

Masked Language Flow Models

Iskander Azangulov, Kianoosh Ashouritaklimi*, Leo Zhang*, Simon Vary, Patrick Rebeschini

Department of Statistics, University of Oxford, *Equal contribution

Abstract *Masked Diffusion Models* (MDMs) promise fast, parallel language generation, but their reverse transition factorises across token positions—an approximation that breaks down in the few-step sampling regime where parallel generation ought to provide the greatest efficiency gains. *Flow Language Models* (FLMs) sidestep this limitation by learning a continuous flow that transports noise toward clean sequences represented in Euclidean space, inducing a flow map that can be distilled for single-step generation. However, this makes complex tasks requiring multi-step reasoning problematic for FLMs, as FLMs are forced to decode every token during generation. To address this, we introduce **Masked Language Flow Models (MLFMs)**, which incorporate masking into FLMs using a continuous stochastic interpolant to bridge partially masked and clean sequences. This design enables conditional generation via continuous flows and allows pretrained MDMs to be converted into MLFMs through a simple, lightweight adaptation. Leveraging this flexibility, we propose a novel sampler that alternates continuous denoising with the discrete unmasking of confident tokens to better support multi-step reasoning. We evaluate our approach on GSM8K and MT-Bench and find, for the first time, that flow-based language models can be scaled to solve downstream reasoning and instruction-following tasks.

Code is available at: github.com/imbirik/mlfm.

1 Introduction

Autoregressive Models (ARMs) have driven much of the recent progress in language modelling by framing language generation as next-token prediction (Brown et al., 2020). However, their left-to-right factorisation of the joint distribution makes this process inherently sequential, as each token must be conditioned on all preceding ones. Consequently, inference costs scale linearly with output length, creating a bottleneck for long sequences. This limitation has motivated growing interest in *Masked Diffusion Models* (MDMs) (Austin et al., 2021; Campbell et al., 2022; Lou et al., 2024; Shi et al., 2024; Sahoo et al., 2024) which replace sequential generation with the parallel decoding of masked tokens.

Although MDMs have demonstrated strong performance across language modelling and downstream tasks (Nie et al., 2025a,b), they typically rely on a factorised reverse transition across masked positions, making inference tractable in the combinatorially large discrete state space. This approximation is accurate in the infinitesimal-step limit but becomes highly restrictive in the few-step regime (Deschenaux and Gulcehre, 2024), where each step must independently decode many masked tokens at once—ignoring the dependencies amongst them. Consequently, the very regimes where parallel decoding promises the greatest speedups (Dieleman, 2023; Zheng et al., 2024) are precisely those where this independence assumption most severely compromises generation quality.

Flow Language Models (FLMs) (Roos et al., 2026; Lee et al., 2026; Potapchik et al., 2026; Davis et al., 2026; Chen et al., 2026; Hu et al., 2026) address this issue by moving from a discrete state space into a continuous one. Specifically, FLMs learn a flow (Song et al., 2021; Lipman et al., 2023; Albergo et al., 2025) transporting noise to continuous embeddings of token sequences, allowing latent states to evolve jointly across token positions. A significant advantage of FLMs is that they also induce a flow map which can be

distilled to support few-step and even one-step generation. However, collapsing generation to a single flow map may be too restrictive for language tasks that require multi-step reasoning. Indeed, many language tasks benefit from iterative generation where partially completed intermediate states provide context for subsequent refinements (Ghazvininejad et al., 2019; Nye et al., 2021; Wei et al., 2022).

To address this limitation, we propose **Masked Language Flow Models (MLFMs)**. MLFMs incorporate masking from MDMs into FLMs by using a Brownian bridge as a stochastic interpolant connecting embedded partially masked sequences with embedded clean sequences. As a result, MLFMs retain the exact any-position conditioning structure of MDMs while learning a coupled continuous flow for generating clean tokens at masked positions. Moreover, this special structure of MLFMs gives us a natural strategy for training and inference. For training, we propose adapting pretrained MDMs into MLFMs: this is possible because, at the boundary of the Brownian bridge, the stochastic interpolant reduces to an embedded partially masked sequence, matching the setup of MDMs. This allows us to warm-start MLFM training from large pretrained MDMs, greatly reducing the compute required to train our models.

For inference, we leverage the flexibility MLFMs provide through any-position conditional generation. Specifically, we introduce a novel sampling scheme that combines a new approach to classifier-free guidance with the online promotion of confident tokens. Under this guidance method, clean tokens are noised in the reference velocity to isolate their contribution. Concurrently, if the posterior mode at a specific token position reaches a probability of at least $1 - \epsilon$, we immediately commit to that token and substitute in its clean embedding. By promoting these resolved tokens early, they instantly become useful context for subsequent generation steps, rather than remaining noisy until the final step.

Empirically, we adapt a pretrained MDM with 1028M parameters from Nie et al. (2025a) into an MLFM and evaluate the resulting model on GSM8K (Cobbe et al., 2021) and MT-Bench (Zheng et al., 2023). MLFM improves over SMDM and similarly sized AR baselines on MT-Bench and obtains encouraging performance on GSM8K, demonstrating its ability to handle both instruction-following and reasoning tasks. Across these benchmarks, we also find that our novel sampling scheme provides significant improvements. To the best of our knowledge, this is the first time that flow-based language models have been scaled to downstream reasoning and instruction-following tasks.

2 Background

In this section, we provide an overview of the background required for our approach. Section 2.1 discusses Masked Diffusion Models and Section 2.2 discusses Flow Language Models, in particular LangFlow (Chen et al., 2026) which we take as the basis for our approach.

Notation. Throughout, we let \mathcal{V} denote a vocabulary of tokens, $|\mathcal{V}|$ the vocabulary size, and L the sequence length. We will identify elements in \mathcal{V} with the natural numbers: $\{1, \dots, |\mathcal{V}|\}$. A (clean) sequence $X \sim p$, where p denotes our data distribution, is written as $X = (X^1, \dots, X^L) \in \mathcal{V}^L$. We use $s \in [0, 1]$ for the masking probability and $t \in [0, 1]$ for the continuous noising time.

2.1 Masked Diffusion Models

Masked Diffusion Models (MDMs) construct a non-autoregressive generative model for p by introducing a special *mask* token [MASK] to the vocabulary \mathcal{V} and defining a continuous-time Markov chain (CTMC) (Del Moral and Penev, 2017) transporting p to the Dirac measure on fully masked sequences. The forward process $X_s \sim p_{s|0}(X_s|X)$ of the CTMC at the masking probability $s \in [0, 1]$ and $X \sim p$ is given by the following sampling procedure:

$$X_s^\ell = \begin{cases} X^\ell, & \text{if } B^\ell = 0, \\ [\text{MASK}], & \text{if } B^\ell = 1, \end{cases} \quad \text{where } B^\ell \stackrel{\text{i.i.d.}}{\sim} \text{Bernoulli}(s). \quad (1)$$

We let $\mathcal{M}_s = \{\ell : X_s^\ell = [\text{MASK}]\}$ denote the positions of mask tokens in X_s and $\mathcal{U}_s = \{\ell : X_s^\ell \neq [\text{MASK}]\}$ the positions of clean tokens in X_s .

We can generate approximate samples from p by simulating the backward process of the CTMC. This involves gradually unmasking tokens to reveal clean tokens which are then fixed for later steps. This relies on access to the ground-truth factorised posterior $p_{\text{data}}^\ell(X^\ell | X_s^{\mathcal{U}_s})$ of clean tokens given unmasked tokens (Zheng et al., 2024; Ou et al., 2025) where $X_s^{\mathcal{U}_s}$ denotes the subcollection of X_s consisting of clean tokens. This can be learned from data via the following likelihood-based objective:

$$\mathcal{L}_{\text{MDM}}(\theta) = \mathbb{E}_{X \sim p, X_s \sim p_{s|0}(\cdot | X), s \sim U(0,1)} \left[\frac{1}{s} \sum_{\ell \in \mathcal{M}_s} -\log p_\theta^\ell(X^\ell | X_s^{\mathcal{U}_s}) \right], \quad (2)$$

where we parametrise p_θ^ℓ with a neural network to approximate p_{data}^ℓ and $U(0,1)$ denotes the uniform distribution over $[0, 1]$.

During inference, MDMs can unmask multiple tokens in parallel in a single step. This is most effective when the tokens to be unmasked are close to conditionally independent given the current masked state, due to the factorised structure of p_θ^ℓ . However, this is only typically true for a small subset of tokens as increasing the number of tokens to be unmasked increases the likelihood of strong dependencies between tokens. As a result, MDMs still require many refinement steps for high quality samples despite their parallel decoding structure.

