Fixed-Point Masked Generative Modeling

Andrea Miele[*,†] University of Basel
Yiming Qin LTS4, EPFL
Alba Carballo-Castro LTS4, EPFL
Justin Deschenaux CLAIRE, EPFL
Pascal Frossard LTS4, EPFL

Abstract

Masked Generative Models (MGMs) enable parallel decoding and achieve strong performance across modalities, but require full-sequence bidirectional transformers at every step, making training costly and degrading quality under low sampling budgets. Existing work improves efficiency via better samplers or cheaper fixed-depth denoisers, but they still allocate a fixed amount of denoiser computation to each refinement step. We introduce Fixed-Point Masked Generative Models (FP-MGMs), which replace part of the denoiser with a fixed-point solver over shared attention layers to enable adaptive depth with fewer parameters. To make it more effective for masked generation, we first introduce a cross-step consistency loss, which aligns hidden representations at neighboring denoising steps and, second, three-state reuse (3SR) which warm-starts the solver using the previous solution by treating differently unchanged, still-masked, and newly revealed tokens respectively. Together, these components define our complete training-to-inference framework for fixed-point masked generation, *CoFRe*. We also convert a pretrained MDLM checkpoint into an FP-MDLM with short fine-tuning, avoiding full retraining in this setting. In separate text and image experiments, CoFRe improves the quality–cost trade-off. On OpenWebText, CoFRe reduces parameters by 38.8%, training time by 11.5%, and VRAM by 16.9%, while improving generative perplexity from 830.8 to 101.8 at a budget of 96 transformer-block forward passes, compared to MDLM. In ImageNette, CoFRe reduces training time by 48.6% and VRAM by 50.7%, while improving FID in all sample budgets tested. Overall, CoFRe offers a practical framework for cheaper training and stronger low-budget masked generation.

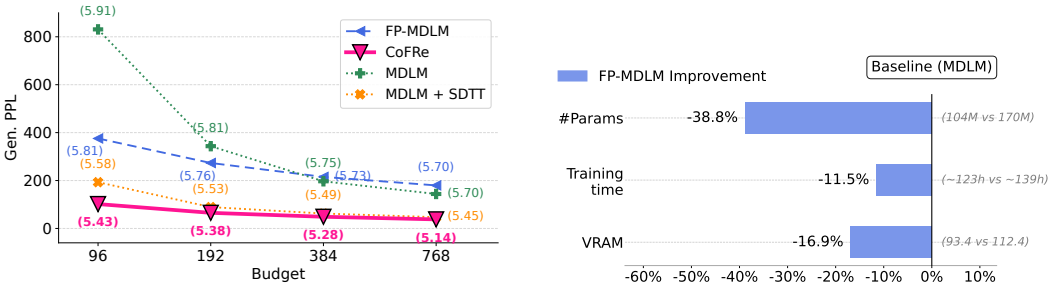


Figure 1: **FP-MDLM and CoFRe improve the quality–cost trade-off on OWT.** (Left) Generative perplexity across forward-pass budgets, with entropy in parentheses; CoFRe gives the lowest perplexity at the displayed operating points; at matched entropy, it leads at low and medium budgets, while MDLM+SDTT is stronger at budget 768 (Appendix F.1.7). (Right) Relative to MDLM, FP-MDLM and CoFRe use fewer parameters, less training time, and less VRAM.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Method family	Adaptive network depth	Cheaper training	Strong low-budget generation
Sampler improvements [5, 9, 26, 27, 52, 64]	✗	✗	✓
Efficient fixed-depth architectures [16, 66]	✗	∼	✗
Controllable-depth / looped / DEQ-style models [3, 12, 22, 35, 54, 69]	✓	∼	✗
Ours: FP-MGMs	✓	✓	✓

Table 1: Comparison of prior approaches and FP-MGMs. ∼ denotes partial coverage. We only focus here on discrete data.

1 Introduction

Masked generative models (MGMs) generate sequences by iteratively denoising masked tokens, enabling parallel decoding and strong generation quality across modalities. Prominent examples include MDLMs for language [50, 58, 62] and MaskGIT for images [7], with other related masked-generation approaches extending to video, audio, and multimodal generation [10, 48, 63, 68]. However, MGMs are computationally expensive [16, 61], since each refinement step runs a full bidirectional transformer pass over the entire sequence. Training therefore consumes large amounts of VRAM and is notably slow. Furthermore, sampling under a low compute budget, i.e., the total number of transformer-block forward passes, produces poor-quality samples [14]. Thus, improving MGMs requires controlling not only the number of denoising steps, but also the cost and effective depth of the denoiser passes at each denoising step.

Prior work addresses these issues and improves MGMs efficiency along three main axes. First, alternative samplers improve sample quality at fixed compute by changing which tokens are revealed or updated at each refinement step [5, 9, 26, 27, 52, 64]. Second, efficient but fixed-depth architectures reduce architectural waste, for example by avoiding computation on [MASK] tokens [16], or by designing more efficient masked generative backbones [66]. Finally, adaptive routing and looped transformers, and DEQ-style models provide controllable effective depth by repeatedly applying shared modules or dynamically allocating computation [3, 12, 22, 35, 54, 69].

However, these directions leave an important gap for masked generative modeling, as detailed in Tab. 1. Sampler improvements change the denoising trajectory but usually leave the training procedure and per-step denoiser unchanged, so the compute spent at each sampling step remains fixed independently of step difficulty. Efficient, fixed-depth masked architectures reduce compute, but still rely on a denoiser with fixed-capacity. This suggests that improving MGM efficiency not only requires changing the sampler, but also acting inside the denoiser: the model should reuse parameters to reduce backbone cost, while still allowing different refinement steps to use different amounts of computation. Fixed-point / DEQ-style layers provide a natural substrate for this goal: they replace an explicit stack of distinct layers with repeated applications of a shared block, whose equilibrium is used as the layer output [3]. Fixed-Point Diffusion Models further show that this idea can improve the quality–cost trade-off in continuous diffusion denoisers [4]. Yet masked generation introduces discrete, token-wise state changes, so continuous-diffusion techniques do not directly transfer and can be suboptimal. This motivates our FP-MGM framework, *CoFRe*, which combines fixed-point denoisers with additional mechanisms tailored to masked generation: cross-step consistency and token-aware three-state reuse.

Contributions We organize our contributions around three claims.

(1) Fixed-point masked denoisers improve the quality-cost trade-off. Compared to using a fixed stack of distinct transformer layers, a fixed-point layer repeatedly applies a shared block and treats the resulting equilibrium as the layer output. Since the same block is reused across solver iterations, the model can increase or decrease its effective depth by changing the number of iterations, without adding parameters. We introduce Fixed-Point Masked Generative Models (FP-MGMs), which replace the middle layers of a masked denoiser with a weight-sharing fixed-point block. We apply this approach to two representative MGMs: MDLM [58] for text, yielding FP-MDLM, and MaskGIT [7] for images, yielding FP-MaskGIT. In both cases, the fixed-point denoiser reduces the

*Correspondence to andrea.miele.pro@gmail.com. Code available at https://github.com/andreamiele/fp-mgm/.
†Work done while interning at EPFL.

number of parameters, training time, and memory, while allowing the effective denoiser depth to vary through the number of fixed-point iterations. This addresses limitations (1) and (2) (Fig. 1, Right).

