SIMPLM: SIMPLEX-BASED FLOW LANGUAGE MODELING VIA AUTOREGRESSIVE ADAPTATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Autoregressive language models remain the dominant language modeling paradigm, yet their sequential decoding limits efficiency. While recent efforts have explored discrete diffusion language models for parallel decoding, these approaches suffer from severe degradation in text quality when decoding multiple tokens simultaneously due to token-wise independence assumptions. Continuous-space diffusion and flow models offer a compelling alternative, but existing methods are often restricted to unconditional full-sequence generation, and some rely heavily on learned embedding encoders to map discrete tokens into continuous latent states, requiring additional pretrained encoders. In this work, we propose SIMPLM, a continuous flow language modeling framework that operates directly on the probability simplex in the vocabulary space. By applying Gaussian interpolation to the probability states, we derive a closed-form posterior logit correction that yields an analytic skip connection for logit transfer, bypassing the hidden-state information bottleneck in Transformer backbones. Beyond this, SIMPLM natively supports conditional and block-based semi-autoregressive generation via a vectorized time formulation that assigns separate time variables to block positions, providing a principled foundation for continuous-time conditional generation. Furthermore, SIMPLM can adapt a pretrained autoregressive model into a continuous flow model, reducing the cost of training non-autoregressive models from scratch. Empirically, with only 30B training tokens, SIMPLM achieves a generative perplexity below 20 at block size and generation length of 128.

1 INTRODUCTION

Autoregressive language models remain the foundational paradigm for modern language modeling (Vaswani et al., 2017; Brown et al., 2020), achieving remarkable success in a wide range of text-generation tasks. However, their inherent sequential decoding bottleneck requires a full forward pass for each generated token. This has triggered rapid interest in parallel decoding as an alternative to fully autoregressive generation (Gu et al., 2017; Lee et al., 2018; Stern et al., 2019; Sahoo et al., 2024; Lee et al., 2026). In this work, we aim to design a parallel language generation framework that achieves high-quality generation, supports conditional prompting, and efficiently leverages the vast knowledge already encoded in pre-trained autoregressive models.

Despite recent progress, achieving all three goals remains an open problem. Discrete diffusion models (Austin et al., 2021; Lou et al., 2023; Sahoo et al., 2024; Nie et al., 2026) have demonstrated promising scaling behavior, but, as emphasized by Lee et al. (2026), they suffer from a fundamental structural limitation: when updating or generating multiple token positions simultaneously, they factorize predictions across positions, imposing conditional independence among the tokens being generated. Continuous diffusion and flow models can avoid these conditional independence limitations by modeling smooth generative trajectories in continuous space (Gulrajani & Hashimoto, 2023; Chen et al., 2026; Hu et al., 2026; Zhang et al., 2026). However, existing approaches mainly target unconditional generation rather than flexible conditioning language modeling, and many rely on dense embedding-space parameterizations or auxiliary encoders.

In this work, we propose SIMPLM, a continuous flow-matching framework built on a Gaussian path in vocabulary space and a probability-simplex representation. Instead of mapping discrete tokens into a learned embedding space, SIMPLM defines the continuous trajectory directly over vocabulary

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.

coordinates, with clean endpoints corresponding to exact one-hot token vectors. This preserves the token-coordinate structure of the state, allowing us to derive a closed-form posterior logit term from each noisy intermediate state.

The key insight is that the Gaussian-interpolated vocabulary state already contains token-level evidence about the clean token. Since the path is defined directly in vocabulary space, this evidence has a closed-form posterior logit term. We add this term to the model output as an analytic logit skip connection. This connection allows vocabulary state information to bypass the compact Transformer hidden-state bottleneck, allowing the Transformer to focus more on modeling contextual semantics. Complementarily, we use the same posterior logits to form a probability simplex input representation: instead of projecting the raw, noisy vocabulary vector, we first convert it into a smoothed token distribution. This filters unlikely-token noise and presents the Transformer with a more coherent input.

Beyond the simplex formulation on the vocabulary space, SIMPLM supports conditional and block-based semi-autoregressive generation through a vectorized time formulation. By assigning separate time coordinates to different sequence positions, the prefix can remain fixed while a suffix block evolves along its own continuous denoising trajectory, providing a principled framework for continuous-time conditional generation. Importantly, this design enables direct AR-to-flow adaptation. Since SIMPLM keeps both the model interface and the prediction target aligned with standard token probabilities, a pretrained autoregressive model can be converted into a continuous flow model rather than trained from scratch. We stabilize this AR-to-flow transition with a progressive block-size curriculum. Training begins at block size 1, where the pretrained AR backbone operates near its original next-token prediction regime, and then gradually increases the block size to expose the model to noisy multi-token suffix blocks. This curriculum shifts the model from clean-prefix autoregressive prediction to blockwise continuous denoising, achieving a generative perplexity below 20 after 30B training tokens, with block size and generation length of 128.

Our main contributions are summarized as follows.

- We introduce a simple posterior logit correction that maps vocabulary-space Gaussian paths into a token-aware simplex representation, making continuous states easier for Transformer backbones to process.
- We develop a vectorized time formulation for conditional and blockwise flow generation.
- We show how to convert pretrained autoregressive language models into continuous flow models with much less training cost than pretraining from scratch.

2 RELATED WORK

Autoregressive language modeling. Autoregressive language models factorize the sequence distribution as $p(y) = \prod_{l=1}^{L} p(y^l \mid y^{<l})$, predicting each token conditioned on its prefix (Bengio et al., 2003). This factorization is widely used in modern language models, but it also imposes a sequential decoding bottleneck: generation requires one forward step per token, limiting inference parallelism (Gu et al., 2017; Stern et al., 2018).

Discrete diffusion language models. A large line of work applies diffusion to discrete variables, including structured discrete diffusion models (Austin et al., 2021) and masked non-autoregressive decoding methods (Chang et al., 2022). Recent token-space diffusion models have scaled impressively. SEDD (Lou et al., 2023) extended continuous score-matching concepts to discrete spaces, while MDLM (Sahoo et al., 2024) showed that masked diffusion objectives can become competitive on standard language-modeling benchmarks. Subsequent work has explored discrete flow matching (Gat et al., 2024), blockwise diffusion (Arriola et al., 2025), and adaptation from pretrained autoregressive models (Gong et al., 2025). LLaDA (Nie et al., 2026) and Dream (Ye et al.) further validated large-scale discrete diffusion pretraining. These approaches operate in discrete token space, whereas our method utilizes a continuous Gaussian path while maintaining a strict probabilistic interpretation of tokens.

Continuous diffusion and flow language models. Complementary to discrete token-space diffusion, continuous diffusion and flow formulations provide a smooth geometry for vector-field

regression and continuous-time sampling. Early representatives include Diffusion-LM (Li et al., 2022) and CDCD (Dieleman et al., 2022), which emphasize controllable generation and continuous latent embeddings. Plaid (Gulrajani & Hashimoto, 2023) showed that continuous embedding-space diffusion can be competitive with autoregressive language models on a small scale. Recent work has further advanced continuous text diffusion and flows along several directions: scaling and likelihood evaluation (Gulrajani & Hashimoto, 2023), Seq2Seq generation (Zhang et al., 2026; Gong et al., 2022; Lin et al., 2023), simplex-based formulations (Han et al., 2023; Cheng et al., 2024), position-dependent diffusion denoising (Wu et al., 2023; Chen et al., 2024), geometry-aware formulations (Jo & Hwang, 2026; Chen et al., 2026), generation through continuous embedding dynamics (Hu et al., 2026), and one-step acceleration via flow maps (Lee et al., 2026).