2.2 Flow Language Models

We consider the *LangFlow* (Chen et al., 2026) approach to *Flow Language Models* (FLMs) which operates within a learnt embedding space instead of one-hot embeddings.

Let H be the embedding dimension. For each token $a \in \mathcal{V}$ in the vocabulary, we represent the token with its corresponding embedding vector $E_a \in \mathbb{R}^H$ and we let $E \in \mathbb{R}^{|\mathcal{V}| \times H}$ denote the overall embedding matrix. Moreover, for a sequence $X \sim p$, we represent X in Euclidean space by the embedding:

$$z_1 = (E_{X^1}, \dots, E_{X^L}) \in \mathbb{R}^{L \times H}.$$

With this continuous representation z_1 of discrete sequences, we can construct a generative model for p by considering the Gaussian-based stochastic interpolant:

$$z_t = tz_1 + (1-t)\epsilon, \quad (3)$$

where $\epsilon \sim \mathcal{N}(0, I_{L \times H})$ and $t \in [0, 1]$. We note that z_0 is Gaussian noise and z_1 is an embedded sequence from p . To sample from the flow induced by this stochastic interpolant, we parametrise the denoiser p_ϕ which approximates the factorised posterior over clean tokens:

$$p_\phi(\cdot | z_t, t) \in \Delta(\mathcal{V})^L \subset \mathbb{R}^{L \times |\mathcal{V}|}.$$

This is trained with the following cross-entropy objective:

$$\mathcal{L}_{\text{FLM}}(\phi) = \mathbb{E}_{t \sim \pi_t, X, \epsilon} \left[\frac{1}{L} \sum_{\ell=1}^L -\log p_\phi^\ell(X^\ell | z_t, t) \right], \quad (4)$$

where π_t denotes some distribution over $[0, 1]$. Although the supervision target X^ℓ is discrete, the learned denoiser induces a continuous endpoint estimate:

$$\widehat{z}_{1,\phi}(z_t, t) := P_\phi(z_t, t)E, \quad \text{where} \quad P_\phi(z_t, t) = p_\phi(\cdot | z_t, t) \in \mathbb{R}^{L \times |\mathcal{V}|}, \quad (5)$$

and $\widehat{z}_{1,\phi}(z_t, t)$ estimates the posterior mean embedding $\mathbb{E}[z_1 | z_t, t]$. Additionally, $\widehat{z}_{1,\phi}(z_t, t)$ allows us to estimate the velocity field for the flow:

$$v_\phi(z_t, t) := \frac{\widehat{z}_{1,\phi}(z_t, t) - z_t}{1-t}, \quad t < 1. \quad (6)$$

Therefore, we can generate approximate samples from p by integrating the ODE:

$$\frac{d}{dt}z_t = v_\phi(z_t, t).$$

In practice, the ODE is discretized over a finite time grid, and the final token sequence is obtained by decoding the denoiser probabilities from p_ϕ in vocabulary space.

Continuous flows are attractive because they define a joint trajectory from noise to clean sequences. This trajectory can be distilled into a direct flow map or a consistency-style solver (Song et al., 2023; Boffi et al., 2026), reducing the number of generation steps. However, an exact one-shot map is an overly demanding object for language models—i.e. some problems (such as maths or coding) may naturally require iterative reasoning in which intermediate commitments are used as context for later decisions.

3 Masked Language Flow Models

We introduce the core framework of MLFMs in Section 3.1. In Section 3.2, we show how pretrained MDMs can naturally be adapted to MLFMs, including the architectural changes required for this adaptation. Finally, Section 3.3 describes how we can apply supervised fine-tuning on instruction-response data to MLFMs, an aspect that, to the best of our knowledge, has not been explored in prior work on FLMs.

3.1 Framework

We retain the use of mask tokens and token embeddings E from MDMs and FLMs. We begin by defining a forward process $z_{s,t} \in \mathbb{R}^{L \times H}$ in Euclidean space. This process is parametrised by $(s, t) \in [0, 1]^2$ where s denotes the masking probability and t denotes the noising time.

Let $m = E_{[\text{MASK}]}$ denote the embedding of the [MASK] token. For $X \sim p$, $s \sim \pi_s$, $t \sim \pi_t$, where π_s, π_t are distributions on $[0, 1]$, we sample a partially masked sequence $X_s \sim p_{s|0}(\cdot|X)$ as in the MDM forward process (1) and we construct $z_{s,t} \in \mathbb{R}^{L \times H}$ as follows. For all positions $\ell \in \mathcal{U}_s$ corresponding to clean tokens in X_s , we fix $z_{s,t}^\ell$ to be the clean embedding:

$$z_{s,t}^\ell = E_{X^\ell}, \quad \forall \ell \in \mathcal{U}_s, t \in [0, 1].$$

For $\ell \in \mathcal{M}_s$, we construct $z_{s,t}^\ell$ via the stochastic interpolant formed by the Brownian bridge connecting m and E_{X^ℓ} :

$$z_{s,t}^\ell | X^\ell \sim \mathcal{N}\left((1-t)m + tE_{X^\ell}, \sigma^2 t(1-t)I_H\right), \quad \forall \ell \in \mathcal{M}_s, t \in [0, 1], \quad (7)$$

where $\sigma > 0$ is some choice of noise scale on masked positions. Since the variance vanishes at both endpoints, we see that the stochastic interpolant satisfies $z_{s,0}^\ell = m$ and $z_{s,1}^\ell = E_{X^\ell}$ almost surely. Thus $z_{s,0}$ recovers the embedding of the partially masked sequence X_s and $z_{s,1}$ recovers the embedding of X . Additionally, for $t \in (0, 1)$, the masked positions are modelled as noisy continuous states that provide partial information about their underlying clean token embeddings.

Following the same reasoning as in Chen et al. (2026), we sample from the flow induced by this stochastic interpolant by parametrising the MLFM denoiser p_θ to output token probabilities from the factorised posterior over clean tokens:

$$p_\theta(\cdot | z_{s,t}, t) \in \Delta(\mathcal{V})^L \subset \mathbb{R}^{L \times |\mathcal{V}|}.$$

This is trained via cross-entropy on only *masked* subsets of tokens:

$$\mathcal{L}_{\text{MLFM}}(\theta) = \mathbb{E}_{s,t,X,\epsilon} \left[\frac{1}{|\mathcal{M}_s|} \sum_{\ell \in \mathcal{M}_s} -\log p_\theta^\ell(X^\ell | z_{s,t}, t) \right]. \quad (8)$$

As in Chen et al. (2026), we find that the choice of π_t is critical for performance. We therefore follow their entropy-based schedule, with a small adaptation to our setting. Further details are provided in Appendix A, in the context of the experimental setup in Section 5.

For sampling, we have that p_θ induces an estimate of the posterior mean embedding

$$\widehat{z}_{1,\theta}(z_{s,t}, t) := P_\theta(z_{s,t}, t)E, \quad P_\theta(z_{s,t}, t) = p_\theta^\ell(\cdot \mid z_{s,t}, t), \quad (9)$$

which can then be used to estimate the velocity field of the flow:

$$v_\theta^\ell(z_{s,t}, t) := \begin{cases} \frac{\widehat{z}_{1,\theta}^\ell(z_{s,t}, t) - z_{s,t}^\ell}{1-t}, & \ell \in \mathcal{M}_s, \\ 0, & \ell \in \mathcal{U}_s, \end{cases} \quad t < 1. \quad (10)$$

With this, we can generate samples by integrating the ODE defined by v_θ . This evolves only the masked positions, while the positions in \mathcal{U}_s remain clamped to their clean embeddings. Note that the incorporation of masks naturally enables *conditional* generation in MLFMs, which has received limited attention in prior work on FLMS. Moreover, this additional flexibility allows us to design more advanced sampling schemes for MLFMs, as discussed in Section 4.

3.2 MDMs to MLFMs

It is easy to see that we can view the MLFM loss as a continuous extension of the MDM loss: at the endpoint $t = 0$, every unresolved position $\ell \in \mathcal{M}_s$ is represented exactly by the mask embedding m , while every observed position $\ell \in \mathcal{U}_s$ is kept at its clean embedding. Thus, $z_{s,0}$ contains the same information as the partially masked sequence X_s . Consequently, the cross-entropy objective $\mathcal{L}_{\text{MLFM}}$ at $t = 0$ has the same minimiser as the MDM objective \mathcal{L}_{MDM} : in both cases, the optimal predictor is the factorised posterior distribution of each masked token given the clean tokens. We formalise this endpoint equivalence in the following proposition.

