: Sample x_t \sim p_0

4: for i = 1 to N do

5: Sample x_1^{(k)} \sim p_{\theta}(x_1^{(k)}|x_t) for k \in [1, ..., S]

6: Sample x_0 \sim p_0

7: Compute l_{t+h} = (1 - t - h) \log(x_0) + (t + h) \log(x_1)

8: Update x_t = \operatorname{Softmax}(l_{t+h})

9: Update t \leftarrow t + h

10: end for

11: Return x_t
```

D COMPARISON OF INFERENCE SCHEMES WITH ANALYSIS OF TOP-K SAMPLING EFFECTS

In this section, we systematically examine the effects of inference hyperparameter selection on model performance, focusing on perplexity (as evaluated by the Llama-2 model) and token-level entropy. Two principal factors are considered. The first corresponds to the threshold parameter t^* , which determines the relative proportion of *basic* and *sampling* steps in the *hybrid* inference process. The results displayed in Figure 5 indicate that an optimal compromise between token entropy and perplexity of generated sequences is achieved at $t^* = 0.28$. Notably, at this value, entropy remains above the established diversity threshold of 5. Deviations from the optimal value result in significant reductions in either entropy or perplexity, thus impairing generative quality or diversity.

The second factor concerns the role of top-k sampling during the *sampling* phase of the *hybrid* inference scheme. The impact of varying the k parameter is also explored in Figure 5. An incremental increase in k initially enhances entropy; however, further increases tend to adversely affect text quality as measured by perplexity. Based on these observations, utilizing k=1 is recommended, since it facilitates the generation of sufficiently diverse (entropy > 5) and high-quality texts.

For completeness, Table 6 provides a summary of the performance metrics for all examined inference strategies, specifically comparing the *basic*, *sampling*, and *hybrid* configurations. It can be observed that the *sampling* scheme yields notably lower entropy, a limitation previously discussed in Section 3.2, paragraph **Approximation of the conditional** $p(x_1|x_t)$. Conversely, the *basic* approach is deficient in producing high-quality texts as indicated by elevated perplexity values. The *hybrid* inference strategy, by integrating both approaches, achieves superior overall performance relative to the alternatives.

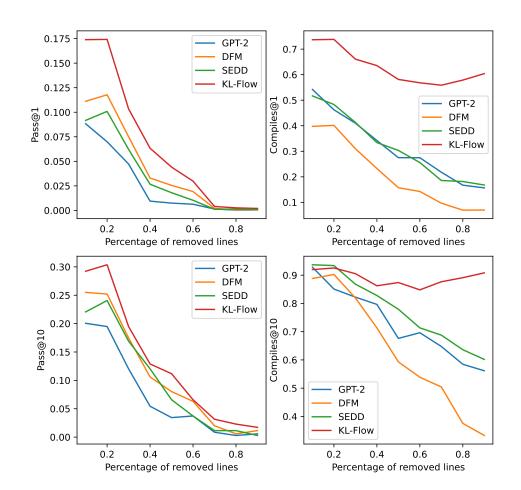


Figure 4: Comparison of the performance of prior models against KL-Flow on the code infilling task

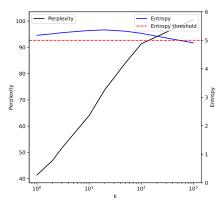
Table 6: Quantitative comparison of inference methodologies with respect to perplexity and entropy.

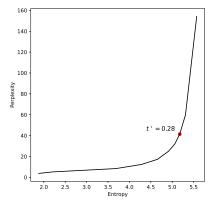
Method	Perplexity	Entropy
KL-Flow (basic)	154.2	5.6
KL-Flow (sampling)	3.8	1.9
KL-Flow (hybrid)	41.4	5.2

E OPTIMAL TRAINING CONFIGURATION

In this section, we discuss several critical aspects and technical strategies for addressing the Flow Matching (FM) problem. The foundational code and architecture employed for training were derived from an open-source GitHub repository featuring an efficient implementation of the GPT-2 model, designed for standard language modeling tasks. However, our investigation revealed that the initially suggested optimal configuration is not truly optimal for the FM problem.

A key factor influencing convergence is the selection of an appropriate learning rate. In Figure 6, we present a comparison of various learning rate values, alongside an assessment of how the integration of the Muon optimizer–proposed in the original repository–affects model performance. We found that the standard learning rate of (lr = 0.0036) is not optimal. A learning rate reduced by a factor of ten significantly accelerates convergence and mitigates the risk of stagnation during the initial phases of training. Furthermore, we determined that the ratio of learning rates between the Adam optimizer and the Muon optimizer yields optimal results. Additionally, the application of the Muon





- (a) Comparison across different top-k values
- (b) Determination of the optimal t^*