3 Preliminary

3.1 Continuous Flow Matching

Continuous flow matching (Lipman et al., 2022) provides a framework for transporting a prior noise distribution p_0 to a complex data distribution p_1 via a continuous-time vector field. For a scalar time $t \in [0, 1]$, let $x_0 \sim p_0$ be a noise sample and $x_1 \sim p_1$ be a clean data sample. A standard probability path is defined by the linear interpolant:

$$x_t = (1 - t)x_0 + tx_1. \tag{1}$$

The marginal probability flow is governed by the ODE $dx_t = v_t(x_t)dt$. The target vector field that generates this path is determined by the conditional expectation of the trajectory derivative:

$$v_t(x_t) = \mathbb{E}[x_1 - x_0 \mid x_t]. \tag{2}$$

To learn this continuous transport, a neural network $v_\theta(x_t, t)$ is typically optimized using a Mean Squared Error objective to regress against the target velocity.

3.2 Vocabulary-Space Flow and Cross-Entropy

Following recent advances in continuous-space language modeling, we build on the vocabulary-space formulation introduced by Lee et al. (2026). Let V be the vocabulary size and L the sequence length. A clean token sequence is represented directly in the vocabulary space as a matrix of one-hot vectors $x_1 \in \{e_1, \dots, e_V\}^L$. The initial noise distribution is a unit Gaussian in this space, $x_0 \sim \mathcal{N}(0, \sigma^2 I)$.

Under the linear interpolant $x_t = (1 - t)x_0 + tx_1$, the instantaneous velocity is $x_1 - x_0$. Because the noise component can be expressed as $x_0 = \frac{x_t - tx_1}{1-t}$, estimating the vector field $v_t(x_t)$ is mathematically equivalent to predicting the clean data state $\hat{x}_1 = \mathbb{E}[x_1 \mid x_t]$.

Crucially, because the target x_1 represents a one-hot categorical variable, its conditional expectation $\mathbb{E}[x_1 \mid x_t]$ is exactly equivalent to the categorical probability distribution $p(x_1 \mid x_t)$. Consequently, the model can be naturally optimized using a standard cross-entropy loss (Lee et al., 2026).

3.3 Likelihood Evaluation via Continuous Normalizing Flows

To evaluate the true perplexity of continuous flow models over given text, LangFlow (Chen et al., 2026) demonstrated that one can compute the exact likelihood of an intermediate state using the instantaneous change-of-variables formula standard in continuous normalizing flows. Along the ODE trajectory, the log-likelihood evolves according to the divergence of the vector field:

$$\frac{d \log p(x_t)}{dt} = -\nabla_{x_t} \cdot v_\theta(x_t, t). \tag{3}$$

Integrating this ordinary differential equation from the noise distribution at $t = 0$ up to a later time t yields the exact log-probability of that continuous state:

$$\log p(x_t) = \log p(x_0) - \int_0^t \nabla_{x_t} \cdot v_\theta(x_t, \tau) d\tau, \tag{4}$$

where $\nabla_{x_t} \cdot v_\theta(x_t, \tau)$ is calculated using the Hutchinson method over the Jacobian.

However, because the velocity field structurally approaches a singularity at the exact clean endpoint $t = 1$ (where the variance collapses to zero), the ODE integration is stopped at a near-clean state $t_{\text{clean}} = 1 - \epsilon$ to ensure numerical stability.

To bridge this continuous near-clean state $x_{t_{\text{clean}}}$ back to the discrete token sequence x_1, the overall model is evaluated using an Evidence Lower Bound (ELBO). By treating $x_{t_{\text{clean}}}$ as a latent variable, the true log-likelihood of the discrete text is bounded by:

$$\log p(x_1) \geq \mathbb{E}_{q(x_{t_{\text{clean}}}|x_1)} \Big[\underbrace{\log p_\theta(x_1 \mid x_{t_{\text{clean}}})}_{\text{Reconstruction Error}} + \underbrace{\log p(x_{t_{\text{clean}}})}_{\text{CNF Integration}} - \underbrace{\log q(x_{t_{\text{clean}}} \mid x_1)}_{\text{Forward Posterior}} \Big]. \tag{5}$$

For our linear Gaussian vocabulary-space flow, the forward posterior $q(x_{t_{\text{clean}}} \mid x_1)$ admits a closed-form analytic expression. The complete mathematical derivation of this bound is provided in Appendix A.1. We also improve this bound by incorporating a multi-sample trick (Appendix A.2).

3.4 Self-Conditioning

As demonstrated in recent continuous language models (Lee et al., 2026; Chen et al., 2026; Hu et al., 2026), self-conditioning is a vital mechanism to ensure global semantic consistency during generation. During training, rather than conditioning the network solely on the noisy intermediate state x_t, the procedure executes an initial forward pass to obtain a preliminary estimate of the clean state $\hat{x}_1 = f_\theta(x_t, \varnothing, t)$. A second forward pass is then performed using this generated estimate as an auxiliary input, computing the final output as $f_\theta(x_t, \hat{x}_1, t)$. During inference, this auxiliary conditioning is provided iteratively by recycling the model's predicted clean state from the previous ODE integration step.

4 Method

To build a continuous flow model that supports flexible conditional generation, we structure our method around the established vocabulary-space flow. We first introduce block size curriculum scheduling to adapt a pretrained autoregressive model to a flow model. We then introduce a theoretical framework for flow matching in a vectorized time domain and prove that independent time schedules can be safely applied to flow models, enabling blockwise transport. Finally, we introduce an analytic probability-simplex representation that aligns this continuous transport with pretrained autoregressive language models, enabling efficient adaptation to continuous flow decoding. Our Detailed algorithm pseudocode is provided in Appendix C.

4.1 Block Size Curriculum Scheduling

Adapting a pre-trained autoregressive model provides a compute-efficient way to leverage the linguistic knowledge and generation capabilities already in large-scale language models. However, adapting such a model to parallel, continuous-time generation introduces a severe distribution shift. The backbone is abruptly exposed to highly noisy, multi-token states that deviate from the one-step, clean-prefix regime on which it was originally trained. To mitigate model collapse, we propose a progressive block-size curriculum.

Training initializes with a block size of 1, effectively preserving the autoregressive paradigm while introducing continuous noise dynamics. As training progresses, we gradually widen the size of the generation block. This curriculum smoothly transitions from near-autoregressive modeling to highly non-autoregressive parallel decoding.