Proposition 3.1 (Mask-endpoint equivalence). *Assume that the unmasked token identities are either provided directly or are recoverable from their embeddings under E . At the mask endpoint $t = 0$, the MLFM prediction problem coincides with the MDM prediction problem: for any $\ell \in \mathcal{M}_s$ and $a \in \mathcal{V}$,*

$$\mathbb{P}(X^\ell = a \mid z_{s,0}) = \mathbb{P}(X^\ell = a \mid X_s^{\mathcal{U}_s}, \mathcal{M}_s) = \mathbb{P}(X^\ell = a \mid X_s). \quad (11)$$

Proof. See Appendix E.1. □

This observation motivates adapting pretrained MDMs to MLFMs: because a pretrained MDM has already learned the denoising problem at the endpoint $z_{s,0}$, it provides a strong initialisation for MLFM training and avoids the cost of training a capable MLFM from scratch. Concretely, we adapt a pretrained bidirectional-transformer from an MDM for MLFM training by adapting three main components in the architecture:

- i) The MDM token embedding layer $E \in \mathbb{R}^{|\mathcal{V}| \times H}$, which maps tokens into the embedding space, is used as the MLFM embedding matrix and is kept fixed during adaptation, similar to Hu et al. (2026).
- ii) The MDM transformer blocks initialise the MLFM denoising transformer and are frozen. We augment the normalisations in these blocks with AdaLN (Nie et al., 2025a) to condition on the continuous corruption time t , and attach LoRA adapters (Hu et al., 2022) to the linear layers to adapt the pretrained transformer to work with continuous corrupted embeddings.
- iii) The MDM head which maps the final activations to vocabulary logits is adapted with its own LoRA adapter.

With these architectural changes, we can continue training the pretrained MDM under the MLFM objective $\mathcal{L}_{\text{MLFM}}(\theta)$, thereby adapting it into an MLFM. We refer to this MDM-to-MLFM training procedure as *adaptation* and we provide a summary in Algorithm 3.

3.3 Supervised Fine-Tuning

In this section, we describe how to perform supervised fine-tuning (SFT) with MLFMs. Let $\mathcal{D}_{\text{FT}} = \{(p_i, a_i)\}_{i=1}^n$ denote a SFT dataset, where p_i is a prompt and a_i is the corresponding target answer. After tokenisation, let $X = X(p_i, a_i)$ denote the sequence obtained from the concatenation of the prompt and answer in a suitable format and let \mathcal{P} denote the positions of the prompt and \mathcal{A} the positions of the answer in X . We fine-tune our MLFM p_θ with the below cross-entropy objective:

$$\mathcal{L}_{\text{FT}}(\theta) = \mathbb{E}_{(p,a) \sim \mathcal{D}_{\text{FT}}, s \sim \pi_s, t \sim \pi_t, z_{s,t}} \left[\frac{1}{|\mathcal{M}_s|} \sum_{\ell \in \mathcal{M}_s} -\log p_\theta^\ell(X^\ell \mid z_{s,t}, t) \right]. \quad (12)$$

For a sequence X constructed from (p, a) , we form $z_{s,t}$ as follows. With probability α , we mask all answer positions in \mathcal{A} , and with probability $1 - \alpha$ we instead sample a subset of answer positions by applying $p_{s|0}$ only to \mathcal{A} . The prompt positions \mathcal{P} are always kept clean to encourage conditional generation from the prompt. After the masked positions \mathcal{M}_s are sampled, we apply the same noising process (7) at time t to these positions. We provide a summary of our SFT procedure in Algorithm 4.

4 Sampling from MLFMs

In this section, we design a sampling scheme that exploits the additional flexibility of MLFMs for conditioning and guidance. The main idea behind our sampler is that high-confidence token predictions are more useful as clean context than as noisy latent states. In a flow-only sampler, each generated position remains latent until the final decoding step, even when its posterior distribution has already concentrated on a single token. MLFMs can use such positions more effectively: once a token becomes highly probable, we *promote* it to the observed-token state, so that later denoising steps condition on its clean embedding rather than on a corrupted representation of the same position.

We can naturally combine this strategy with a variant of classifier-free guidance (Ho and Salimans, 2022). Specifically, we compare two predicted vector fields: a guided field, which conditions on the promoted tokens as clean observed context, and a reference field, which uses the same tokens and positions but corrupts their embeddings to the current time. The difference between these fields isolates the effect of clean context on the model’s predicted direction. We then sample from the guided flow while promoting high-confidence tokens online, yielding a single sampling procedure that combines FLM sampling with MDM-style unmasking.

We begin by recalling the DDPM sampler in Section 4.1, then introduce our variant of classifier-free guidance (Section 4.2) and online promotion strategy (Section 4.3).

4.1 DDPM Sampler

We first describe the standard DDPM sampler (Ho et al., 2020) for conditional generation. Let $\mathcal{U}_0 \subseteq [L]$ be the unmasked context positions and let $\mathcal{M}_0 = [L] \setminus \mathcal{U}_0$ be the masked positions we want to generate. The sampler is initialised at the masked endpoint z_0^ℓ :

$$z_0^\ell = \begin{cases} E_{X^\ell}, & \ell \in \mathcal{U}_0, \\ m, & \ell \in \mathcal{M}_0. \end{cases}$$

During sampling, the context positions remain fixed, so that $z_t^\ell = E_{X^\ell}$ for $\ell \in \mathcal{U}_0$, while the masked positions in \mathcal{M}_0 are jointly denoised by simulating the SDE

$$dz_t^\ell = v_\theta^\ell(z_{s,t}, t) dt + \sigma dW_t^\ell, \quad \ell \in \mathcal{M}_0.$$

Algorithm 1 Context Corrupted Classifier-Free Guidance (CCFG)

Require: State z_t , time t , clean token positions \mathcal{U}_0 , clean tokens $X^{\mathcal{U}_0}$, guidance scale w .

- 1: Initialize $z_{t,\text{corr}} \leftarrow z_t$.
 - 2: **for** $\ell \in \mathcal{U}_0$ **do**
 - 3: Draw $\tilde{z}_t^\ell \mid X^\ell$ from the corruption marginal in (7).
 - 4: Set $z_{t,\text{corr}}^\ell \leftarrow \tilde{z}_t^\ell$.
 - 5: **end for**
 - 6: **return** $v_w = v_\theta(z_t, t) + w(v_\theta(z_t, t) - v_\theta(z_{t,\text{corr}}, t))$.
-

We discretize this process over a mesh $0 = t_0 < t_1 < \dots < t_N = 1$. Between t_i and t_{i+1} , the process is approximated using the DDPM transition

$$z_{t_{i+1}}^\ell \mid z_{t_i} \sim \mathcal{N}\left(\frac{1-t_{i+1}}{1-t_i}z_{t_i}^\ell + \frac{t_{i+1}-t_i}{1-t_i}\hat{z}_1^\ell(z_{t_i}, t_i), \sigma^2\frac{(t_{i+1}-t_i)(1-t_{i+1})}{1-t_i}I_H\right), \quad (13)$$

which samples from the Brownian bridge connecting z_{t_i} to the current endpoint estimate $\hat{z}_1(z_{t_i}, t_i)$, given by

$$\hat{z}_1(z_{t_i}, t_i) = z_{t_i} + (1-t_i)v_\theta(z_{t_i}, t_i).$$

Finally, at $t_N = 1$, each generated position ℓ is decoded by sampling $X^\ell \sim p_\theta^\ell(\cdot \mid z_{t_N}, t_N)$, yielding the sample X .

4.2 Classifier-Free Guidance With Corrupted Context

We first describe our variant of classifier-free guidance mentioned earlier. Let z_t be the state at time t . For $\ell \in \mathcal{U}_0$, the observed token is represented, as before, by its clean embedding,

$$z_t^\ell = E_{X^\ell}, \quad \ell \in \mathcal{U}_0.$$

We construct a corrupted-context reference state $z_{t,\text{corr}}$ by applying the forward corruption marginal in (7) only to context positions, while leaving all other positions unchanged:

$$z_{t,\text{corr}}^\ell = \begin{cases} \tilde{z}_t^\ell, & \ell \in \mathcal{U}_0, \\ z_t^\ell, & \ell \notin \mathcal{U}_0, \end{cases}$$

where, for $\ell \in \mathcal{U}_0$,

$$\tilde{z}_t^\ell \mid X^\ell \sim \mathcal{N}\left((1-t)m + tE_{X^\ell}, \quad \sigma^2t(1-t)I_H\right).$$

Let v_θ denote the MLFM velocity defined in (10). We define context-corrupted guidance with scale w by

$$v_w(z_t, t) = v_\theta(z_t, t) + w(v_\theta(z_t, t) - v_\theta(z_{t,\text{corr}}, t)).$$

The difference term captures the effect of clean context on the predicted velocity: both model calls use the same tokens at the same positions, but only $v_\theta(z_t, t)$ observes the context through exact clean embeddings. In effect, this encourages the sampler to follow directions that are specific to the clean-context prediction, rather than directions that persist when the context is corrupted. Note that both z_t and $z_{t,\text{corr}}$ remain on the support of the forward process. We refer to this sampling strategy as *context-corrupted classifier-free guidance* (CCFG) and summarise it in Algorithm 1.