(2) Three-state reuse makes FP-MDLM practical. Warm-starting from a fixed-point solution from the last denoising step is a standard way to make FP models efficient. However, in MGMs, the input sequence changes abruptly across refinement steps as tokens are revealed or replaced. As a result, the previous fixed-point solution is not equally reliable across positions. We therefore introduce a three-state reuse rule, referred to as *3SR*, that treats unchanged visible tokens, still-masked tokens, and newly revealed tokens differently. By design, unchanged visible tokens fully reuse the previous fixed-point solution, still-masked tokens partially reuse it, and newly revealed tokens rely more on the current input injection.

(3) Cross-time regularization is key for strong low-budget generation. Architecture and reuse alone are not sufficient for strong low-budget sampling: abrupt changes in the input space also induce non-smooth changes in the representation space across denoising steps. Neighboring-state consistency has been used before in masked diffusion [38]; we use a representation-space form, $\mathcal{L}_{\text{CONS}}$, that aligns the hidden states of a noisier student state and a cleaner teacher state, and integrate it with the fixed-point solver and state reuse. Empirically, this loss behaves like cross-time self-distillation, sharpening masked-token predictions and driving most of the low-budget generation gains of FP-MDLM, whereas the same loss does not improve a standard MDLM. This addresses limitation (3) (Fig. 1, Left). Together, the fixed-point denoiser, cross-step consistency, and 3SR define *CoFRe*: a complete training-to-inference recipe. Finally, we also show that pretrained MDLM checkpoints can be converted into FP-MDLMs with a short distillation stage, avoiding full retraining from scratch. These components improve low-budget sampling and reduce the cost of obtaining strong FP-MDLM checkpoints, addressing limitations (2) and (3).

We evaluate FP-MDLM on OpenWebText (OWT) [23] and downstream tasks [20], and FP-MaskGIT on ImageNette [13, 32]. On OWT, FP-MDLM reduces parameters by 38.8%, training time by 11.5%, and VRAM by 16.9% relative to MDLM, while improving low-budget generative perplexity from 830.8 to 375.6 at budget 96. With cross-step consistency and 3SR, our model CoFRe further improves over MDLM+SDTT [15] at the reported operating points, reducing generative perplexity from 193.1 to 101.8 at budget 96 and from 47.0 to 37.8 at budget 768; at matched entropy, its advantage holds at low and medium budgets, while MDLM+SDTT remains stronger at budget 768. On ImageNette, CoFRe reduces training time by 48.6% and VRAM by 50.7% relative to MaskGIT-Large, while improving FID across all reported budgets. We also show that a pretrained MDLM can be efficiently converted into a more compute-efficient FP architecture: improving over the 1M-step FP-MDLM baseline at every sampling budget, with only 4% of the original pretraining steps. Overall, CoFRe makes masked generative models cheaper to train, easier to adapt, and stronger under limited sampling budgets.

2 Background

In this paper, we use $\mathbf{x} = (x^1, \dots, x^d) \in \mathcal{V}^d$ to denote a clean sequence of length d over vocabulary $\mathcal{V} := [V]$. We denote by $\mathbf{z}_t$ a corrupted version of $\mathbf{x}$ at noise level or refinement state t, and by [MASK] the special mask token. A masked denoiser maps $(\mathbf{z}_t, t)$ to token logits $\boldsymbol{\ell}_\theta(\mathbf{z}_t, t)$, from which predictions or samples are obtained.

2.1 Discrete generative models and masked generative models

Discrete generative models learn distributions over sequences of categorical variables, such as text tokens or quantized image latents.

In this work, we focus on MGMs, which corrupt data by replacing tokens with a special mask token [MASK] and train a denoiser parameterized by θ to recover the clean sequence. The denoiser outputs logits $\boldsymbol{\ell}_\theta(\mathbf{z}_t, t)$, and the training objective is

$$\mathcal{L}_{\text{MGM}} := \mathbb{E}_{\mathbf{x}\sim\mathcal{D},\, t\sim\mathcal{U}[0,1]} \left[w(t)\, \text{CE}_{\mathcal{M}_t} \left(\boldsymbol{\ell}_\theta(\mathbf{z}_t, t), \mathbf{x} \right) \right]. \tag{1}$$

where $w : [0,1] \to \mathbb{R}^+$ is a weighting function, $\mathcal{M}_t = \{i : z_t^i = [\text{MASK}]\}$ is the set of masked positions, and $\text{CE}_{\mathcal{M}_t}(\boldsymbol{\ell}_\theta(\mathbf{z}_t, t), \mathbf{x})$ denotes the cross-entropy between the predicted logits and the

clean tokens, evaluated only on positions in $\mathcal{M}_t$. Sampling starts from a fully masked sequence and iteratively reveals subsets of tokens using repeated denoiser evaluations.

MDLM MDLMs [50, 58, 62] instantiate MGMs for language. MDLMs define an absorbing-state discrete diffusion process in which clean tokens are independently replaced by [MASK] tokens according to a time-dependent noise schedule. Training corresponds to the MGM objective in Eq. 1 with the MDLM weighting $w(t) = \frac{\alpha'_t}{1-\alpha_t}$, where α_t is the noise schedule giving the probability that a token remains clean at time t, using the notation of Sahoo et al. [58]. To keep predictions coherent across noise levels during accelerated sampling, auxiliary temporal objectives such as consistency regularization or self-distillation through time (SDTT) [15] are commonly used.

MaskGIT MaskGIT [7] is a MGM for image tokens, typically operating in the latent space of a pretrained tokenizer. Unlike MDLM, which follows a diffusion-time masking process, MaskGIT relies on confidence-based iterative decoding: at each step, it predicts masked tokens, scores their confidence, and permanently reveals a subset of high-confidence positions. Following Besnier et al. [5], we use Halton low-discrepancy schedules [25] to obtain more uniform spatial coverage; further details are given in Appendix D.3.

2.2 Deep equilibrium models, and fixed-point diffusion models

A fixed point of a map F_θ is a state $\mathbf{h}^\star$ that remains unchanged after applying the map: $\mathbf{h}^\star = F_\theta(\mathbf{h}^\star; \mathbf{u})$, where $\mathbf{u}$ is an external input. An n_{th} fixed-point layer uses this equilibrium state as its output, and approximates it by iterating a shared transformation, $\mathbf{h}^{n+1} = F_\theta(\mathbf{h}^n; \mathbf{u})$, or by using a numerical solver such as Broyden's method or Anderson acceleration. This can be viewed as a weight-sharing network whose effective depth is controlled by the number of solver iterations.

Deep Equilibrium Models (DEQs) [3] use this principle to define hidden representations implicitly, with gradients computed by implicit differentiation or approximate Jacobian-free methods. Fixed-Point Diffusion Models (FPDMs) [4] have shown that fixed-point denoisers can improve the quality–cost trade-off in continuous diffusion by replacing part of the denoiser with an implicit weight-sharing layer. At each diffusion timestep, the denoiser solves a fixed-point problem over hidden representations, and nearby timesteps often have similar solutions, enabling efficient warm starts from previous fixed-point states. This makes fixed-point layers a natural substrate for parameter-efficient and controllable-depth denoisers. Directly adapting FPDMs to masked generation is not sufficient, however. Masked refinement changes the input state discretely and token-wise: some positions remain visible, some remain masked, and others are newly revealed or replaced. Thus, previous fixed-point solutions are not uniformly reusable, and low-budget generation can still suffer from cross-step representation drift. We therefore introduce FP-MGMs together with CoFRe, a complete training-to-inference framework that adds cross-step consistency and token-aware three-state reuse to fixed-point masked denoisers.