4.2 Vectorized Time and Arbitrary Path Transport

To support conditional generation, we extend the scalar time $t \in [0, 1]$ to a *vectorized time* $\mathbf{t} \in [0, 1]^L$, where L is the sequence length. Given a noise state x_0 and a clean one-hot token sequence x_1, the coordinate-wise linear interpolant and local-time target vector field are:

$$x_{\mathbf{t}} = (\mathbf{1} - \mathbf{t}) \odot x_0 + \mathbf{t} \odot x_1, \quad v_{\mathbf{t}}(x_{\mathbf{t}}) = \mathbb{E}[x_1 - x_0 \mid x_{\mathbf{t}}].$$

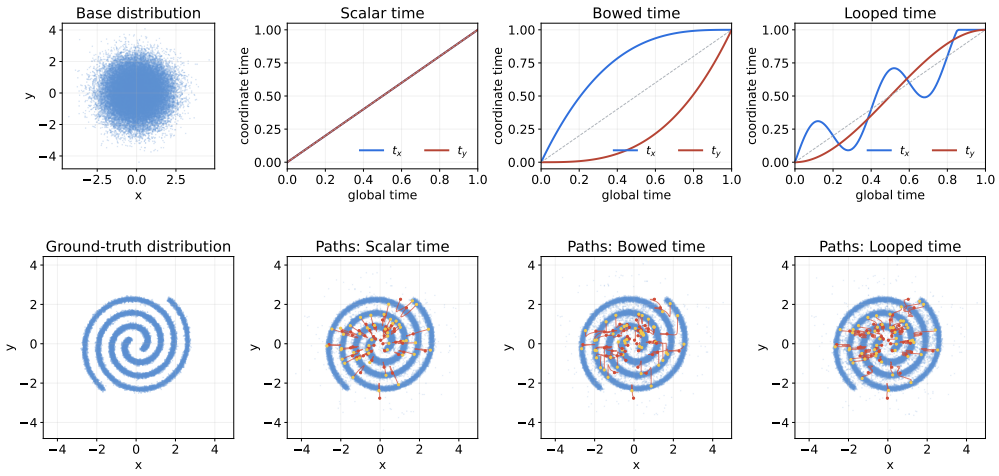


Figure 1: **2D visualization of vectorized time flow modeling.** Using different vectorized time schedules (scalar, bowed, and looped) successfully recovers the identical ground-truth distribution. The start and end points of the transport paths are marked in red and yellow, respectively. Details on generating this visualization can be found in Appendix. E.

We parameterize a scalar-time flow $u \in [0, 1]$ along an arbitrary transport curve using a differentiable schedule $\mathbf{f} : [0, 1] \to [0, 1]^L$, where $\mathbf{f}(0) = \mathbf{0}$ and $\mathbf{f}(1) = \mathbf{1}$. The state along this scheduled path is $x_u = (\mathbf{1} - \mathbf{f}(u)) \odot x_0 + \mathbf{f}(u) \odot x_1$.

Because x_u interpolates between the noise prior p_0 and the data distribution p_1, it forms a valid flow model with target vector field $v_u(x_u) = \mathbb{E}\big[\frac{dx_u}{du} \mid x_u\big]$. Crucially, as derived in Appendix A.3, differentiating this path reveals a direct equivalence to the underlying vectorized field:

$$v_u(x_u) = \mathbf{f}'(u) \odot v_{\mathbf{f}(u)}(x_u). \tag{6}$$

Thus, integrating the scalar-time flow along any differentiable schedule $\mathbf{f}$ is exactly equivalent to following the vectorized-time field with coordinate-wise time rescaling. Figure 1 provides a 2D illustration of this idea: different coordinate-wise schedules, including scalar, bowed, and looped schedules, induce different transport trajectories and coupling while recovering the same endpoint distribution.

Corollary (Blockwise Conditional Generation). *Let $\mathcal{P}$ denote the index set of a conditional prefix. By defining a schedule where $f^i(u) = 1$ for all $i \in \mathcal{P}$, we have $f^{i\prime}(u) = 0$. Consequently, $v_u^i(x_u) = 0\ \forall i \in \mathcal{P}$.*

This guarantees that integration $v_u(x_u)$ natively transports the probability mass along suffix dimensions without perturbing the exact semantic state of the prefix. Furthermore, this theorem demonstrates that our formulation is not restricted to block schedules. By assigning different local times to different positions (e.g., $[1, 0.75, 0.5, 0.25, 0, 0]$), vectorized time naturally supports overlapping denoising trajectories across successive tokens. This unlocks a continuous spectrum of decoding strategies that interpolate between pure parallel block generation and strict autoregressive decoding.

4.3 Analytic Posterior for Logit Skip Connection

We now turn to the model parameterization. In vocabulary-space flows, models need to predict a distribution over a large vocabulary (e.g., $|V| \approx 50$k) using a compact hidden dimension (e.g., $d = 768$). This dimension mismatch induces a severe representation bottleneck, preventing models from accurately recovering the information x_t carries. To alleviate this, we introduce an analytic logit skip connection. This connection explicitly offloads the information recovery process, allowing models to focus on meaningful semantics.

Under the Gaussian path, estimating the velocity reduces to predicting the clean-token posterior $p(x_1 \mid x_t)$ (Section 3.2). Because the forward path is Gaussian, this posterior has a closed-form structure. For a specific position with clean token i and time t, the forward process yields $P(x_t \mid x_1 = e_i) \sim \mathcal{N}(te_i, (1-t)^2\sigma^2 I)$. Using the Bayesian formula (detailed derivation can be found in

Appendix A.4), we have the following:

$$\log p(x_1 = e_i \mid x_t) \propto \frac{t}{(1-t)^2\sigma^2} x_t^i.$$

Based on this, we define the analytic skip connection as:

$$\ell_i^{\text{skip}}(x_t, t) = \frac{t}{(1-t)^2\sigma^2} x_t^i. \tag{7}$$

In practice, we add this term directly to the neural network's output. Let $\ell_\theta^{\text{net}}(x_t, t)$ denote the residual logits predicted by the model, then actual prediction logit is:

$$\log p_\theta(x_1 = e_i \mid x_t) \propto \ell_\theta^{\text{net}}(x_t, t)_i + \ell_i^{\text{skip}}(x_t, t).$$

In addition to resolving the representation bottleneck, this skip connection ensures that the predicted clean-state distribution becomes strictly one-hot as $t \to 1$. Because the scaling factor $t/(1-t)^2$ approaches infinity near the clean endpoint, the skip connection amplifies the logit margin between the clean-token coordinate and all other coordinates, forcing the softmax distribution to collapse to the clean one-hot token in the limit. Thus, inference does not require an additional hard argmax projection.

4.4 Flow Model on the Probability Simplex

Just as CNNs leverage shift invariance as a crucial structural inductive bias in images (Cohen & Welling, 2016), continuous language models can benefit from an inductive bias that respects the inherent symmetry of the probabilistic state space. However, direct projection of the raw continuous state x_t through the embedding matrix $E \in \mathbb{R}^{V \times d}$ lacks this symmetry. Moreover, direct projection of x_t mixes Gaussian perturbations across all vocabulary coordinates, including low-probability ones, obscuring the informative signal.

To resolve these inefficiencies, we introduce a probability-simplex representation:

$$q_t(i \mid x_t) = \text{softmax}_i\left(\frac{t}{(1-t)^2\sigma^2} x_t\right). \tag{8}$$

Rather than directly projecting the raw state, we construct the model input as $h_t = q_t E$. The network then processes h_t to predict the residual logits $\ell_\theta^{\text{net}}(x_t, t)_i$ This explicit information extraction naturally enforces the desired structural inductive bias: any two states x_t with the same probabilistic information are mapped to the same distribution (e.g., any x_0 maps to a uniform distribution). It also filters noise by reducing the influence of unlikely tokens.

5 Experiments

In this section, we validate SimpLM through a series of ablations designed to isolate the contributions of the logit skip connection and the probability-simplex representation.

5.1 Experimental Setup

Backbone and Architecture. We initialize our backbone using the pre-trained Pythia-160M model (Biderman et al., 2023). To support self-conditioning, we introduce an auxiliary MLP component that projects the model's previous clean prediction back into the embedding space and aggregates it with the current input. Notably, because our probability-simplex representation inherently encodes time-dependent posterior information, we can remove the explicit time conditioner. In contrast, for baseline models operating directly on Euclidean noise vectors, we explicitly inject the scalar time t using sinusoidal embeddings followed by an MLP projection.