Algorithm 2 CCFG with Online Token Promotion (CCFG w/ OTP)

Require: Prompt tokens $X^{\mathcal{U}_0}$, unresolved positions \mathcal{M}_0 , mesh $0 = t_0 < \dots < t_N = 1$, guidance scale w , tolerance ε .

- 1: Initialize z_{t_0} with clean embeddings on \mathcal{U}_0 and mask embeddings on \mathcal{M}_0 .
 - 2: **for** $i = 0, \dots, N - 1$ **do**
 - 3: Set $\widehat{X}_i^\ell = \arg \max_{a \in \mathcal{V}} p_\theta^\ell(a | z_{t_i}, t_i)$ for $\ell \in \mathcal{M}_i$.
 - 4: Promote $\mathcal{P}_i = \{\ell \in \mathcal{M}_i : p_\theta^\ell(\widehat{X}_i^\ell | z_{t_i}, t_i) \geq 1 - \varepsilon\}$.
 - 5: For $\ell \in \mathcal{P}_i$, set $X^\ell = \widehat{X}_i^\ell$ and $z_{t_i}^\ell = E_{X^\ell}$.
 - 6: Set $\mathcal{U}_{i+1} = \mathcal{U}_i \cup \mathcal{P}_i$ and $\mathcal{M}_{i+1} = \mathcal{M}_i \setminus \mathcal{P}_i$.
 - 7: Compute guided velocity v_w using Algorithm 1.
 - 8: Make one DDPM step (13) on \mathcal{M}_{i+1} using v_w to obtain $z_{t_{i+1}}$.
 - 9: **end for**
 - 10: Decode any remaining unresolved positions and **return** X .
-

4.3 Online Token Promotion

The standard DDPM sampler, as well as CCFG, treat the prompt as clean context but keep every generated position latent until the final decoding step. This can be inefficient as different positions resolve at different times: indeed, positions adjacent to the prompt, deterministic formatting tokens, and padding tokens in short answers often have sharply peaked posteriors well before $t = 1$. If such positions remain latent, however, later model calls still observe them through corrupted representations rather than through the clean embeddings of the predicted tokens.

We therefore *promote* high-confidence positions online, fixing a token to its clean embedding as soon as the model is sufficiently confident in it. Formally, let \mathcal{U}_i and $\mathcal{M}_i = [L] \setminus \mathcal{U}_i$ denote the set of unmasked, context and masked positions at step i , respectively. For each masked position, the posterior $p_\theta(\cdot | z_{t_i}, t_i)$ gives the predicted token

$$\widehat{X}_i^\ell = \arg \max_{a \in \mathcal{V}} p_\theta^\ell(a | z_{t_i}, t_i),$$

and, given a tolerance $\varepsilon > 0$, we promote the positions

$$\mathcal{P}_i = \left\{ \ell \in \mathcal{M}_i : p_\theta^\ell(\widehat{X}_i^\ell | z_{t_i}, t_i) \geq 1 - \varepsilon \right\}.$$

For each $\ell \in \mathcal{P}_i$ we set $X^\ell = \widehat{X}_i^\ell$ and fix $z_{t_i}^\ell = E_{X^\ell}$. We then apply the DDPM transition (13) only to the still-masked positions, holding all positions in $\mathcal{U}_i \cup \mathcal{P}_i$ fixed, and update the sets \mathcal{U}_i and \mathcal{M}_i :

$$\mathcal{U}_{i+1} = \mathcal{U}_i \cup \mathcal{P}_i, \quad \mathcal{M}_{i+1} = \mathcal{M}_i \setminus \mathcal{P}_i.$$

The sampler terminates once $t_N = 1$ is reached or all positions have been promoted. We call this sampling strategy *online token promotion* (OTP) and in practice combine it with CCFG. Algorithm 2 summarises our full sampling procedure.

4.3.1 Error from Online Token Promotion

We note that OTP can introduce errors into the sampling process by promoting tokens too early. Indeed, even when the posterior mode has high probability, it may still be incorrect, causing the sampler to fix the position to the clean embedding of an incorrect token before terminal time. The following result upper bounds the error caused by such early promotions.

Proposition 4.1 (Promotion Error). *Let p be the target distribution on \mathcal{V}^L , and assume that each denoising samples exactly from $p(z_{t_{i+1}} | z_{t_i})$. Let \tilde{p} be the output law of the corresponding sampler that uses the promotion rule above with the true posteriors under p , and then continues with the same exact denoising dynamics conditioned on promoted values. Then*

$$\text{TV}(p, \tilde{p}) \leq \varepsilon L.$$

Proof. See Appendix E.2. □

The above results show that the overall accumulated probability of at least one incorrect promotion is bounded by εL . Note that this error does not depend on the number of discretization steps, and can be made arbitrarily small by taking ε small enough.

5 Experiments

Most prior work on FLMs has evaluated unconditional generation, typically using metrics such as generative perplexity and entropy. While these metrics measure distributional modelling quality, they do not establish whether flow-based language models can serve as useful conditional generators in practical settings. We therefore evaluate MLFM on more demanding downstream tasks that require mathematical reasoning and instruction following. Concretely, we adapt the pretrained SMDM model of Nie et al. (2025a) into an MLFM, following Section 3.1, and evaluate it on GSM8K (Cobbe et al., 2021) and MT-Bench (Zheng et al., 2023).

Below, we first describe our experimental setup in Section 5.1 then present our main results and ablations in Sections 5.2 and 5.3 respectively.

5.1 Experimental Setup

5.1.1 Adaptation Setup

For all of our experiments, we initialise our model from a pretrained SMDM with 1028M parameters¹. The pretrained backbone and input embedding matrix are kept frozen throughout all training stages. We train only a small set of adapters: LoRA adapters on attention and MLP modules, an output-head LoRA adapter, and AdaLN adapters for time conditioning. The backbone LoRA rank is 256 with $\alpha = 512$ and dropout 0.05; the output-head LoRA rank is 256 with $\alpha = 256$ and no dropout. The MLP that produces the AdaLN time-conditioning parameters has hidden dimension 512. The total number of trainable parameters in our model is 319M. For further details, see Appendix B.

We note that aside from Davis et al. (2026), who train a 1.7B-parameter categorical flow model at trillion-token scale, MLFM is, to the best of our knowledge, the first flow-language model at billion-parameter scale.

5.1.2 Adaptation Training

We train our model on SlimPajama (Soboleva et al., 2023), tokenized with the LLaMA-2 tokenizer (Touvron et al., 2023) using sequences of length 1024. We use a training budget of $\approx 100\text{B}$ processed sequence positions—this corresponds to 200k optimiser updates with an effective batch size of 512 sequences, obtained by accumulating two global batches of 256 sequences. We use the AdamW optimiser (Loshchilov and Hutter, 2017) with 3k warmup steps followed by cosine decay to 10% of the peak learning rate. The learning rate is 10^{-4} for the backbone LoRA, output-head LoRA, and AdaLN parameters. We apply weight decay of 0.01

¹More specifically, we use the official SMDM 1028M checkpoint, `mdm-1028M-3300e18-rs1-0.01-bs-1024.safetensors`, from the <https://huggingface.co/nieshen/SMDM> Hugging Face repository.

and clip the global gradient norm at 1.0. We also maintain an exponential moving average (EMA) of the adapter weights with decay 0.999, and use these EMA weights for validation.

Moreover, following Chen et al. (2026), we sample the time t in terms of the log noise-to-signal ratio (NSR) γ rather than sampling t directly. We sample γ from a three-component mixture consisting of uniform, normal, and fitted generalised-logistic components, with mixture weights 0.1/0.2/0.7 respectively (see Appendix A for further details). For our distribution π_s over masking probabilities s , we use the MaskGIT cosine schedule (Chang et al., 2022) clipped to $[0.05, 1.0]$. For our bridge (7), we set $\sigma = 0.2$. In addition, we use an auxiliary embedding loss with weight 10: from the predicted token distribution, we form the corresponding posterior-mean embedding and penalise its distance to the clean token embedding. Our adaptation stage took approximately three days on 16 NVIDIA GH200 GPUs.