3 Fixed-Point Denoising Networks for Masked Sequence Modeling

Having introduced masked generative models and the fixed-point perspective in the previous sections, we now describe how to combine them into controllable-depth denoisers designed specifically for discrete masked generation.

3.1 Fixed-point MGMs

A standard masked generative denoiser maps $(\mathbf{z}_t, t)$ to token logits through a finite stack of transformer layers. We instead decompose the denoiser into four parts: an explicit preprocessing stack P_{θ_P}, an input-conditioning projection G_{θ_G}, an implicit fixed-point block F_{θ_F}, and an explicit postprocessing stack H_{θ_H}. These respectively produce the initial hidden state, transform it into a conditioning signal for the fixed-point layers, solve for the denoising representation, and map this representation to logits:

$$\mathbf{h}_{\text{pre},t} = P_{\theta_P}(\mathbf{z}_t, t),\ \tilde{\mathbf{h}}_t = G_{\theta_G}(\mathbf{h}_{\text{pre},t}),\ \mathbf{h}_t^\star = \text{Fix}\Big(F_{\theta_F}(\cdot; \tilde{\mathbf{h}}_t, t)\Big),\ \boldsymbol{\ell}_\theta(\mathbf{z}_t, t) = H_{\theta_H}(\mathbf{h}_t^\star, t). \quad (2)$$

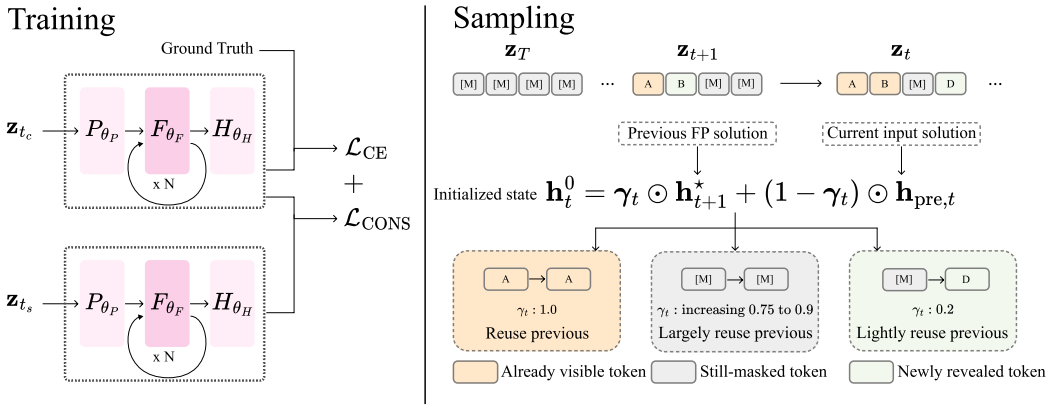


Figure 2: **Training and sampling for fixed-point masked generative models.** (Left) During training, FP-MGMs keep the masked modeling objective while replacing the middle transformer stack with an iterated shared fixed-point block. For *cross-step consistency*, correlated masks from the same clean sequence define a noisier student state and cleaner teacher state ($t_c < t_s$); the model is trained with the base cross-entropy loss plus $\mathcal{L}_{\text{CONS}}$ to align their hidden representations. (Right) During sampling, the fixed-point solver is warm-started from the previous denoising step using *three-state reuse*: visible tokens reuse fully, still-masked tokens partially reuse, and newly revealed tokens rely more on the current pre-layer representation.

where $\theta = \{\theta_P, \theta_G, \theta_F, \theta_H\}$ and $\boldsymbol{\ell}_\theta(\mathbf{z}_t, t)$ are the output token logits. If no separate projection is used, G_{θ_G} is the identity. In practice, the fixed point is approximated by N iterations:

$$\mathbf{h}_t^0 = \mathbf{h}_{\text{pre},t}, \qquad \mathbf{h}_t^{n+1} = F_{\theta_F}(\mathbf{h}_t^n; \tilde{\mathbf{h}}_t, t), \qquad n = 0, \ldots, N-1. \tag{3}$$

Let K_{pre}, K_{fp}, and K_{post} denote the number of transformer blocks in the preprocessing stack, the fixed-point block, and the postprocessing stack, respectively. A refinement step then uses $K_{\text{pre}} + NK_{\text{fp}} + K_{\text{post}}$ transformer-block evaluations, while only parameterizing $K_{\text{pre}} + K_{\text{fp}} + K_{\text{post}}$ distinct layers. Thus, weight sharing reduces parameter count, while the number of solver iterations controls the effective denoiser depth. The original MGM objective and sampling rule are unchanged; only the architecture used to compute the logits is modified.

Model variants We apply FP-MGMs to two masked generative models with different transformer denoisers: MDLM, which uses a diffusion transformer, and MaskGIT, which uses a bidirectional masked-token transformer. We detail ablations on architecture parameters in Appendices D and F.

3.2 Training a FP-MGM

Training uses the same task objective as the original masked generative model; our main modification is architectural.

Following Bai and Melas-Kyriazi [4], we train with Stochastic Jacobian-Free Backpropagation (SJFB), which avoids backpropagating through the full solver trajectory. At each training step, we sample $N_{\text{ng}} \sim \mathcal{U}\{0, \ldots, 4\}$ no-gradient iterations and $N_{\text{g}} \sim \mathcal{U}\{3, \ldots, 6\}$ gradient-tracked iterations, where $\mathcal{U}$ denotes a discrete uniform distribution, and with $N = N_{\text{ng}} + N_{\text{g}}$, with N defined in Section 3.1. The no-gradient iterations move the hidden state closer to the fixed point solution without storing activations, while the gradient-tracked iterations provide a tractable training signal for the fixed-point block. The resulting logits are passed to the original MGM loss. Full hyperparameter details are given in Appendix D.1 and more details about SJFB in Appendix C.1.

Cross-step consistency regularization The fixed-point architecture improves efficiency by reducing the cost of each denoising step, while strong low-budget generation requires more than cheaper updates. First, each update must predict the clean data accurately, since errors can quickly accumulate when only a few denoising steps are used. Second, fixed-point solutions should be reusable across adjacent denoising states, so the model does not need to resolve each state from scratch. Our lagged

logit analysis highlights the first challenge: low-budget denoising can exhibit substantial cross-step logit drift, motivating an additional consistency signal to align student-step predictions with cleaner future states, improving prediction accuracy with noisier data (Figure 16). Moreover, in masked sequence generation, tokens may be revealed between steps, so the previous fixed-point solution is a useful but imperfect warm start for the next state, further motivating stabilizing predictions across adjacent steps. Specifically, we add a short post-training stage, aligning the representation of a noisier student state with that of a cleaner teacher state from the same trajectory, using correlated masks, i.e., nested masks where the teacher context is always at least as clean as the student's. For a clean sequence $\mathbf{x}$, we construct a noisier student input $(\mathbf{z}_{t_s}, t_s)$ and a cleaner teacher input $(\mathbf{z}_{t_c}, t_c)$ from the same underlying example, with $t_c < t_s$. We then add a consistency term to the base MDLM objective, $\mathcal{L} = \mathcal{L}_{\text{MDLM}} + \lambda \mathcal{L}_{\text{CONS}}$. In our main experiments, the consistency term is an MSE loss on hidden states, $\mathcal{L}_{\text{CONS}} = \|\mathbf{h}_s - \text{sg}(\mathbf{h}_c)\|_2^2$, where $\mathbf{h}_s$ and $\mathbf{h}_c$ are the student and teacher final tokenwise pre-logit hidden states, after the FP and postprocessing blocks, and $\text{sg}(\cdot)$ denotes stop-gradient.