Datasets. We train all models in the OpenWebText corpus (Gokaslan et al., 2019), which is partitioned into a 90% training split and a 10% validation split.

Baselines. Given the prohibitive computational cost of large-scale pre-training, our empirical evaluation focuses on rigorous ablation studies.We isolate the core methodological contributions by comparing five configurations: (1) a baseline using standard Euclidean-space dense vectors without skip connections (Lee et al., 2026), (2) standard Euclidean-space dense vectors combined with our analytic skip connection, (3) our probability-simplex representation without skip connections, (4) the full SIMPLM incorporating both the simplex representation and the posterior logit skip connection, and (5) the full SIMPLM but without initialization from a pretrained autoregressive model.

Training Recipe. To stabilize the adaptation of a pre-trained autoregressive model into a parallel flow framework, we employ a block-size curriculum (Section 4.1). The curriculum consists of three stages: 320M tokens with block size 1, 640M tokens with block size 4, and 2.7B tokens with block size 32. For training without AR initialization, we do not use block size scheduling. For qualitative sampling, we further train a block-size-128 model for 24B tokens.

5.2 MAIN RESULTS

In the following evaluation, we demonstrate the effectiveness of combining the analytic logit skip connection with the probability-simplex representation. We believe that combining these two methods greatly offloads the process of information extraction and recovery, freeing the model to focus on meaningful semantics. We also show that initializing from a pretrained AR model greatly saves training compute.

Loss Trajectories. We used ablation studies to isolate the contributions of our structural improvements. As illustrated in Figure 2, SIMPLM establishes a Pareto-optimal loss profile throughout the continuous integration trajectory. For any given time t, regardless of stages, our complete SIMPLM training recipe yields a lower cross-entropy loss than all ablated variants in the validation set, including the naive dense baseline lacking both the simplex representation and the skip connection. Moreover, SIMPLM achieves a much lower validation loss than training from scratch, indicating more efficient training behavior.

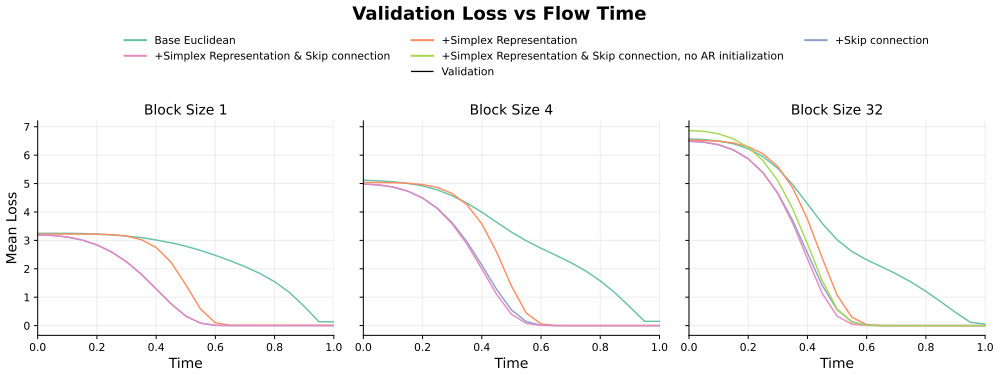


Figure 2: **Validation cross-entropy loss vs. flow time t.** Our method achieves a Pareto-optimal curve. Curves for block sizes 1, 4, and 32 are extracted at steps 8k, 24k, and 100k (i.e., at the end of training for each curriculum stage).

Likelihood Evaluation. We further evaluate generative capacity using multi-sample ELBO perplexity estimation (defined in Section 3.3). As shown in Figure 3, SIMPLM achieves a substantially lower ELBO perplexity compared to all baselines in all generation settings. This convergence behavior directly supports the Pareto-optimal loss curves, confirming both the superiority of the modeling over the probability simplex and the efficiency of adaptation from an AR-pretrained model.

In particular, the base Euclidean baseline exhibits exceptionally high ELBO perplexity, and we attribute this to a representational bottleneck: Without the skip connection and the probability-simplex representation, the model lacks the capacity to handle the token-level evidence in x_t. The skip connection plays the primary role, which, like residual connections, directly injects the prior signal from x_t into the output logits, avoiding the need to transmit this high-dimensional vocabulary-state information entirely through compact hidden states of the model. This explains the large loss

reduction observed when adding the skip connection. Moreover, the simplex representation provides a complementary input-side benefit. By converting the raw vocabulary vector into a smoothed token distribution, it filters unlikely-token noise and makes the input more coherent. Without these structures, the baseline tends to underuse the evidence in x_t and falls back toward a marginal token distribution, leading to higher loss and worse ELBO perplexity.

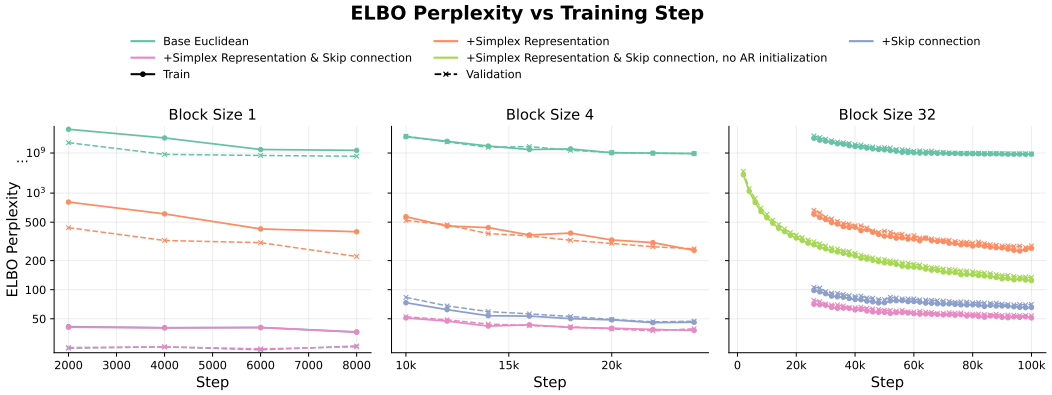


Figure 3: **Training & Validation ELBO perplexity.** Our full formulation achieves the lowest ELBO perplexity. The ELBO integration is computed using an RK4 solver with 64 steps.

Generation Quality and Diversity. Following prior work in continuous language modeling (Chen et al., 2026; Hu et al., 2026), we assess generation quality using generative perplexity from an external GPT-2 Large evaluator (Radford et al., 2019), together with unigram entropy as a coarse measure of lexical diversity. Figure 4 tracks these metrics during training for a block size of 32. The skip-connection baseline and our full method form a clearly stronger group than the remaining baselines, achieving substantially lower generative perplexity while maintaining comparable lexical diversity. Within this group, however, the two models perform similarly on both indirect generation metrics.

This similarity does not necessarily imply equivalent model quality. External generative perplexity is measured using a separate autoregressive evaluator rather than the actual likelihood of the flow model itself, while unigram entropy captures only token-level diversity. We therefore treat these sampling-based metrics as complementary diagnostics and regard validation loss and multi-sample ELBO perplexity as more direct indicators of model quality. Both likelihood-based evaluations consistently favor our full formulation over the skip-connection baseline.

For further manual inspection, we present text samples generated with a block size of 128 in Appendix D. With a prefix length of 128, a generation length of 128, and 256 sampling steps, this model achieves a generative perplexity of 15.9 and a unigram entropy of 3.91.