5.1.3 Supervised Fine-Tuning

After the adaptation stage, we further train the adapted MLFM model on supervised fine-tuning data consisting of prompt-response pairs. The data mixture includes general instruction data from first-turn ShareGPT² examples, mathematical reasoning data from NuminaMath-CoT (LI et al., 2024), GSM8K-Aug-NL (Cobbe et al., 2021; Deng et al., 2023), and MetaMathQA (Yu et al., 2024), and code data from OpenCodeInstruct (Ahmad et al., 2025). We sample these three groups with weights 0.2, 0.6, and 0.2, respectively.

As discussed in Section 3.3, prompt tokens are always kept fixed during supervised fine-tuning and the MLFM objective is applied only to response tokens. With probability 0.5, all response tokens are masked; otherwise, the response mask ratio is sampled using the same MaskGIT cosine schedule as in Section 5.1.2. We use an SFT budget of $\approx 15\text{B}$ processed sequence positions. We use AdamW for 50k optimizer updates, using an effective batch size of 512 sequences obtained by accumulating two global batches of size 256. We use 3k warmup steps followed by cosine learning-rate decay, with peak learning rate 5×10^{-5} for the LoRA, output-head LoRA, and AdaLN parameters. Our SFT took ≈ 18 hours on 16 NVIDIA GH200 GPUs.

5.1.4 Datasets

We evaluate our SFT model on two conditional generation benchmarks: GSM8K (Cobbe et al., 2021) and MT-Bench (Zheng et al., 2023). GSM8K is a benchmark of grade-school mathematical word problems, where each example consists of a natural-language question with a numerical answer. We use GSM8K to evaluate mathematical reasoning, and measure performance using exact-match accuracy after extracting the final numerical answer from the model output.

MT-Bench is an open-ended instruction-following benchmark consisting of multi-turn user queries spanning diverse categories (Zheng et al., 2023). Following standard practice, model responses are instead evaluated by a strong language-model judge, which assigns a scalar score reflecting the quality of the answer. In our experiments, we report the first-turn MT-Bench score with GPT-4o (Achiam et al., 2023) as our judge.

For both datasets, we prompt the model using the Vicuna prompt template (Chiang et al., 2023).

5.1.5 Baselines

We compare our approach with the 1.1B MDM model of Nie et al. (2025a), which we refer to as SMDM, as well as the AR models considered in their work: LLaMA-2 (Touvron et al., 2023) for GSM8K, and their similarly sized AR model for MT-Bench.

For inference, we follow Algorithm 2 with $\epsilon = 0.05$. We use a maximum sequence length of 512 tokens and 256 sampling steps for GSM8K, and 1024 tokens and 128 sampling steps for MT-Bench. Similar to Nie et al. (2025a), we report the results with the best guidance scale $w \in \{0, 2, 4, 6\}$ and ablate with different w in Section 5.3.

²The dataset can be accessed from <https://sharegpt.com/>.

5.2 Main Results

Table 1 shows that MLFM improves substantially over both baselines on MT-Bench, achieving a first-turn score of 2.27 compared with 1.60 for SMDM and 1.57 for the similarly sized AR baseline. Notably, this gain is obtained with 128 sampling steps, half of the 256 steps used by SMDM. On GSM8K, however, MLFM remains significantly behind both LLaMA-2 and SMDM, obtaining 31.24% accuracy compared with 58.6% and 58.5%, respectively. One plausible reason is the difference in fine-tuning protocol: the SMDM GSM8K result is obtained after task-specific fine-tuning on augmented GSM8K data for 40 epochs, whereas our MLFM is fine-tuned on a broader instruction mixture. Nevertheless, these results are encouraging: to the best of our knowledge, they provide the first evidence that flow-based language models can be scaled beyond unconditional generation to downstream reasoning and instruction-following tasks. The qualitative examples in Figures 2, 3, provide further evidence that MLFM can produce coherent, multi-step responses.

Table 1: Main results on GSM8K and MT-Bench comparing MLFM with SMDM and the AR and LLaMA-2 baselines of Nie et al. (2025a). Note that SMDM uses 256 sampling steps for both datasets.

| Approach | GSM8K (accuracy% \uparrow) | MT-Bench (first-turn score \uparrow) |
|---------------------------------|----------------------------------|--|
| LLaMA-2 (Touvron et al., 2023) | 58.6 | – |
| AR baseline (Nie et al., 2025a) | – | 1.57 |
| SMDM (Nie et al., 2025a) | 58.5 | 1.60 |
| MLFM | 31.24 | 2.27 |

5.3 Ablations

Here, we study the effect of different guidance scales w , different numbers of sampling steps, and different sampling strategies.

Different sampling strategies. Table 2 shows the GSM8K and MT-Bench results across three sampling settings: no guidance (the standard DDPM sampler from Section 4.1), CCFG (Algorithm 1), and CCFG with Online Token Promotion (Algorithm 2). On both datasets, we observe that both CCFG and online token promotion significantly boost performance. This is not surprising: OTP commits high-confidence posterior modes as clean observed tokens, giving later denoising steps more reliable context than corrupted continuous states.

Table 2: Results for GSM8K and MT-Bench for different sampling strategies.

| Sampling | GSM8K (accuracy% \uparrow) | MT-Bench (first-turn score \uparrow) |
|---------------------------|----------------------------------|--|
| No guidance | 13.19 | 1.22 |
| CCFG (Algorithm 1) | 21.38 | 1.85 |
| CCFG w/ OTP (Algorithm 2) | 31.24 | 2.27 |

Different guidance scales. Figure 1 shows the GSM8K and MT-Bench results for different guidance scales. In general, we see that larger guidance scales provide the largest gains in performance. This too is not surprising, as stronger guidance makes the sampler rely more heavily on the clean observed context when resolving the remaining tokens.

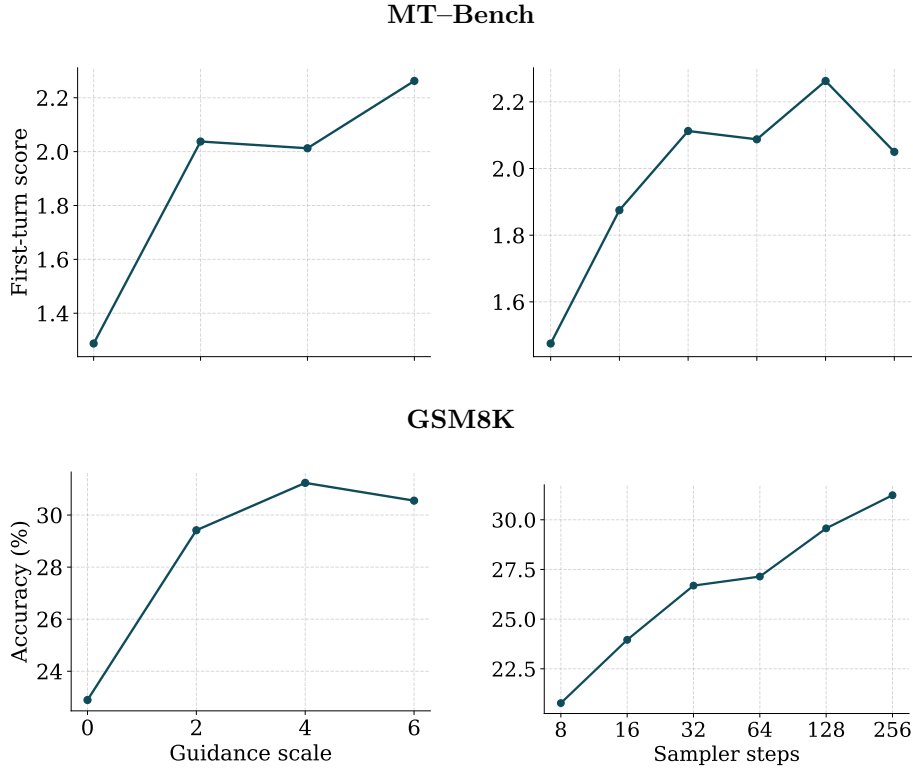


Figure 1: MT-Bench and GSM8K results across different guidance scales (left) and sampler steps (right). The plots on the right use the optimal guidance scales given by the corresponding plots on the left.

Different numbers of sampling steps. Similarly, Figure 1 shows the GSM8K and MT-Bench results for different numbers of sampling steps. We see that, in general, larger numbers of sampling steps provide the largest gains in performance. Moreover, we note that MLFM still outperforms the SMDM and AR baselines on MT-Bench even at 16 sampling steps.

6 Conclusion

In this work, we introduced Masked Language Flow Models which integrate masking from Masked Diffusion Models into Flow Language Models via a Brownian bridge connecting partially masked sequences with clean sequences. This enables exact, any-position conditional generation, allowing MLFMs to anchor continuous generation on partially masked sequences. Paired with our novel sampler, this facilitates complex, multi-step reasoning. Additionally, MLFMs support efficient training with a lightweight adaptation of pretrained MDMs. For future work, it is interesting to continue scaling MLFMs as well as to investigate the distillation of such models.