Although this loss is applied in representation space, it also helps improve the model output: empirically, it behaves like cross-time self-distillation, sharpening masked-token predictions and improving low-budget prediction quality. More details about the loss choice, training criterion, and correlated masks are in Appendices F.7, F.8, and C.5.3. Unlike output-space cross-step KL objectives such as CDLM's [38], our objective aligns hidden representations; on the same FP-MDLM backbone, it outperforms output-distribution KL at every budget (Table 3). As the aligned states lie after the postprocessing blocks, a lower $\mathcal{L}_{\text{CONS}}$ does not by itself bound the displacement of the reused fixed point (Appendix C.3).

3.3 Sampling with three-state reuse

At inference time, sampling follows the standard masked denoising process over a fixed number of denoising steps. Since each step requires solving a fixed-point problem, the solver initialization directly affects how much useful denoising can be obtained under a limited forward-pass budget. Following the allocation ablation in Appendix F.10, we use a decreasing fixed-point iteration schedule, which allocates more solver steps early in the denoising trajectory.

At denoising step t, the fixed-point block is conditioned on the current input injection $\tilde{\mathbf{h}}_t = G_{\theta_G}(\mathbf{h}_{\text{pre},t})$, where $\mathbf{h}_{\text{pre},t} = P_{\theta_P}(\mathbf{z}_t, t)$. Without reuse, the solver is initialized from the current pre-layer output, $\mathbf{h}_t^0 = \mathbf{h}_{\text{pre},t}$. A natural reuse strategy is to warm-start the solver from the fixed-point solution of the previous denoising step, $\mathbf{h}_t^0 = \mathbf{h}_{t+1}^\star$. The fixed-point problem is unchanged; only the solver initialization differs.

However, full reuse applies the same initialization rule to all positions, implicitly assuming that the previous fixed-point solution remains equally well aligned with the current fixed-point problem. In masked denoising, this assumption is violated in a token-dependent way: unchanged visible tokens preserve their local evidence, still-masked tokens keep the same local mask symbol but receive an updated context, and newly revealed tokens undergo a local conditioning shift. Thus, the initialization error induced by reuse is not uniform across positions. For each denoising transition under no reuse, we measure the tokenwise movement of the solved fixed-point state, $\|\mathbf{h}_{t+1}^\star(i) - \mathbf{h}_t^\star(i)\|_2$, and group positions by their discrete transition type (Figure 3). Newly revealed tokens undergo the largest representation shifts, while already visible tokens move the least. Still-masked tokens' movement significantly decreases as denoising progresses, reflecting that their conditional context changes less once more tokens are visible; this supports stronger reuse at lower noise levels.

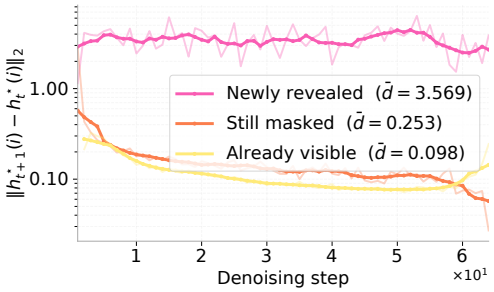


Figure 3: **Token transition type determines how reusable fixed-point states are.** Newly revealed tokens move much more than stable tokens, motivating strong reuse for visible tokens, partial reuse for masked tokens, and weak reuse for newly revealed tokens.

Inspired by this, we introduce a *three-state reuse* rule, $\mathbf{h}_t^0 = \boldsymbol{\gamma}_t \odot \mathbf{h}_{t+1}^\star + (1 - \boldsymbol{\gamma}_t) \odot \mathbf{h}_{\text{pre},t}$ where the token-wise modulation coefficient γ_t^i is broadcast over hidden dimensions and is defined as $\gamma_t^i = 1$

for unchanged visible tokens, $\gamma_t^i = \gamma_{\text{mask}}$ for still-masked tokens, and $\gamma_t^i = 0.2$ for newly revealed tokens. Thus, stable visible positions inherit the previous fixed-point solution, still-masked positions use partial reuse, and changed positions move closer to the current pre-layer output. For still-masked tokens, we increase the partial-reuse coefficient as denoising progresses, linearly interpolating from $\gamma_{\text{mask,min}}$ to $\gamma_{\text{mask,max}}$. We select the coefficients of $\boldsymbol{\gamma}_t$, including $\gamma_{\text{mask,min}}$ and $\gamma_{\text{mask,max}}$, by grid search analyzing the generation quality and diversity metrics. The full coefficient schedule, tuning procedure, and sampling algorithm are given in Appendix D.1.2 and Algorithm 1. Appendix C.3 bounds the solver error in terms of iteration depth, previous-step error, and cross-step fixed-point displacement, and Appendix C.4 tests 3SR on a controlled linear surrogate.

3.4 Pretrained model conversion and short adaptation

Finally, we show that FP-MDLMs need not be trained from scratch – given a pretrained MDLM, we are able to convert it into a more parameter-efficient FP-MDLM by mapping selected transformer layers to the preprocessing, fixed-point, and postprocessing blocks, with FP-specific projections initialized close to identity.

We then use a short teacher-student adaptation stage, similar in spirit to network distillation, to transfer the behavior of the original MDLM into the converted FP architecture. The original MDLM is kept frozen as the teacher. Using correlated masks from the same clean sequence, the converted FP-MDLM is trained with the base MDLM cross-entropy loss plus a temperature-scaled KL loss on student-masked positions $\mathcal{M}_s$: $\mathcal{L} = \mathcal{L}_{\text{base}} + \lambda\tau^2 \frac{1}{|\mathcal{M}_s|} \sum_{i \in \mathcal{M}_s} \text{KL}(p_c^\tau(i) \,\|\, p_s^\tau(i))$. This validates that transformer denoisers can be distilled into fixed-point denoisers with only short adaptation; details are given in Appendix C.5.

4 Experiments

We ask three questions. Does CoFRe improve the end-to-end quality–cost trade-off in language and image generation (Section 4.1)? Can a pretrained MDLM be converted into an effective FP-MDLM with a short adaptation stage (Section 4.2)? Which components are responsible for the gains (Section 4.3)? Additional comparisons and sweeps are in Appendix F.

4.1 CoFRe improves the quality-cost trade-off

Experimental setup. For language modeling, we evaluate on OWT [23] with context length 1024, sentence packing, and the GPT-2 tokenizer. We follow the MDLM training setup of Sahoo et al. [58]; CoFRe uses the same data, tokenizer, and objective, but replaces the middle transformer stack with a fixed-point block. We report generative perplexity (Gen. PPL, via GPT-2 Large) as a measure of quality, and unigram entropy as a measure of diversity and uncertainty [58], across fixed transformer-block budgets; see Appendix E for details. Following the allocation ablation in Appendix F.10, we use a decreasing fixed-point iteration schedule.