Varying Generation Lengths. We evaluate conditional generation across varying prefix and generation lengths (Table 1) using a fixed block size of 32. Across all four settings, the two skip-connected models achieve much lower generative perplexity than the other baselines, while remaining close to each other. These results establish the importance of the skip connection but do not, by themselves, establish the advantage of simplex representation.

The likelihood-based evaluations resolve this ambiguity. SIMPLM has a consistently lower time t loss and ELBO perplexity than the skip-connection baseline (Figures 2 and 3). Because these metrics evaluate the model's predictive distribution on held-out data rather than only the properties of sampled text under an external evaluator, we regard them as the more direct evidence that the simplex representation improves the learned flow model.

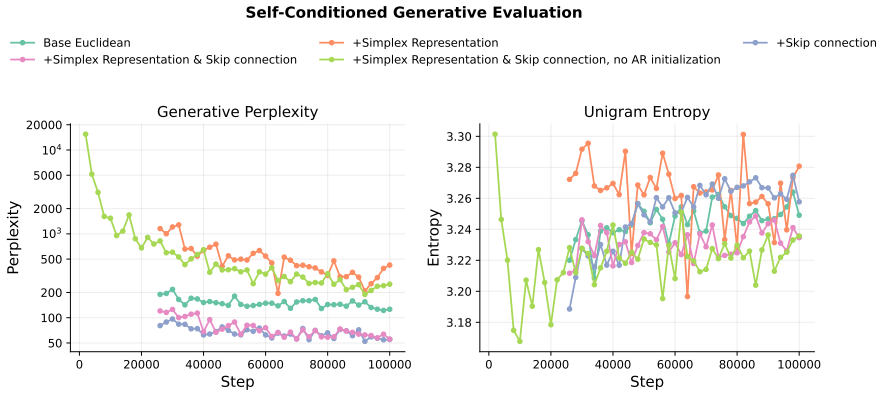


Figure 4: **Generative perplexity and unigram entropy.** The skip-connection baseline and our full method achieve substantially lower generative perplexity than the remaining baselines and perform comparably to each other on both indirect generation metrics. Evaluated with a prefix length of 64, a generation length of 32, and 64 flow steps.

Table 1: **Generative perplexity (PPL) and unigram entropy (Ent.)** across prefix/suffix settings. 'Simp.' and 'Skip' refer to the simplex representation and skip connection, respectively. Full results with 95% confidence intervals are provided in Appendix F.

Prefix	Suffix	Base		+Simp.		+Skip		+Simp. & Skip (Ours)		+Simp. & Skip (w/o AR)	
		PPL ↓	Ent. ↑	PPL ↓	Ent. ↑	PPL ↓	Ent. ↑	PPL ↓	Ent. ↑	PPL ↓	Ent. ↑
128	128	126.8	4.20	616.3	4.36	45.0	4.03	45.1	4.01	162.6	4.12
128	256	105.2	4.56	757.1	4.89	28.9	4.18	28.3	4.17	108.3	4.39
256	128	116.9	4.21	548.4	4.36	43.5	4.04	44.4	4.01	157.3	4.12
256	256	105.5	4.57	659.6	4.87	31.5	4.22	32.4	4.21	111.5	4.40

6 Conclusion and Limitations

In this work, we presented SimpLM, a flow-based language-modeling formulation that combines a vocabulary-space Gaussian path with a probability-simplex representation and an analytic posterior logit skip connection. The resulting model keeps a direct connection to token probabilities while enabling continuous-time conditional block generation with a standard Transformer backbone. We also discovered a simple way to adapt pretrained AR models into continuous vocabulary-space flow models using staged block-size scaling training.

Limitations. Due to computational constraints, we are not able to scale our empirical evaluation to compete with state-of-the-art continuous language models. Instead, our experiments are centered on rigorous ablation studies. This design explicitly isolates and validates the algorithmic benefits of our proposed simplex representation and analytic posterior correction, establishing a structural foundation for future large-scale continuous language modeling.

AI use statement

In this work, we used generative AI tools for polishing paper language and coding. We have not used generative AI tools for idea generation, running experiments or other tasks. We have reviewed all AI-assisted work. We checked LLM-generated language and code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

Reproducibility statement

We have uploaded core code to reproduce our results in the supplementary material. We have also described our methods in details in C.

REFERENCES

Marianne Arriola, Aaron Gokaslan, Justin Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, volume 2025, pp. 50726–50753, 2025.

Jacob Austin, Daniel D Johnson, Jonathan Ho, Daniel Tarlow, and Rianne Van Den Berg. Structured denoising diffusion models in discrete state-spaces. *Advances in neural information processing systems*, 34:17981–17993, 2021.

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. *Journal of machine learning research*, 3(Feb):1137–1155, 2003.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International conference on machine learning*, pp. 2397–2430. PMLR, 2023.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

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*, pp. 11315–11325, 2022.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. *Advances in Neural Information Processing Systems*, 37:24081–24125, 2024.

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

Chaoran Cheng, Jiahan Li, Jian Peng, and Ge Liu. Categorical flow matching on statistical manifolds. *Advances in Neural Information Processing Systems*, 37:54787–54819, 2024.

Taco Cohen and Max Welling. Group equivariant convolutional networks. In *International conference on machine learning*, pp. 2990–2999. PMLR, 2016.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky TQ Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. *Advances in Neural Information Processing Systems*, 37: 133345–133385, 2024.

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

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and LingPeng Kong. Diffuseq: Sequence to sequence text generation with diffusion models. *arXiv preprint arXiv:2210.08933*, 2022.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, et al. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations*, volume 2025, pp. 5046–5073, 2025.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor OK Li, and Richard Socher. Non-autoregressive neural machine translation. *arXiv preprint arXiv:1711.02281*, 2017.

Ishaan Gulrajani and Tatsunori B Hashimoto. Likelihood-based diffusion language models. *Advances in Neural Information Processing Systems*, 36:16693–16715, 2023.

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 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11575–11596, 2023.

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

Jaehyeong Jo and Sung Ju Hwang. Continuous diffusion model for language modeling. *Advances in Neural Information Processing Systems*, 38:97394–97430, 2026.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M Boffi, and Jinwoo Kim. One-step language modeling via continuous denoising. *arXiv e-prints*, pp. arXiv–2602, 2026.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1173–1182, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1149. URL https://aclanthology.org/D18-1149/.

Xiang Li, John Thickstun, Ishaan Gulrajani, Percy S Liang, and Tatsunori B Hashimoto. Diffusion-lm improves controllable text generation. *Advances in neural information processing systems*, 35: 4328–4343, 2022.

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. In *International Conference on Machine Learning*, pp. 21051–21064. PMLR, 2023.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2023.

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. *Advances in Neural Information Processing Systems*, 38:50608–50646, 2026.

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

Subham S Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. *Advances in Neural Information Processing Systems*, 31, 2018.

Mitchell Stern, William Chan, Jamie Kiros, and Jakob Uszkoreit. Insertion transformer: Flexible sequence generation via insertion operations. In Kamalika Chaudhuri and Ruslan Salakhutdinov (eds.), *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5976–5985. PMLR, 09–15 Jun 2019. URL https://proceedings.mlr.press/v97/stern19a.html.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Tong Wu, Zhihao Fan, Xiao Liu, Hai-Tao Zheng, Yeyun Gong, Jian Jiao, Juntao Li, Jian Guo, Nan Duan, Weizhu Chen, et al. Ar-diffusion: Auto-regressive diffusion model for text generation. *Advances in Neural Information Processing Systems*, 36:39957–39974, 2023.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7b: Diffusion large language models, 2025. *URL https://arxiv. org/abs/2508.15487*.