Figure 5: Qualitative evaluation of inference hyperparameter selection on model performance. (a) Evaluation of the influence of the top-k parameter within the *sampling* inference scheme. Performance is assessed through perplexity computed via the Llama 2 model and token distribution entropy, with the entropy threshold associated with diverse text generation also indicated. (b) Analysis of the impact of the t^* hyperparameter in the *hybrid* inference approach, identifying the optimal $t^* = 0.28$.

optimizer for specific model parameters enhances convergence, even when employing a non-optimal learning rate.

Another critical consideration is the method of incorporating temporal information into the model architecture. We identified three primary strategies for this purpose:

- Time Token: Transform the time value into an embedding vector and incorporate it as a separate token within the sequence.
- Layer Normalization: Employ a method akin to that used in the DiT architecture, where
 the time embedding is utilized to adjust the mean and standard deviation of the data within
 the layer normalization module.
- Standard Addition: Simply append the time embedding to each token embedding.

Our findings, as presented in Figure 7, indicate that the Layer Normalization strategy is the most effective approach, as it provides better convergence and achieves a lower loss value after 200k training steps.

F RELATED WORK FULL DISCUSSION

In this section, we review the literature on modeling discrete sequences. The authors in Campbell et al. (2024) present Discrete Flow Models (DFMs) that combine discrete and continuous data using Continuous Time Markov Chains, improving traditional diffusion methods for protein co-design and achieving state-of-the-art results in protein structure generation.

Additionally, Song et al. (2021) propose a stochastic differential equation (SDE) for transforming complex data distributions using neural networks for accurate score estimation. The work by Campbell et al. (2022) introduces a continuous time framework for denoising diffusion models of discrete data, resulting in high-performance samplers that surpass traditional methods.

Research by Gat et al. (2024) introduces Discrete Flow Matching, focusing on generating high-dimensional discrete data, such as language, while enhancing generative perplexity. Meanwhile, Ghazvininejad et al. (2019) use masked language modeling to predict target words based on input text, and Austin et al. (2021a) improve multinomial diffusion models. Finally, Hoogeboom et al. (2021) provide extensions for categorical data, demonstrating high efficacy in text modeling and image segmentation.

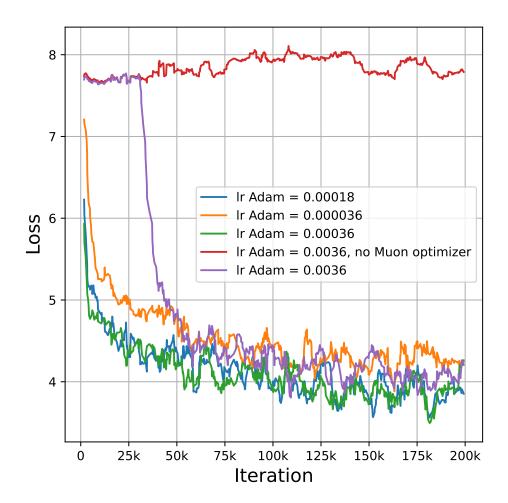


Figure 6: Comparison of the impact of learning rate values on training a GPT-like model for the Flow Matching problem. The base implementation utilizes the Muon optimizer for certain model parameters, while the tag "no Muon optimizer" indicates that the Muon optimizer has been replaced with the Adam optimizer.

Recent advancements have focused on applying continuous space diffusion methods to discrete datasets Dieleman et al. (2022); Li et al. (2022); Han et al. (2022). Notable contributions from Lin et al. (2023) improve diffusion flow modeling, while new Continuous Flow Matching techniques are introduced by Lovelace et al. (2022) and Stärk et al. (2024).

Autoregressive models have been crucial in natural language processing Zhao et al. (2023), exemplified by the GPT-2 model Radford et al. (2019), which showcased the potential of autoregressive approaches in generating coherent text. Research highlights the effectiveness of autoregressive methods in addressing complex linguistic challenges.

Masked generative modeling has emerged as a promising area, utilizing techniques to generate content by obscuring parts of input data Ghazvininejad et al. (2019). Studies by Savinov et al. (2022) refined traditional masking methods, leading to innovations like MaskGIT, which employs advanced techniques for high-resolution image synthesis Chang et al. (2022). Furthermore, Ziv et al. (2024) demonstrated the effectiveness of a text-to-music model, showing that the MaskGIT framework significantly improves the quality of generated outputs.

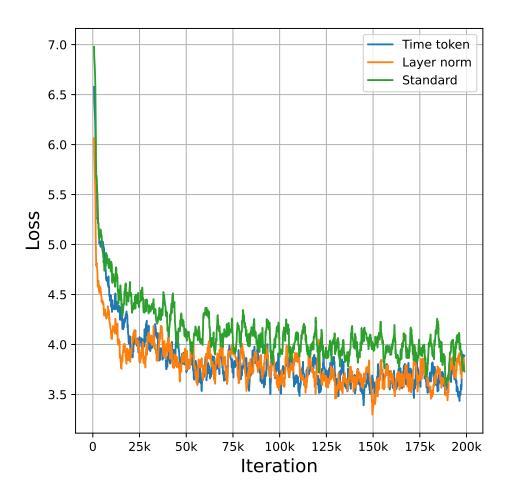


Figure 7: Comparison of various strategies for time insertion within model architecture.