For image generation, we evaluate on ImageNette [13, 32] at 256×256 resolution. Following Besnier et al. [5], images are tokenized into 16×16=256 latent tokens using the ImageFolder VQ-4096/XQGAN-4096 tokenizer [42, 43]. We compare MaskGIT-Large and CoFRe under the same setup, evaluating FID (realism and alignment with the data distribution) [28] and IS (quality and diversity) [60], as well as latency, training time, and VRAM; further details are in Appendix E.

Results. Table 2 reports the main quality–cost comparison for both modalities. The reported CoFRe numbers correspond to one operating point per budget, at lower entropy than MDLM+SDTT. At these operating points, CoFRe improves generative perplexity over MDLM+SDTT at every reported budget, reducing it from 193.1 to 101.8 at budget 96 and from 47.0 to 37.8 at budget 768; the diversity-aware metrics below test whether this reflects distributional narrowing. Beyond these gains, CoFRe also reduces backbone cost: CoFRe uses 93.44 GiB/GPU and approximately 123h of training, compared to 112.4 GiB/GPU and approximately 139h for MDLM before the additional SDTT stage (Figure 1).

The same pattern holds for image generation. CoFRe improves FID over MaskGIT-Large at every reported budget, for example from 174.1 to 96.7 at budget 48 and from 30.0 to 22.8 at budget 384. It also reduces training time from 17h46m to 9h08m and VRAM from 72.45 GiB to 35.74 GiB. Overall, Table 2 shows that fixed-point denoisers improve the quality–cost trade-off in separate text and image

Language generation on OWT					Image generation on ImageNette				
	MDLM + SDTT Train: ≈139h + SDTT VRAM: 112.4 GiB/GPU		CoFRe Train: ≈123h + 30k VRAM: 93.44 GiB/GPU			MaskGIT-Large Train: 17h46m VRAM: 72.45 GiB		CoFRe Train: 9h08m VRAM: 35.74 GiB	
Budget	**Gen. PPL ↓**	**Entropy ↑**	**Gen. PPL ↓**	**Entropy ↑**	**Budget**	**FID ↓**	**IS ↑**	**FID ↓**	**IS ↑**
96	193.050	**5.580**	**101.791**	5.434	48	174.0856	9.2860	**96.7331**	**14.4074**
192	89.170	**5.530**	**65.182**	5.380	96	117.6439	13.3696	**51.0077**	**15.9572**
384	62.290	**5.490**	**48.755**	5.283	192	54.6172	**16.0220**	**27.6242**	15.0213
768	47.040	**5.450**	**37.846**	5.142	384	30.0202	**14.6473**	**22.8381**	14.2128

Table 2: **Main quality–cost results for language (Left) and image generation (Right).** Budgets count transformer-block forward passes. For language, the training/VRAM values indicate the main backbone cost; SDTT uses additional short distillation stages. Entropy is reported to contextualize diversity; these values correspond to the selected operating point for each budget. Entropy-matched frontiers are compared in Appendix F.1.7.

OWT Budget	96	192	384	768
MAUVE ↑ (5k samples)				
MDLM	0.022	0.154	0.448	0.649
MDLM + SDTT	0.273	0.592	0.631	0.613
CoFRe	**0.440**	**0.720**	**0.787**	**0.738**
Gen PPL ↓, consistency control (mean of 3 seeds)				
MDLM	839.1	346.9	196.1	143.6
MDLM + $\mathcal{L}_{\text{CONS}}$	979.8	429.4	247.6	181.0
FP-MDLM	375.0	273.5	216.6	178.7
FP-MDLM + $\mathcal{L}_{\text{CONS}}$	**104.0**	**69.7**	**53.4**	**42.1**
Gen PPL ↓, consistency objective (FP-MDLM + 3SR)				
Hidden-state MSE (ours)	**101.8**	**65.2**	**48.8**	**37.8**
Output-distribution KL	104.4	75.6	58.9	45.3

ImageNette	48	96	192	384
Precision ↑				
MaskGIT	0.206	0.362	0.427	0.476
FP-MaskGIT	0.227	0.389	0.495	0.542
CoFRe	**0.234**	**0.389**	**0.507**	**0.557**
Recall ↑				
MaskGIT	**0.734**	0.665	0.664	0.664
FP-MaskGIT	0.702	**0.670**	0.671	0.677
CoFRe	0.699	0.655	**0.676**	**0.679**

Table 3: **Fidelity–diversity metrics and controls.** (Left) OWT: MAUVE, the consistency-loss control on MDLM and FP-MDLM, and hidden-state versus output-distribution consistency on the same backbone. (Right) ImageNette precision and recall. Full tables, including all six MAUVE variants and standard deviations, are in Appendices F.1.6, F.2, F.8, and F.4.2.

experiments, and that CoFRe turns this efficiency gain into stronger low-budget generation. Further work and additional experiments in Appendix F.

Fidelity, diversity, and controls. Since CoFRe has lower entropy than the baselines, we also report distribution-level metrics (Table 3). On OWT, CoFRe has the highest MAUVE among six model variants at every budget. At matched entropy, CoFRe has lower Gen PPL than MDLM and MDLM+SDTT at budgets 96 and 192, while MDLM+SDTT is better at budget 768 (Appendix F.1.7); at high budgets, CoFRe's advantage is mainly lower parameter count, memory, and training cost at comparable quality. On ImageNette, fixed-point variants improve precision [40] at every budget; recall is preserved or slightly improved at budgets 192 and 384 and modestly lower at 48 and 96, a fidelity–coverage trade-off that FID and IS do not show. As the text and image setups differ in models, data, and metrics, we do not attribute the different trends to modality. Matched qualitative samples, including failure cases, are shown in Appendix F.1.2 (text) and Figures 14–15 (images).

4.2 Adapting a pretrained MDLM checkpoint into FP-MDLM

Experimental setup We use a pretrained MDLM teacher from Sahoo et al. [58] trained on OWT with sequence length 1024. The converted FP-MDLM is initialized from this checkpoint as described in Section 3.4, and then adapted on OWT with the same sequence length. During adaptation, we use the KL loss on correlated masked inputs, with its coefficient warmed up linearly from 0 to 0.1 over 5k steps. We compare base and adapted FP-MDLM across budgets.

Results Figure 4 and Table 11 show that a pretrained MDLM checkpoint can be converted into a stronger FP-MDLM with only 40k adaptation steps. We first compare the models without reuse in order to isolate the effect of the adaptation on the vanilla fixed-point model itself. In this setting, the adapted checkpoint improves generative perplexity at every budget, from 375.6 to 276.0 at budget 96 and from 179.7 to 139.0 at budget 768, while keeping entropy close to the baseline.

The adaptation also improves the behaviour of reuse. For the baseline FP-MDLM, reuse is inconsistent and can hurt generation quality at larger budgets. After adaptation, however, reuse becomes beneficial in the medium- and high-budget regimes: three-state reuse improves over no reuse at budgets 192, 384, and 768, and full reuse at budgets 384 and 768. The strongest results are obtained with three-state reuse, which reaches generative perplexities of 184.7, 147.5, and 126.9 at these budgets. Overall, these results show that pretrained MDLM checkpoints can be turned into effective FP-MDLM generators with a short adaptation stage, and that this adaptation restores the benefit of reuse when the sampling budget is sufficiently large. More details and results in Table 11.