Runzhe Zhang, Letian Chen, Wenpeng Zhang, Zhouhan Lin, and Peilin Zhao. Flowlm: Few-step language modeling via diffusion-to-flow adaptation. *arXiv preprint arXiv:2605.20199*, 2026.

A Detailed Math Derivations

A.1 Detailed ELBO Derivation

We evaluate SimpLM as a latent-variable generative model by stopping the continuous trajectory at a near-clean time $t_{\text{clean}} < 1$ and treating $x_{t_{\text{clean}}}$ as a latent continuous state. Starting from

$$p(x_1) = \int p_\theta(x_1 \mid x_{t_{\text{clean}}})\, p(x_{t_{\text{clean}}})\, dx_{t_{\text{clean}}}, \tag{9}$$

we introduce the forward posterior $q(x_{t_{\text{clean}}} \mid x_1)$ and rewrite the marginal likelihood as

$$p(x_1) = \int q(x_{t_{\text{clean}}} \mid x_1) \frac{p_\theta(x_1 \mid x_{t_{\text{clean}}})\, p(x_{t_{\text{clean}}})}{q(x_{t_{\text{clean}}} \mid x_1)}\, dx_{t_{\text{clean}}}. \tag{10}$$

Applying Jensen's inequality yields the standard Evidence Lower Bound (ELBO):

$$\log p(x_1) \geq \mathbb{E}_{q(x_{t_{\text{clean}}}|x_1)} \left[\log p_\theta(x_1 \mid x_{t_{\text{clean}}}) + \log p(x_{t_{\text{clean}}}) - \log q(x_{t_{\text{clean}}} \mid x_1)\right]. \tag{11}$$

For our specific linear Gaussian flow defined over the vocabulary space of dimension V and sequence length L, the analytic forward posterior reduces to:

$$\log q(x_{t_{\text{clean}}} \mid x_1) = -\frac{VL}{2} \log(2\pi\sigma^2_{t_{\text{clean}}}) - \frac{1}{2\sigma^2_{t_{\text{clean}}}} \left\| x_{t_{\text{clean}}} - t_{\text{clean}} x_1 \right\|^2, \tag{12}$$

where the time-dependent variance is given by $\sigma_{t_{\text{clean}}} = (1 - t_{\text{clean}})\sigma$. The marginal prior $\log p(x_{t_{\text{clean}}})$ is computed rigorously via the continuous normalizing flow (CNF) ODE integration.

A.2 Multi-sample trick

To establish a tighter lower bound for SimpLM than the standard single-sample ELBO, we utilize a multi-sample trick. By drawing n independent samples $x^{(i)}_{t_{\text{clean}}}$, the final likelihood of the discrete sequence is rigorously bounded by:

$$\log p(x_1) \geq \mathbb{E}_{x_{t_{\text{clean}}}|x_1} \left[\log \left(\frac{1}{n} \sum_{i=1}^{n} \frac{p(x_1^{(i)} \mid x^{(i)}_{t_{\text{clean}}}) p(x^{(i)}_{t_{\text{clean}}})}{p(x^{(i)}_{t_{\text{clean}}} \mid x_1^{(i)})} \right)\right]. \tag{13}$$

A.3 Derivation of Vectorized Time Transport Equivalence

Here we provide the derivation for the instantaneous velocity $v_u(x_u)$ under an arbitrary differentiable schedule $\mathbf{f}(u)$, showing its equivalence to the underlying vectorized-time field.

Differentiating the scheduled path $x_u = (\mathbf{1} - \mathbf{f}(u)) \odot x_0 + \mathbf{f}(u) \odot x_1$ with respect to the scalar time u yields:

$$\frac{dx_u}{du} = \mathbf{f}'(u) \odot (x_1 - x_0).$$

The target vector field is the conditional expectation of this derivative:

$$\begin{aligned} v_u(x_u) &= \mathbb{E}\left[\frac{dx_u}{du} \,\middle|\, x_u \right] \\ &= \mathbb{E}\left[\mathbf{f}'(u) \odot (x_1 - x_0) \mid x_u\right] \\ &= \mathbf{f}'(u) \odot \mathbb{E}[x_1 - x_0 \mid x_u]. \end{aligned}$$

By definition, the scheduled state x_u is equivalent to the vectorized state $x_{\mathbf{t}}$ evaluated at $\mathbf{t} = \mathbf{f}(u)$. Substituting the local-time target vector field $v_{\mathbf{f}(u)}(x_u) = \mathbb{E}[x_1 - x_0 \mid x_u]$ into the equation gives the final equivalence:

$$v_u(x_u) = \mathbf{f}'(u) \odot v_{\mathbf{f}(u)}(x_u).$$

A.4 Derivation of the Analytic Logit Skip Connection

We provide the Bayesian derivation of the analytic clean-token posterior $p(x_1 \mid x_t)$ under our vocabulary-space continuous flow, which motivates the exact functional form of our logit skip connection.

Let V denote the vocabulary size. A clean token $i \in \{1, \dots, V\}$ is represented as a one-hot vector $x_1 = e_i \in \mathbb{R}^V$. The forward diffusion path conditioned on e_i is an isotropic Gaussian, meaning its density is proportional to the exponentiated scaled distance:

$$p(x_t \mid x_1 = e_i) = \mathcal{N}\left(x_t; te_i, (1-t)^2\sigma^2 I\right) \propto \exp\left(-\frac{\|x_t - te_i\|_2^2}{2(1-t)^2\sigma^2}\right). \tag{14}$$

By Bayes' theorem, the posterior probability over the discrete vocabulary is proportional to the joint probability:

$$p(x_1 = e_i \mid x_t) \propto p(x_t \mid x_1 = e_i)p(x_1 = e_i). \tag{15}$$

Taking the natural logarithm yields:

$$\log p(x_1 = e_i \mid x_t) = -\frac{\|x_t - te_i\|_2^2}{2(1-t)^2\sigma^2} + \log p(x_1 = e_i) + C, \tag{16}$$

where C absorbs all normalization factors independent of the token index i.

We expand the squared Euclidean distance as:

$$\|x_t - te_i\|_2^2 = \|x_t\|_2^2 - 2t\langle x_t, e_i\rangle + t^2\|e_i\|_2^2. \tag{17}$$

Because e_i is a one-hot vector, its squared norm is exactly $\|e_i\|_2^2 = 1$, and the inner product strictly extracts the i-th component of the continuous state, $\langle x_t, e_i\rangle = x_t^i$.

Crucially, because $\|x_t\|_2^2$ and t^2 do not depend on the specific token index i, they act as uniform baseline shifts across all logits. Since the softmax operation is invariant to uniform scalar addition, we can absorb these terms into the constant. The log-posterior simplifies to:

$$\log p(x_1 = e_i \mid x_t) = \frac{t}{(1-t)^2\sigma^2}x_t^i + \log p(x_1 = e_i) + \text{const}. \tag{18}$$

The term $\log p(x_1 = e_i)$ represents the complex contextual prior probability of token i, which is modeled by the residual logits of the Transformer backbone, $\ell_\theta^{\text{net}}(x_t, t)_i$.