Acknowledgments

IA, KA and LZ would like to thank Jinwoo Kim and Pete Patterson for helpful conversations.

IA is supported by the Engineering and Physical Sciences Research Council [grant number EP/T517811/1]. LZ and KA are supported by the EPSRC CDT in Modern Statistics and Statistical Machine Learning (EP/S023151/1). This work was supported by the UKRI AI Research Resource (AIRR) through Isambard-AI (project AIRR-GW - Diffusion Models for Language Modelling) and by an Amazon Research Award awarded to Patrick Rebeschini (Fall 2024). SV and PR are funded by UK Research and Innovation (UKRI) under the UK government’s Horizon Europe funding guarantee [grant number EP/Y028333/1].

References

- Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023. (page 10)
- Wasi Uddin Ahmad, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Vahid Noroozi, Somshubra Majumdar, and Boris Ginsburg. Opencodestruct: A large-scale instruction tuning dataset for code llms. *arXiv preprint arXiv:2504.04030*, 2025. (pages 10 and 20)
- Michael Albergo, Nicholas M Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *Journal of Machine Learning Research*, 26(209):1–80, 2025. (page 1)
- 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. (page 1)
- Nicholas Boffi, Michael Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. *Advances in Neural Information Processing Systems*, 38:33346–33382, 2026. (page 4)
- 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, Jeffrey 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>. (page 1)
- Andrew Campbell, Joe Benton, Valentin De Bortoli, Thomas Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. *Advances in Neural Information Processing Systems*, 35:28266–28279, 2022. (page 1)
- 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*, pages 11315–11325, 2022. (pages 10 and 20)
- Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiakuan You, and Ge Liu. Langflow: Continuous diffusion rivals discrete in language modeling. *arXiv preprint arXiv:2604.11748*, 2026. (pages 1, 2, 3, 4, 5, 10, and 18)
- Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality, March 2023. URL <https://lmsys.org/blog/2023-03-30-vicuna/>. (pages 10 and 20)

- Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. (pages 2, 9, 10, and 20)
- Oscar Davis, Anastasiia Filippova, Pierre Ablin, Victor Turrisi, Amitis Shidani, Marco Cuturi, and Louis Béthune. Scaling categorical flow maps, 2026. URL <https://arxiv.org/abs/2605.07820>. (pages 1 and 9)
- Pierre Del Moral and Spiridon Penev. *Stochastic processes: From applications to theory*. Chapman and Hall/CRC, 2017. (page 2)
- Yuntian Deng, Kiran Prasad, Roland Fernandez, Paul Smolensky, Vishrav Chaudhary, and Stuart Shieber. Implicit chain of thought reasoning via knowledge distillation. *arXiv preprint arXiv:2311.01460*, 2023. (pages 10 and 20)
- Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast llms via self-distillation through time. *arXiv preprint arXiv:2410.21035*, 2024. (page 1)
- Sander Dieleman. Diffusion language models. <https://benanne.github.io/2023/01/09/diffusion-language.html>, 2023. Accessed: 2026-01-25. (page 1)
- Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 6112–6121, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1633. URL <https://aclanthology.org/D19-1633/>. (page 2)
- Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance. *arXiv preprint arXiv:2207.12598*, 2022. (page 6)
- Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020. (page 6)
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Liang Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *Iclr*, 1(2):3, 2022. (pages 5 and 19)
- Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. Elf: Embedded language flows, 2026. URL <https://arxiv.org/abs/2605.10938>. (pages 1 and 5)
- Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021. (page 18)
- Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M. Boffi, and Jinwoo Kim. Flow map language models: One-step language modeling via continuous denoising, 2026. URL <https://arxiv.org/abs/2602.16813>. (page 1)
- Jia LI, Edward Beeching, Lewis Tunstall, Ben Lipkin, Roman Soletskyi, Shengyi Costa Huang, Kashif Rasul, Longhui Yu, Albert Jiang, Ziju Shen, Zihan Qin, Bin Dong, Li Zhou, Yann Fleureau, Guillaume Lample, and Stanislas Polu. NuminaMath. [<https://huggingface.co/AI-MO/NuminaMath-CoT>] (https://github.com/project-numina/aimo-progress-prize/blob/main/report/numina_dataset.pdf), 2024. (pages 10 and 20)
- Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matthew Le. Flow matching for generative modeling. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=PqvMRDCJT9t>. (page 1)

- Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017. (pages 9 and 19)
- Alex Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. *arXiv preprint arXiv:2310.16834*, 2024. (page 1)
- Shengqi Nie, Fenglin Zhu, Chengpeng Du, Tianyu Pang, Qi Liu, Gang Zeng, Min Lin, and Chenguang Li. Scaling up masked diffusion models on text. *arXiv preprint arXiv:2410.18514*, 2025a. (pages 1, 2, 5, 9, 10, 11, and 19)
- Shengqi Nie, Fenglin Zhu, Zhen You, Xin Zhang, Jing Ou, Jing Hu, Jun Zhou, Yichang Lin, Ji-Rong Wen, and Chenguang Li. Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025b. (page 1)
- Maxwell Nye, Anders Johan Andreassen, Guy Gur-Ari, Henryk Michalewski, Jacob Austin, David Bieber, David Dohan, Aitor Lewkowycz, Maarten Bosma, David Luan, et al. Show your work: Scratchpads for intermediate computation with language models. 2021. (page 2)
- Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *International Conference on Learning Representations*, volume 2025, pages 64972–65009, 2025. (page 3)
- William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 4195–4205, 2023. (page 19)
- Peter Potaptchik, Jason Yim, Adhi Saravanan, Peter Holderrieth, Eric Vanden-Eijnden, and Michael S. Albergo. Discrete flow maps, 2026. URL <https://arxiv.org/abs/2604.09784>. (page 1)
- Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, İsmail İlkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps, 2026. URL <https://arxiv.org/abs/2602.12233>. (page 1)
- Subham Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. *Advances in Neural Information Processing Systems*, 37:130136–130184, 2024. (page 1)
- Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. *Advances in neural information processing systems*, 37:103131–103167, 2024. (page 1)
- Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. <https://cerebras.ai/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama>, 2023. URL <https://huggingface.co/datasets/cerebras/SlimPajama-627B>. (pages 9 and 20)
- Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=PxtTIG12RRHS>. (page 1)
- Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In *International Conference on Machine Learning*, pages 32211–32252. PMLR, 2023. (page 4)
- Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023. (pages 9, 10, 11, 19, and 20)
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022. (page 2)

- Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. Metamath: Bootstrap your own mathematical questions for large language models. In *International Conference on Learning Representations*, volume 2024, pages 45040–45061, 2024. (pages 10 and 20)
- Kaiwen Zheng, Yongxin Chen, Hanzi Mao, Ming-Yu Liu, Jun Zhu, and Qinsheng Zhang. Masked diffusion models are secretly time-agnostic masked models and exploit inaccurate categorical sampling. *arXiv preprint arXiv:2409.02908*, 2024. (pages 1 and 3)
- Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in neural information processing systems*, 36:46595–46623, 2023. (pages 2, 9, and 10)

Appendix Contents

| | | |
|----------|---|-----------|
| A | Gamma Schedule and Fitting | 18 |
| B | Additional Experimental Details | 19 |
| B.1 | Model and Adapter Hyperparameters | 19 |
| B.2 | Optimisation Hyperparameters | 19 |
| B.3 | Masking Details | 20 |
| B.4 | Dataset Details | 20 |
| B.5 | SFT Details | 20 |
| C | Algorithm Blocks | 22 |
| D | Additional Results | 23 |
| D.1 | Samples | 23 |
| E | Proofs | 25 |
| E.1 | Proof of Proposition 3.1 | 25 |
| E.2 | Proof of Proposition 4.1 | 25 |

A Gamma Schedule and Fitting

To sample the bridge time during training, we use an approach similar in spirit to LangFlow (Chen et al., 2026). Rather than sampling t directly, we sample the log noise-to-signal ratio γ induced by the Brownian bridge in (7):

$$\gamma = \log \left(\sigma^2 \frac{1-t}{t} \right).$$

The inverse map is

$$t = \frac{\sigma^2}{e^\gamma + \sigma^2}. \tag{14}$$

We clip γ to $[\gamma_{\min}, \gamma_{\max}] = [-10, 6]$ to avoid numerically extreme values near the bridge endpoints and to keep sampling within the range covered by our empirical observations.