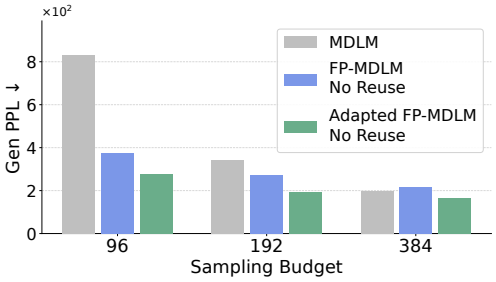


Figure 4: **Short adaptation of a pretrained MDLM checkpoint improves FP-MDLM on OWT.** Adapted FP-MDLM improves generation quality at every budget tested.

4.3 Ablations

Role of each component. The controls in Table 3 separate the three components of CoFRe. The fixed-point architecture provides the parameter, memory, and training-time savings and a large low-budget gain on its own (Gen PPL 839.1 to 375.0 at budget 96), but is weaker than MDLM at high budgets. $\mathcal{L}_{\text{CONS}}$ is the largest contributor to quality, reducing FP-MDLM Gen PPL by 72–76%, whereas it worsens a standard MDLM, so it complements the fixed-point solver rather than acting as a generic regularizer. 3SR mainly makes reuse robust: it avoids the entropy collapse of full reuse on base FP-MDLM at budget 768 and improves Gen PPL at budgets 96–384, but is tied with full reuse at budget 768 once $\mathcal{L}_{\text{CONS}}$ is used (37.8 vs. 37.6). Below, each ablation changes only the component under study.

4.3.1 Three-state reuse

Experimental setup We keep the FP-MDLM checkpoint and sampling setup fixed and vary only the solver initialization: no reuse, full reuse, or three-state reuse.

Results Figure 5 separates inference-time reuse from consistency post-training. On the base FP-MDLM (left), 3SR improves over full reuse mainly at low and medium budgets, with the gap reducing at larger budgets. Figure 3 explains this behavior: both reuse variants reduce the initialization-to-solution distance, but full reuse treats all tokens uniformly, including newly revealed tokens whose fixed-point states change the most. In contrast, 3SR weakens reuse for newly revealed tokens, partially reuses still-masked tokens, and strongly reuses stable visible tokens.

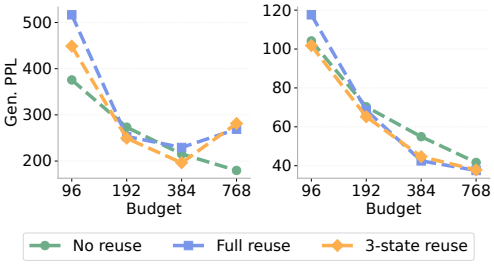


Figure 5: **Effect of different warm-start of the fixed-point on FP-MDLM base (Left) and FP-MDLM+$\mathcal{L}_{\text{CONS}}$ (Right).**

With $\mathcal{L}_{\text{CONS}}$ (right), the same trend holds, but all methods achieve much lower generative perplexity. Thus, $\mathcal{L}_{\text{CONS}}$ and 3SR are complementary: consistency improves prediction quality, while 3SR provides better token-aware solver initialization. Additional results in Appendices F.1, F.10, and C.2.

4.3.2 Adaptation initialization.

Experimental setup We compare two 40k-step FP-MDLM adaptation runs on OWT with sequence length 1024. The *Initialized* model is initialized from a pretrained MDLM checkpoint using the layer-mapping procedure described in Section 3.4, while the *Not initialized* model uses the same FP-MDLM architecture and adaptation objective but does not use this pretrained layer initialization. Both models are adapted with the same frozen teacher, correlated-mask KL objective, optimizer and sampling settings.

Results Tables 4 and 10 show that pretrained initialization improves short adaptation. Without reuse, generative perplexity drops from 296.8 to 276.0 at budget 96 and from 149.6 to 139.0 at budget 768. With 3SR, initialization is consistently better from budget 192 onward, reaching 126.9 at budget 768. The initialized run also trains faster (Figure 9), showing that the layer mapping provides a better starting point.

Table 4: **Initializing from a pretrained MDLM improves FP-MDLM adaptation with 3SR.**

		Budget			
Model	**Metric**	**96**	**192**	**384**	**768**
No init	Gen PPL ↓	298.643	192.227	149.427	131.291
	Entropy ↑	**5.722**	**5.658**	5.611	**5.597**
Init	Gen PPL ↓	**286.403**	**184.708**	**147.497**	**126.872**
	Entropy ↑	5.695	5.649	**5.619**	5.572

5 Related work

Efficient training of MGMs PUMA [37] aligns training with inference-time unmasking patterns. DiffuGPT/DiffuLLaMA [24] and Dream [65] adapt pretrained autoregressive models into bidirectional diffusion models, rather than training from scratch. We reduce training cost via a weight-shared fixed-point solver, and demonstrate conversion of one pretrained MDLM checkpoint into FP-MDLM.

Efficient few-step generation with MGMs Unmasking schedules and token-order policies select which positions to update at each step [5, 30, 33, 36, 45, 52]. Discrete solvers and timestep schedules reduce the number of sampling steps [18, 47, 51, 57]. Distillation compresses many-step teachers into few-step students [15, 31, 41, 59, 71]. Self-speculative decoding produces non-factorized predictions over masked positions in a single forward pass by drafting and validating tokens [6]. PGM [16] removes explicit mask tokens to improve the throughput during sampling. CDLM [38] reduces the number of sampling steps via a cross-step consistency objective on output distributions and uses block-wise causal attention to enable KV caching. We instead keep bidirectional attention, reduce the per-step cost via a weight-shared fixed-point solver, and apply consistency to hidden states, the quantity that 3SR reuses (Table 3). We compare against PGM and IDLM in Appendix F.3.

Implicit depth and looped models Universal [12] and looped Transformers [34, 54, 67] repeatedly apply shared blocks for adaptive depth. DEQ [3] solve for the fixed point of a shared layer via implicit differentiation, and the Generative Equilibrium Transformer [21] uses a DEQ for one-step diffusion distillation. In autoregressive LLMs, Mixture-of-Recursions and Relaxed Recursive Transformers [1, 2] use different depths for different tokens. In continuous diffusion, Fixed-Point Diffusion Models [4] combine an implicit solver with state reuse. The implicit-depth methods above apply to continuous diffusion or autoregressive language models. MGMs differ because the local conditioning of each position changes non-uniformly across denoising steps in one step, in an arbitrary order. We add two mechanisms to the fixed-point solver. (1) The three-state reuse rule handles clean, masked, and newly decoded tokens differently. (2) The cross-step consistency behaves like self-distillation and substantially improves in low-budget generative perplexity.

6 Conclusion

We introduce Fixed-Point Masked Generative Models (FP-MGMs), which replace part of the denoising transformer with an implicit weight-sharing block. In separate MDLM and MaskGIT experiments, FP-MGMs reduce parameters, training time, and memory while improving performance on low-budget generation. The fixed-point architecture provides the efficiency gains, cross-step consistency drives low-budget generation quality, and three-state reuse enables token-aware warm starts; together, they make CoFRe a complete training-to-inference recipe. We also show that one pretrained MDLM checkpoint can be converted into a fixed-point model with only short adaptation. Overall, CoFRe offers a practical path toward cheaper and more flexible masked generative models.