The remaining term is the explicit geometric contribution derived directly from the Gaussian corruption process. Dropping the token-independent constants, this provides the exact mathematical formulation for our analytic skip connection:

$$\ell_i^{\text{skip}}(x_t, t) = \frac{t}{(1-t)^2\sigma^2}x_t^i. \tag{19}$$

B Training Hyperparameters

All models are initialized from the pre-trained Pythia-160M backbone (vocabulary size $V = 50,304$). Training is conducted across 4 NVIDIA A6000 GPUs. The detailed hyperparameter configuration is summarized below:

- **Optimizer**: AdamW
- **Learning Rate**: 3×10^{-5}
- **Weight Decay**: 0.0
- **Gradient Clipping**: 1.0
- **Batch Size**: 160
- **Sequence Length**: 512

Block-Size Curriculum Schedule:

- **Stage 1 (Block Size 1):** Trained for 2,000 steps.
- **Stage 2 (Block Size 4):** Trained for 23,000 steps.
- **Stage 3 (Block Size 32):** Trained for 25,000 steps.
- **Stage 4 (Block Size 32 + Self-Conditioning):** Trained for an additional 20,000 steps using a self-conditioning probability of $p_{sc} = 0.5$.

C ALGORITHMS

We provide the formal algorithms for training with our block-size curriculum (Algorithm 2), for conducting blockwise continuous inference (Algorithm 1), and for computing the multi-sample ELBO (Algorithm 3).

To natively support block-conditional generation, the model processes a concatenated sequence $h_t = [x_{\text{prefix}}\,;\, q_{\text{blk}}]$, comprising a clean context prefix and a noisy probability-simplex block. To preserve autoregressive prefix semantics while enabling parallel block generation, the Transformer operates under a hybrid attention mask $\mathcal{M}_{\text{hybrid}}$. Under $\mathcal{M}_{\text{hybrid}}$, tokens within x_{prefix} apply standard causal masking, whereas tokens within q_{blk} apply fully bidirectional masking and are permitted to attend to all prefix tokens.

Note the algorithms below show sampling random block index in entire sequence, but in practice for efficient training, we use tree structure attention to compute loss on all blocks in one sentence simultaneously.

Algorithm 1 SIMPLM Blockwise Inference (Euler Integration)

Require: Trained model Transformer$_\theta$, Prompt x_{prefix}, Block size B, Steps N, Noise σ

$x_0 \sim \mathcal{N}(0, \sigma^2 I_{B\times V})$
$x_t \leftarrow x_0$
$\hat{x}_{\text{blk}} \leftarrow \varnothing$
for $n = 0$ **to** $N-1$ **do**
  $t \leftarrow \frac{n}{N}$
  $\Delta t \leftarrow \frac{1}{N}$
  $\ell^{\text{skip}} \leftarrow \frac{t}{(1-t)^2\sigma^2} x_t$
  $q_{\text{blk}} \leftarrow \text{softmax}(\ell^{\text{skip}})$
  $h_t \leftarrow [x_{\text{prefix}}\,;\, q_{\text{blk}}]$
  $\ell^{\text{net}} \leftarrow \text{Transformer}_\theta(h_t, \hat{x}_{\text{blk}}, t, \mathcal{M}_{\text{hybrid}})$
  $p_\theta \leftarrow \text{softmax}(\ell^{\text{net}}_{\text{blk}} + \ell^{\text{skip}})$
  $v_t \leftarrow \frac{p_\theta - x_t}{1-t}$
  $x_t \leftarrow x_t + v_t \Delta t$
  $\hat{x}_{\text{blk}} \leftarrow p_\theta$
end for
return $\arg\max(x_t)$

Algorithm 2 Training SIMPLM with Curriculum and Self-Conditioning

Require: Backbone Transformer$_\theta$, Dataset $\mathcal{D}$, Noise scale σ, Self-cond. probability p_{sc}
Require: Curriculum schedule $\mathcal{C} = \{(B_1, S_1), (B_2, S_2), \dots\}$
Initialize parameters θ
for $(B, S) \in \mathcal{C}$ **do**
  for $s = 1$ **to** S **do**
    $X \sim \mathcal{D}$ ▷ Sample full discrete sequence of length L
    $k \sim \mathcal{U}(\{1, \dots, \lfloor L/B \rfloor\})$ ▷ Sample a random block index
    $x_{\text{prefix}} \leftarrow X_{<(k-1)B}$ ▷ Clean conditional prefix
    $x_{\text{blk}} \leftarrow X_{(k-1)B:kB}$ ▷ Target generation block
    $t \sim \mathcal{U}(0, 1)$
    $x_0 \sim \mathcal{N}(0, \sigma^2 I_{B \times V})$
    $x_t \leftarrow (1-t)x_0 + t x_{\text{blk}}$ ▷ Continuous interpolation on the block
    $\ell^{\text{skip}} \leftarrow \frac{t}{(1-t)^2\sigma^2} x_t$
    $q_{\text{blk}} \leftarrow \text{softmax}(\ell^{\text{skip}})$
    $h_t \leftarrow [x_{\text{prefix}}\,;\, q_{\text{blk}}]$ ▷ Concatenate prefix and active simplex block
    $z \sim \mathcal{U}(0, 1)$
    if $z < p_{sc}$ **then**
      $\ell_1^{\text{net}} \leftarrow \text{Transformer}_\theta(h_t, \varnothing, t, \mathcal{M}_{\text{hybrid}})$
      $\hat{x}_{\text{blk}} \leftarrow \text{StopGradient}\left(\text{softmax}(\ell_{1,\text{blk}}^{\text{net}} + \ell^{\text{skip}})\right)$
      $\ell_2^{\text{net}} \leftarrow \text{Transformer}_\theta(h_t, \hat{x}_{\text{blk}}, t, \mathcal{M}_{\text{hybrid}})$
      $\hat{p}_{\text{blk}} \leftarrow \text{softmax}(\ell_{2,\text{blk}}^{\text{net}} + \ell^{\text{skip}})$
    else
      $\ell^{\text{net}} \leftarrow \text{Transformer}_\theta(h_t, \varnothing, t, \mathcal{M}_{\text{hybrid}})$
      $\hat{p}_{\text{blk}} \leftarrow \text{softmax}(\ell_{\text{blk}}^{\text{net}} + \ell^{\text{skip}})$
    end if
    $\mathcal{L} \leftarrow \text{CrossEntropy}(\hat{p}_{\text{blk}}, x_{\text{blk}})$
    $\theta \leftarrow \theta - \eta \nabla_\theta \mathcal{L}$
  end for
end for

Algorithm 3 SIMPLM Multi-Sample ELBO Likelihood Estimation

Require: Full discrete sequence X of length L, Block size B, Samples n, Stopping time t_{clean}
Require: Trained model Transformer$_\theta$, ODE Solver