As in Chen et al. (2026), the main component of our time-sampling distribution is fitted from the empirical difficulty of denoising at different noise levels. However, we found that using only a fitted component was less stable in our setting. We therefore sample γ from the three-component mixture

$$q(\gamma) = 0.1 q_{\text{unif}}(\gamma) + 0.2 q_{\text{norm}}(\gamma) + 0.7 q_{\text{fit}}(\gamma), \quad \gamma \in [-10, 6],$$

and then convert γ to t using (14). The uniform component q_{unif} is supported on $[-10, 6]$. The normal component q_{norm} is a Gaussian with location -2.5 and scale 2.0 , with samples clipped to $[-10, 6]$; these values were chosen from empirical observations of stable training regions.

The fitted component q_{fit} is constructed from cross-entropy diagnostics as a function of γ . This is the relevant diagnostic because the MLFM denoiser is trained with masked-token cross-entropy, and the average CE at a fixed noise level measures the remaining uncertainty in predicting the clean token. Thus, changes in CE across γ indicate where the model gains information along the bridge.

Concretely, we bin training examples by γ and compute the mean masked-token CE in each bin. After smoothing, we normalize this CE curve to obtain an empirical CDF-like curve on $[-10, 6]$. We then fit the generalized-logistic CDF

$$F_{\text{glog}}(\gamma) = \sigma \left(\frac{\gamma - \mu}{b} \right)^a,$$

where $a > 0$ is a shape parameter, μ is a location parameter, $b > 0$ is a scale parameter, and $\sigma(\cdot)$ denotes the logistic sigmoid. The corresponding density is

$$f_{\text{glog}}(\gamma) = \frac{a}{b} \sigma \left(\frac{\gamma - \mu}{b} \right)^a \left(1 - \sigma \left(\frac{\gamma - \mu}{b} \right) \right),$$

and we take $q_{\text{fit}} = f_{\text{glog}}$. We sample from this component by inverse transform sampling, using the quantile function

$$F_{\text{glog}}^{-1}(u) = \mu - b \log \left(u^{-1/a} - 1 \right), \quad u \in (0, 1).$$

The CE- γ summaries are updated every 200 batches using exponential smoothing with coefficient 0.999. We accept a new generalized-logistic fit only when its fit quality satisfies $R^2 \geq 0.95$; otherwise, we keep the previous fitted component. For numerical stability, the fitted scale is lower bounded by 0.05, and the shape parameter is clipped to $[0.05, 20]$. We also use low-discrepancy stratification (Kingma et al., 2021) for both γ samples and mask-ratio samples within each batch.

B Additional Experimental Details

Here, we provide further details on the experimental setup used in Section 5. Unless stated otherwise, the settings below are shared between the MLFM adaptation and SFT phases.

B.1 Model and Adapter Hyperparameters

The base MDM model we adapt is the official 1028M-parameter SMDM checkpoint³ of Nie et al. (2025a), which uses a Diff-LLaMA architecture and the LLaMA-2 tokenizer (Touvron et al., 2023). The resulting vocabulary size, hidden dimension, and number of transformer layers are 32000, 1792, and 20, respectively.

The pretrained backbone and token embedding matrix are kept frozen. We attach LoRA adapters (Hu et al., 2022) to attention and projection modules, and use a separate additive LoRA adapter for the output head. The time conditioning is provided by DiT-style AdaLN adapters (Peebles and Xie, 2023). Specifically, we apply AdaLN to the normalisation layers inside each transformer block and to the final normalisation layer before the output head. We do not apply AdaLN to the token embedding layer or to the output-head module.

Table 3: Adapter hyperparameters.

| Component | Rank / Width | α | Dropout |
|----------------------|--------------|----------|---------|
| Backbone LoRA | 256 | 512 | 0.05 |
| Output-head LoRA | 256 | 256 | 0 |
| AdaLN time embedding | 256 | – | – |
| AdaLN hidden layer | 512 | – | – |

Our total number of trainable parameters is 319M.

B.2 Optimisation Hyperparameters

Table 4 lists the optimisation settings for the adaptation and SFT phase of our MLFM. We use the AdamW optimiser (Loshchilov and Hutter, 2017) with (0.9, 0.95) for the β parameters. Weight decay is applied only to matrix-valued trainable weights. We exclude bias terms and all one-dimensional affine parameters from weight decay, including AdaLN scale/shift modulation parameters. Learning rates are linearly warmed up and then decayed by a cosine schedule to 10% of their peak value. We maintain an adapter-only EMA with decay 0.999 and use EMA weights for validation.

Table 4: Optimisation hyperparameters. We abbreviate learning rate to "LR" here.

| Hyperparameter | Adaptation | SFT |
|-----------------------|------------|--------------------|
| Global batch size | 256 | 256 |
| Gradient accumulation | 2 | 2 |
| Optimiser steps | 200k | 50k |
| Warmup steps | 3k | 3k |
| LoRA LR | 10^{-4} | 5×10^{-5} |
| Output-head LoRA LR | 10^{-4} | 5×10^{-5} |
| AdaLN LR | 10^{-4} | 5×10^{-5} |
| Weight decay | 0.01 | 0.01 |
| Gradient clipping | 1.0 | 1.0 |
| EMA decay | 0.999 | 0.999 |

³More specifically, we use the official SMDM 1028M checkpoint, `mdm-1028M-3300e18-rs1-0.01-bs-1024.safetensors`, from the <https://huggingface.co/nieshen/SMDM> Hugging Face repository.

The adaptation stage uses 200k optimiser updates with effective batch size 512 and maximum sequence length 1024, corresponding to approximately 100B processed token positions. The SFT stage uses 50k optimiser updates with effective batch size 512. The final, realised budget is approximately 15B processed token positions.

B.3 Masking Details

We sample the masking probability s from a cosine schedule inspired by MaskGIT (Chang et al., 2022). Specifically, we draw $u \sim \text{Unif}(0, 1)$ and set

$$s = \rho_{\min} + (\rho_{\max} - \rho_{\min}) \cos\left(\frac{\pi u}{2}\right),$$

with $\rho_{\min} = 0.05$ and $\rho_{\max} = 1.0$. Padding tokens, special tokens, and invalid positions are excluded from masking.

During SFT, masking is restricted to answer tokens: prompt tokens are always kept clean. With probability 0.5, the full answer span is masked. Otherwise, the answer-token masking probability is sampled from the same cosine schedule.

B.4 Dataset Details

The adaptation stage uses SlimPajama (Soboleva et al., 2023) tokenised with the LLaMA-2 tokenizer (Touvron et al., 2023) using sequences of length 1024. We use the provided train and validation splits. During adaptation, with probability 0.01 all sequences in the batch are cropped to a shared random prefix length sampled uniformly from $\{1, \dots, 1024\}$.

The SFT stage uses a mixture of general instruction, math, and code data, with the mixture weights being 0.20, 0.60 and 0.20 respectively. The general instruction data come from first-turn ShareGPT⁴ conversations. The math data come from the GSM8K-Aug-NL (Cobbe et al., 2021; Deng et al., 2023), MetaMathQA (Yu et al., 2024), and NuminaMath-CoT (LI et al., 2024) datasets with sub-mixture weights 0.60, 0.20 and 0.20 respectively. The code data come from the short OpenCodeInstruct (Ahmad et al., 2025) split, with the prefix fraction sampled from $[0.25, 0.75]$. SFT examples are capped at 1024 tokens, with dynamic cropping to multiples of 64. All SFT sources use the chat-style prompt `USER:\n{prompt}\nASSISTANT:\n`, followed by the response and an EOS token. Math targets append a separate answer field as `### {answer}` when available; code-instruction examples include sampled unit tests in the prompt when available.

SFT examples are truncated to a maximum length of 1024 tokens and 512 for MT-Bench and GSM8K respectively. To reduce padding, batches are dynamically cropped to sequence lengths that are multiples of 64. All SFT datasets use the same Vicuna prompt template (Chiang et al., 2023), `USER:\n{prompt}\nASSISTANT:\n`, followed by the response and an EOS token. For math examples, when a separate final-answer field is available, we append it to the target as `### {answer}`. For code-instruction examples, when unit tests are available, we include a sampled subset of them in the prompt.

B.5 SFT Details

The SFT stage is initialised from the EMA-smoothed adapter weights obtained after adaptation. We reset the optimizer and learning-rate schedule, keep the pretrained backbone and token embeddings frozen, and train only the LoRA adapters and AdaLN time-conditioning parameters.

During SFT, masking and continuous noising are applied only to response tokens; prompt tokens are always kept clean and visible. With probability 0.5, all response tokens are masked. Otherwise, the response mask ratio is sampled from the same MaskGIT cosine schedule used during adaptation.

⁴The dataset can be accessed from <https://sharegpt.com/>.