Limitations. Our experiments are limited to OWT-scale language modeling and ImageNette image generation, and conversion is shown for one pretrained MDLM. Whether FP-MGMs and the conversion recipe hold for larger models, ImageNet-scale data, or other masked backbones remains open (Appendix A).

Acknowledgments and Disclosure of Funding

Yiming Qin and Alba Carballo-Castro were supported by the Swiss National Science Foundation (SNSF grant 10001445). Justin Deschenaux has received funding from the Swiss State Secretariat for Education, Research and Innovation (SERI).

References

[1] Sangmin Bae, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Seungyeon Kim, and Tal Schuster. Relaxed Recursive Transformers: Effective Parameter Sharing with Layer-wise LoRA. *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.20672. 10

[2] Sangmin Bae, Yujin Kim, Reza Bayat, Sungnyun Kim, Jiyoun Ha, Tal Schuster, Adam Fisch, Hrayr Harutyunyan, Ziwei Ji, Aaron Courville, and Se-Young Yun. Mixture-of-Recursions: Learning Dynamic Recursive Depths for Adaptive Token-Level Computation. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2507.10524. 10

[3] Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep Equilibrium Models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://arxiv.org/abs/1909.01377v2. 2, 4, 10

[4] Xingjian Bai and Luke Melas-Kyriazi. Fixed Point Diffusion Models. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2024. URL http://arxiv.org/abs/2401.08741. 2, 4, 5, 10, 21, 29

[5] Victor Besnier, Mickael Chen, David Hurych, Eduardo Valle, and Matthieu Cord. Halton Scheduler For Masked Generative Image Transformer. *International Conference on Learning Representations (ICLR)*, 2025. URL http://arxiv.org/abs/2503.17076. 2, 4, 7, 10, 30

[6] Andrew Campbell, Valentin De Bortoli, Jiaxin Shi, and Arnaud Doucet. Self-Speculative Masked Diffusions. *International Conference on Learning Representations (ICLR)*, 2026. URL http://arxiv.org/abs/2510.03929. 10

[7] Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked Generative Image Transformer. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. URL http://arxiv.org/abs/2202.04200. 2, 4

[8] Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. One Billion Word Benchmark for Measuring Progress in Statistical Language Modeling. *arXiv preprint arXiv:1312.3005*, 2014. URL http://arxiv.org/abs/1312.3005. 30

[9] Sitong Chen, Shen Nie, Jiacheng Sun, Zijin Feng, Zhenguo Li, Ji-Rong Wen, and Chongxuan Li. Masked Diffusion Models as Energy Minimization. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL http://arxiv.org/abs/2509.13866. 2

[10] Marco Comunità, Zhi Zhong, Akira Takahashi, Shiqi Yang, Mengjie Zhao, Koichi Saito, Yukara Ikemiya, Takashi Shibuya, Shusuke Takahashi, and Yuki Mitsufuji. SpecMaskGIT: Masked Generative Modeling of Audio Spectrograms for Efficient Audio Synthesis and Beyond. *arXiv preprint arXiv:2406.17672*, 2024. URL http://arxiv.org/abs/2406.17672. 2

[11] Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision Transformers Need Registers. *International Conference on Learning Representations (ICLR)*, 2024. URL http://arxiv.org/abs/2309.16588. 30

[12] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal Transformers. *International Conference on Learning Representations (ICLR)*, 2019. URL http://arxiv.org/abs/1807.03819. 2, 10

[13] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2009. URL https://ieeexplore.ieee.org/document/5206848. 3, 7

[14] Justin Deschenaux and Caglar Gulcehre. Promises, outlooks and challenges of Diffusion Language Modeling. *arXiv preprint arXiv:2406.11473*, 2024. URL https://arxiv.org/abs/2406.11473. 2, 30

[15] Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast LLMs via Self-Distillation Through Time. *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.21035. 3, 4, 10, 31, 50

[16] Justin Deschenaux, Lan Tran, and Caglar Gulcehre. Partition Generative Modeling: Masked Modeling Without Masks. *International Conference on Learning Representations (ICLR)*, 2026. URL http://arxiv.org/abs/2505.18883. 2, 10, 31, 32, 41, 42

[17] Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022. URL https://arxiv.org/abs/2211.15089. 31

[18] Alberto Foresti, Mustapha Bounoua, Giulio Franzese, Luca Ambrogioni, and Pietro Michiardi. Improved Sampling Schedules for Discrete Diffusion Models. *arXiv preprint arXiv:2602.06849*, 2026. URL http://arxiv.org/abs/2602.06849. 10

[19] Samy Wu Fung, Howard Heaton, Qiuwei Li, Daniel McKenzie, Stanley Osher, and Wotao Yin. JFB: Jacobian-Free Backpropagation for Implicit Networks. *Association for the Advancement of Artificial Intelligence (AAAI)*, 2022. URL https://arxiv.org/abs/2103.12803v4. 21

[20] Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. The language model evaluation harness, 07 2024. URL https://zenodo.org/records/12608602. 3, 30

[21] Zhengyang Geng, Ashwini Pokle, and J. Zico Kolter. One-Step Diffusion Distillation via Deep Equilibrium Models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL https://arxiv.org/abs/2401.08639. 10

[22] Angeliki Giannou, Shashank Rajput, Jy yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. *International Conference on Machine Learning (ICML)*, 2023. URL https://arxiv.org/abs/2301.13196. 2

[23] Aaron Gokaslan and Vanya Cohen. OpenWebText corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019. 3, 7

[24] Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling Diffusion Language Models via Adaptation from Autoregressive Models. *International Conference on Learning Representations (ICLR)*, 2025. URL http://arxiv.org/abs/2410.17891. 10

[25] J. H. Halton. Algorithm 247: Radical-inverse quasi-random point sequence. *Communications of the ACM*, 7(12):701–702, 1964. ISSN 0001-0782. doi: 10.1145/355588.365104. URL https://doi.org/10.1145/355588.365104. 4, 30

[26] Satoshi Hayakawa, Yuhta Takida, Masaaki Imaizumi, Hiromi Wakaki, and Yuki Mitsufuji. Demystifying MaskGIT Sampler and Beyond: Adaptive Order Selection in Masked Diffusion. *Transactions on Machine Learning Research*, 2026. URL http://arxiv.org/abs/2510.04525. 2

[27] Andre He, Sean Welleck, and Daniel Fried. Reasoning with Latent Tokens in Diffusion Language Models. *arXiv preprint arXiv:2602.03769*, 2026. URL http://arxiv.org/abs/2602.03769. 2

[28] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://arxiv.org/abs/1706.08500. 7

[29] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration. *International Conference on Learning Representations (ICLR)*, 2020. URL https://openreview.net/forum?id=rygGQyrFvH. 41

[30] Chunsan Hong, Seonho An, Min-Soo Kim, and Jong Chul Ye. Improving Discrete Diffusion Unmasking Policies Beyond Explicit Reference Policies. *International Conference on Learning Representations (ICLR)*, 2026. URL http://arxiv.org/abs/2510.05725. 10

[31] Emiel Hoogeboom, David Ruhe, Jonathan Heek, Thomas Mensink, and Tim Salimans. Beyond Single Tokens: Distilling Discrete Diffusion Models via Discrete MMD. *arXiv preprint arXiv:2603.20155*, 2026. URL https://arxiv.org/abs/2603.20155. 10

[32] Jeremy Howard. Imagenette: A smaller subset of 10 easily classified classes from ImageNet. https://github.com/fastai/imagenette, 2019. 3, 7