function $\Phi_\theta(x_{\text{prefix}}, x_{\text{blk}}, t)$
  $\ell^{\text{skip}} \leftarrow \frac{t}{(1-t)^2\sigma^2} x_{\text{blk}}$
  $h_t \leftarrow [x_{\text{prefix}}\,;\, \text{softmax}(\ell^{\text{skip}})]$
  $\ell^{\text{net}} \leftarrow \text{Transformer}_\theta(h_t, \varnothing, t, \mathcal{M}_{\text{hybrid}})$
  return $\text{softmax}(\ell^{\text{net}}_{\text{blk}} + \ell^{\text{skip}})$
end function
$\mathcal{L}_{\text{seq}} \leftarrow 0$ ▷ Initialize total sequence log-likelihood
$K \leftarrow \lfloor L/B \rfloor$ ▷ Total number of blocks
for $k = 1$ **to** K **do**
  $x_{\text{prefix}} \leftarrow X_{<(k-1)B}$
  $x_{\text{blk}} \leftarrow X_{(k-1)B:kB}$
  $W \leftarrow \mathbf{0} \in \mathbb{R}^n$
  for $i = 1$ **to** n **do**
    $x^{(i)}_{t_{\text{clean}}} \sim \mathcal{N}(t_{\text{clean}} x_{\text{blk}}, (1 - t_{\text{clean}})^2\sigma^2 I)$
    $\hat{p}_{\text{blk}} \leftarrow \Phi_\theta(x_{\text{prefix}}, x^{(i)}_{t_{\text{clean}}}, t_{\text{clean}})$
    $\log P_{\text{recon}} \leftarrow \log \hat{p}_{\text{blk}}(x_{\text{blk}})$
    $v_\theta(x, t) := \frac{\Phi_\theta(x_{\text{prefix}}, x, t) - x}{1-t}$
    $x^{(i)}_0, \Delta \log p \leftarrow \text{ODESolve}\left(x^{(i)}_{t_{\text{clean}}},\, 0,\, \int_{t_{\text{clean}}}^0 v_\theta(x,\tau)d\tau,\, \int_{t_{\text{clean}}}^0 \nabla_x \cdot v_\theta(x,\tau)d\tau\right)$
    $\log P_{\text{prior}} \leftarrow \log \mathcal{N}(x^{(i)}_0; 0, \sigma^2 I) + \Delta \log p$
    $\log Q_{\text{fwd}} \leftarrow \log \mathcal{N}(x^{(i)}_{t_{\text{clean}}}; t_{\text{clean}} x_{\text{blk}}, (1 - t_{\text{clean}})^2\sigma^2 I)$
    $W[i] \leftarrow \log P_{\text{recon}} + \log P_{\text{prior}} - \log Q_{\text{fwd}}$
  end for
  $\mathcal{L}_{\text{seq}} \leftarrow \mathcal{L}_{\text{seq}} + \text{LogSumExp}(W) - \log(n)$ ▷ Accumulate block log-likelihood
end for
return $-\frac{1}{L}\mathcal{L}_{\text{seq}}$ ▷ Return Mean Negative Log-Likelihood (NLL)

D GENERATED SAMPLES

Here we present qualitative samples of generated text. Black color indicates prompt, blue color indicates generated text.

FlowLLM, Block Size: 128, Sampling Steps: 256 PPL=13.66, Entropy=4.05

What difference does policy make? Craig Morris says that a comparison of the low prices for installed solar arrays in Germany with more expensive arrays in the US is a good way to start answering that question.
Typically, whenever the Anglo press speaks of German policy support for photovoltaics (PV), we read of things like lavish subsidies (Nature) and generous feed-in tariffs (CNN). So how does that sound when compared with the chart below?
Here, we see that the price of a completely installed solar array has been and continues to be considerably cheaper in Germany than in the US. The gap seems to have widened recently with the advent of solar PV in 2008. The price of large-scale photovoltaics in the US has subsequently only raised sharply since 2008/07.
But there are very obvious economic differences between large-scale solar PV and large-mounted plants in Germany and the US. Large-scale photovoltaics are about eight times more efficient PVs in Germany than multi-scale plants in China.
Part of the difference lies is that larger-scale photovoltaics make nearly eight times more expensive PV units in the US than multi-scale plants in China. Today we perceive the fact that Chinese get

FlowLLM, Block Size: 128, Sampling Steps: 256 PPL=19.44, Entropy=4.25

Lest anyone think that Marvel is the only film studio with multiyear plans to keep moviegoers thigh-deep in capes and spandex, Warner Bros. CEO Kevin Tsujihara announced his studio's upcoming slate of DC Comics adapted properties, a plan that stretches out to 2020. And so the eternal war between DC and Marvel — which both companies pretend isn't a thing but totally is a thing — continues apace.
Despite the critical and box office success of Christopher Nolan's Dark Knight trilogy, one gets the sense that Warners was caught flat-footed by the stupefiling big Warner Bros Studios to go forward with a movie based on a beloved, cinematic origin. In an back-and-forth conversation with Tsujihara, the studio has been struggling with at least three projects for a long time, and he said that it hopes to have their films finished at least next year.
"I don't know if we're going to do something weird, including Warner Bros. or something else," Tsujihara reportedly told the Hollywood Reporter in a statement. "We're not sure what we're doing, but we're trying to have an agreement.

FlowLLM, Block Size: 128, Sampling Steps: 64 PPL=77.82, Entropy=4.29

What difference does policy make? Craig Morris says that a comparison of the low prices for installed solar arrays in Germany with more expensive arrays in the US is a good way to start answering that question.
Typically, whenever the Anglo press speaks of German policy support for photovoltaics (PV), we read of things like lavish subsidies (Nature) and generous feed-in tariffs (CNN). So how does that sound when compared with the chart below?
Here, we see that the price of a completely installed solar array has been and continues to be considerably cheaper in Germany than in the US. The gap seems to have widened because the relative growth of installed PV in Germany means that in the case of no parovoltaics in the US, lower PV installation may seem to be efficient.
If there are PV less expensive than the United States, what does that mean?
The US has a very little overall solar for boovoltaics as compared to very low in 's. Here is this market. A couple % of the 800MW homes was installed using PV in Germany compared with lithiumnoltaics more (73 times more expensive PVSH's) than the US and Germany. In both, it is 94% expensive (£37

E 2D Visualization Geneation Details

To build the 2D visualization of the vectorized time-flow model, we created a ground-truth distribution with two spiral arms. The plane is divided into 120 x 120 grids, and time is divided into 20 bins. We then enumerate each x_0, x_1 pair and accumulate velocity estimates along the transport path for the corresponding space-grid and time-bin pair. Finally, we use forward Euler to generate the flow path for each x_0.

F Full Results of Generative Perplexity and Unigram Entropy

Table 2 reports the full generative evaluation results with 95% confidence intervals. These results correspond to the mean-only results shown in Table 1, where confidence intervals are omitted for readability.

Table 2: Generative perplexity and unigram entropy across prefix/suffix settings. It reports the full generative evaluation results with 95% confidence intervals.

Prefix	Suffix	Base Euclidean		+Simplex Representation		+Skip connection		+Simplex Representation & Skip connection		+Simplex Rep. & Skip (w/o AR init)	
		PPL ↓	Entropy ↑	PPL ↓	Entropy ↑	PPL ↓	Entropy ↑	PPL ↓	Entropy ↑	PPL ↓	Entropy ↑
128	128	126.8 ± 4.0	4.20 ± 0.01	616.3 ± 39.4	4.36 ± 0.01	45.0 ± 2.3	4.03 ± 0.01	45.1 ± 2.7	4.01 ± 0.02	162.6 ± 9.4	4.12 ± 0.02
128	256	105.2 ± 2.8	4.56 ± 0.02	757.1 ± 42.6	4.89 ± 0.01	28.9 ± 1.4	4.18 ± 0.02	28.3 ± 1.8	4.17 ± 0.02	108.3 ± 6.1	4.39 ± 0.02
256	128	116.9 ± 3.8	4.21 ± 0.01	548.4 ± 31.1	.36 ± 0.01	43.5 ± 2.0	4.04 ± 0.01	44.4 ± 2.5	4.01 ± 0.02	157.3 ± 9.1	4.12 ± 0.02
256	256	105.5 ± 2.9	4.57 ± 0.01	659.6 ± 37.5	4.87 ± 0.02	31.5 ± 1.5	4.22 ± 0.02	32.4 ± 1.9	4.21 ± 0.02	111.5 ± 6.4	4.40 ± 0.02