We also adapt the γ -sampling distribution during SFT. Unlike the adaptation stage, which uses a fitted generalized-logistic component, SFT uses an active empirical gamma curve. The SFT sampler draws

$$q_{\text{SFT}}(\gamma) = 0.1 q_{\text{unif}}(\gamma) + 0.9 q_{\text{active}}(\gamma), \quad \gamma \in [-10, 6],$$

with a normal component $\mathcal{N}(-2.5, 2.0^2)$ used as a fallback until the active curve is initialized. The active component is estimated from the high-mask SFT diagnostics with response mask ratios in $[0.95, 1.0]$. Because SFT contains many easy EOS and padding targets, the diagnostic curve uses response-token CE with EOS targets removed. We form an empirical inverse CDF over γ , smooth it with isotonic regression, and represent it with 101 quantile knots, and update the active inverse CDF by EMA:

$$Q_{k+1}(u) = (1 - \eta)Q_k(u) + \eta \widehat{Q}_k(u), \quad \eta = 0.05.$$

An update is applied only after at least 8 populated gamma bins and 4096 diagnostic examples are available. Algorithm 4 summarizes one SFT learning step.

C Algorithm Blocks

Algorithms 3 and 4 summarise the corresponding MLFM learning steps for adaptation and supervised fine-tuning.

Algorithm 3 One MLFM Adaptation Step

Require: Clean-text distribution \mathcal{D} , model p_θ , mask distribution π_s , corruption-level distribution π_t , optimizer.

- 1: Draw a minibatch $X \sim \mathcal{D}$.
- 2: Sample a mask pattern \mathcal{M}_s using $s \sim \pi_s$, and set $\mathcal{U}_s = [L] \setminus \mathcal{M}_s$.
- 3: Sample $\gamma \sim \pi_t$ and then set t using (14).
- 4: Construct $z_{s,t}$: set $z_{s,t}^\ell = E_{X^\ell}$ for $\ell \in \mathcal{U}_s$, and sample $z_{s,t}^\ell \mid X^\ell$ from (7) for $\ell \in \mathcal{M}_s$.
- 5: Compute

$$\widehat{\mathcal{L}}_{\text{MLFM}} = \frac{1}{|\mathcal{M}_s|} \sum_{\ell \in \mathcal{M}_s} -\log p_\theta^\ell(X^\ell \mid z_{s,t}, t).$$

- 6: Update θ with one optimizer step on $\widehat{\mathcal{L}}_{\text{MLFM}}$.
 - 7: Update the distribution π_t using the batch diagnostics at γ .
 - 8: **return** updated θ and π_t .
-

Algorithm 4 One MLFM Supervised Fine-Tuning Step

Require: Prompt-answer dataset \mathcal{D}_{FT} , model p_θ , response-mask distribution π_s^{FT} , corruption-level distribution π_t , optimizer.

- 1: Draw a minibatch $(p, a) \sim \mathcal{D}_{\text{FT}}$ and tokenize it as $X = X(p, a)$.
- 2: Let \mathcal{P} be prompt positions and \mathcal{A} be response positions.
- 3: Sample a nonempty response mask $\mathcal{M}_s \sim \pi_s^{\text{FT}}(\cdot \mid \mathcal{A})$, and set $\mathcal{U}_s = [L] \setminus \mathcal{M}_s$.
- 4: Sample $\gamma \sim \pi_t$ and set t using (14).
- 5: Construct $z_{s,t}$: keep all positions in \mathcal{U}_s clean, including every prompt position in \mathcal{P} , and sample $z_{s,t}^\ell \mid X^\ell$ from (7) for $\ell \in \mathcal{M}_s$.
- 6: Compute the response-token objective

$$\widehat{\mathcal{L}}_{\text{FT}} = \frac{1}{|\mathcal{M}_s|} \sum_{\ell \in \mathcal{M}_s} -\log p_\theta^\ell(X^\ell \mid z_{s,t}, t).$$

- 7: Update θ with one optimizer step on $\widehat{\mathcal{L}}_{\text{FT}}$.
 - 8: Update the distribution π_t using the response-token diagnostics at γ .
 - 9: **return** updated θ and π_t .
-

D Additional Results

D.1 Samples

Figures 2 and 3 show two qualitative examples of MLFM performing conditional generation on MT-Bench and GSM8K respectively. These examples complement our quantitative results by demonstrating that MLFM can successfully handle mathematical reasoning and instruction-following tasks, which have largely remained outside the scope of prior FLM evaluations due to their inability to handle conditional generation.

MT-Bench

Prompt

Compose an engaging travel blog post about a recent trip to Hawaii, highlighting cultural experiences and must-see attractions.

Answer

As I returned to the tropical state of California, I was struck by the incred diversity and beauty of the islands. I arrived in the island and met with the people of my country, and I was amazed by the beauty of the blue beaches, the lush green plants, and the crystal blue waters.

One of the most famous attractions of Hawaii is the unique natural beauty of the island. From the red mountains to the tropical forests, the island is a master of natural beauty. The Kilauea’s Palace, which is the largest in the United States, serves as a museum of science and technology.

As the sunseted over the island, I was curious about the rich culture of the Hawaii. The Duke Palace is the oldest and the largest museum of the islands. It is the largest collection of historical and cultural art in the world, and it is a stunning destination for tourists.

As the sunseted in the mountains, I had a fresh perspective on the history of Hawaii, a country with a rich history in the Pacific. One of the most well-known attractions in the world is the Hualaiian Cathedrals, located in the footsteps of the island’s Mau National Park. The park is located in the tropical mountains, with a beautiful lana and views on the ocean.

In addition, there was a diverse culture in the island, with different customs and traditions. For example, the fire dance is a popular dance, and the Hawaii are also interested in the cone dance, which is a traditional dance that still maintains in the culture of the islands.

Another place to visit is the Hawaiolian National Zoo. It is the largest animal park in the world, where a visitor can see a variety of animals from around the world.

Overall, my trip to Hawaii was a unique experience that cannot be beat, with its unique attractions, beautiful natural beauty, and diverse cultural experiences. If you are interested in the culture of the islands, I would recommend the Hawaii Museum and the Hawaiolian National Museum, and if you want to learn more, this assistant is very helpful.

Figure 2: Qualitative example of MLFM performing conditional generation on MT-Bench.

GSM8K

Prompt

In a dance class of 20 students, 20% enrolled in contemporary dance, 25% of the remaining enrolled in jazz dance, and the rest enrolled in hip-hop dance. What percentage of the entire students enrolled in hip-hop dance?

Answer

20% of 20 students is $20 * 0.2 = 4$. The remaining number after enrolling in contemporary dance is $20 - 4 = 16$ students. 25% of 16 students enrolled in jazz dance, which is $.25 * 16 = 4$ students. So, the number of students who are enrolled in hip-hop dance is $16 - 4 = 12$. Therefore, the percentage of students enrolled in hip-hop dance is $12 / 20 * 100 = 60$

Figure 3: Qualitative example of MLFM performing conditional generation on GSM8K.

E Proofs

E.1 Proof of Proposition 3.1

Proof. At $t = 0$, every corrupted coordinate is equal to the deterministic mask embedding m , while every uncorrupted coordinate is equal to its clean embedding E_{X^ℓ} . Therefore $z_{s,0}$ contains exactly the same information as the partially masked sequence X_s , up to the deterministic embedding map. Conditioning on either representation induces the same posterior over masked token identities. \square

E.2 Proof of Proposition 4.1

Couple $X \sim p$ with the promoted sampler. Let

$$E = \left\{ \exists i, \exists \ell \in P_i : \widehat{X}_i^\ell \neq X^\ell \right\}$$

be the event that some promoted token is wrong. Since P_i is \mathcal{F}_i -measurable,

$$\begin{aligned} \mathbb{P}(E) &\leq \sum_i \sum_{\ell=1}^L \mathbb{P}(\ell \in P_i, \widehat{X}_i^\ell \neq X^\ell) \\ &= \sum_i \sum_{\ell=1}^L \mathbb{E} \left[\mathbf{1}_{\{\ell \in P_i\}} \mathbb{P}(\widehat{X}_i^\ell \neq X^\ell \mid \mathcal{F}_i) \right] \\ &\leq \varepsilon \mathbb{E} \left[\sum_i |P_i| \right] \leq \varepsilon L, \end{aligned}$$

because each coordinate is promoted at most once.

On E^c , every promoted value agrees with the corresponding coordinate of X . By exact conditional dynamics and exact terminal decoding, the remaining randomness can be coupled so that the promoted sampler finishes with output $\widetilde{X} = X$. Therefore

$$\text{TV}(p, \widetilde{p}) \leq \mathbb{P}(\widetilde{X} \neq X) \leq \mathbb{P}(E) \leq \varepsilon L.$$