[33] Metod Jazbec, Theo X. Olausson, Louis Béthune, Pierre Ablin, Michael Kirchhof, João Monteiro, Victor Turrisi, Jason Ramapuram, and Marco Cuturi. Learning Unmasking Policies for Diffusion Language Models. *arXiv preprint arXiv:2512.09106*, 2026. URL http://arxiv.org/abs/2512.09106. 10

[34] Ahmadreza Jeddi, Marco Ciccone, and Babak Taati. LoopFormer: Elastic-Depth Looped Transformers for Latent Reasoning via Shortcut Modulation. *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2602.11451. 10

[35] Alexia Jolicoeur-Martineau. Less is more: Recursive reasoning with tiny networks. *arXiv preprint arXiv:2510.04871*, 2025. URL https://arxiv.org/abs/2510.04871. 2

[36] Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the Worst, Plan for the Best: Understanding Token Ordering in Masked Diffusions. *International Conference on Machine Learning (ICML)*, 2025. URL http://arxiv.org/abs/2502.06768. 10

[37] Jaeyeon Kim, Jonathan Geuter, David Alvarez-Melis, Sham Kakade, and Sitan Chen. Stop Training for the Worst: Progressive Unmasking Accelerates Masked Diffusion Training. *arXiv preprint arXiv:2602.10314*, 2026. URL http://arxiv.org/abs/2602.10314. 10

[38] Minseo Kim, Chenfeng Xu, Coleman Hooper, Harman Singh, Ben Athiwaratkun, Ce Zhang, Kurt Keutzer, and Amir Gholami. CDLM: Consistency Diffusion Language Models for Faster Sampling. *Conference on Machine Learning and Systems (MLSys)*, 2026. URL https://arxiv.org/abs/2511.19269. 3, 6, 10

[39] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. *International Conference on Machine Learning (ICML)*, 2019. URL https://arxiv.org/abs/1905.00414. 26

[40] Tuomas Kynkäänniemi, Tero Karras, Samuli Laine, Jaakko Lehtinen, and Timo Aila. Improved Precision and Recall Metric for Assessing Generative Models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://arxiv.org/abs/1904.06991. 8, 44

[41] David Li, Nikita Gushchin, Dmitry Abulkhanov, Eric Moulines, Ivan Oseledets, Maxim Panov, and Alexander Korotin. IDLM: Inverse-distilled Diffusion Language Models. *arXiv preprint arXiv:2602.19066*, 2026. URL https://arxiv.org/abs/2602.19066. 10, 42

[42] Xiang Li, Kai Qiu, Hao Chen, Jason Kuen, Jiuxiang Gu, Bhiksha Raj, and Zhe Lin. Imagefolder: Autoregressive image generation with folded tokens. *arXiv preprint arXiv:2410.01756*, 2024. URL https://arxiv.org/abs/2410.01756. 7, 30

[43] Xiang Li, Kai Qiu, Hao Chen, Jason Kuen, Jiuxiang Gu, Jindong Wang, Zhe Lin, and Bhiksha Raj. XQ-GAN: An open-source image tokenization framework for autoregressive generation. *arXiv preprint arXiv:2412.01762*, 2024. URL https://arxiv.org/abs/2412.01762. 7, 30

[44] Lang Liu, Krishna Pillutla, Sean Welleck, Sewoong Oh, Yejin Choi, and Zaid Harchaoui. Divergence frontiers for generative models: Sample complexity, quantization effects, and frontier integrals. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://openreview.net/forum?id=Z_J5bCb4Rra. 37

[45] Sulin Liu, Juno Nam, Andrew Campbell, Hannes Stärk, Yilun Xu, Tommi Jaakkola, and Rafael Gómez-Bombarelli. Think While You Generate: Discrete Diffusion with Planned Denoising. *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.06264. 10

[46] Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution. *International Conference on Machine Learning (ICML)*, 2024. URL http://arxiv.org/abs/2310.16834. 31

[47] Omer Luxembourg, Haim Permuter, and Eliya Nachmani. Plan for Speed: Dilated Scheduling for Masked Diffusion Language Models. *arXiv preprint arXiv:2506.19037*, 2025. URL http://arxiv.org/abs/2506.19037. 10

[48] David Mizrahi, Roman Bachmann, Oğuzhan Fatih Kar, Teresa Yeo, Mingfei Gao, Afshin Dehghan, and Amir Zamir. 4M: Massively Multimodal Masked Modeling. *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. URL http://arxiv.org/abs/2312.06647. 2

[49] Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up Masked Diffusion Models on Text. *International Conference on Learning Representations (ICLR)*, 2025. URL http://arxiv.org/abs/2410.18514. 30

[50] 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. *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2406.03736. 2, 4

[51] Yong-Hyun Park, Chieh-Hsin Lai, Satoshi Hayakawa, Yuhta Takida, and Yuki Mitsufuji. "Jump Your Steps": Optimizing Sampling Schedule of Discrete Diffusion Models. *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.07761. 10

[52] Fred Zhangzhi Peng, Zachary Bezemek, Sawan Patel, Jarrid Rector-Brooks, Sherwood Yao, Avishek Joey Bose, Alexander Tong, and Pranam Chatterjee. Path Planning for Masked Diffusion Model Sampling. *arXiv preprint arXiv:2502.03540*, 2025. URL http://arxiv.org/abs/2502.03540. 2, 10

[53] Krishna Pillutla, Swabha Swayamdipta, Rowan Zellers, John Thickstun, Sean Welleck, Yejin Choi, and Zaid Harchaoui. MAUVE: Measuring the gap between neural text and human text using divergence frontiers. *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://openreview.net/forum?id=Tqx7nJp7PR. 37

[54] Hayden Prairie, Zachary Novack, Taylor Berg-Kirkpatrick, and Daniel Y. Fu. Parcae: Scaling laws for stable looped language models. *arXiv preprint arXiv:2604.12946*, 2026. URL https://arxiv.org/abs/2604.12946. 2, 10, 29

[55] Patrick Pynadath, Jiaxin Shi, and Ruqi Zhang. Generative frontiers: Why evaluation matters for diffusion language models. *arXiv preprint arXiv:2604.02718*, 2026. URL https://arxiv.org/abs/2604.02718. 38, 57

[56] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019. URL https://openai.com/blog/better-language-models/. 31

[57] Yinuo Ren, Haoxuan Chen, Yuchen Zhu, Wei Guo, Yongxin Chen, Grant M. Rotskoff, Molei Tao, and Lexing Ying. Fast Solvers for Discrete Diffusion Models: Theory and Applications of High-Order Algorithms. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL http://arxiv.org/abs/2502.00234. 10

[58] Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and Effective Masked Diffusion Language Models. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL http://arxiv.org/abs/2406.07524. 2, 4, 7, 8, 29, 30, 31, 56

[59] Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The Diffusion Duality. *International Conference on Machine Learning (ICML)*, 2025. URL http://arxiv.org/abs/2506.10892. 10

[60] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans. *Advances in Neural Information Processing Systems (NeurIPS)*, 2016. URL https://arxiv.org/abs/1606.03498. 7

[61] Ivan Sedykh, Nikita Sorokin, and Valentin Malykh. Not All Denoising Steps Are Equal: Model Scheduling for Faster Masked Diffusion Language Models. *arXiv preprint arXiv:2604.02340*, 2026. URL http://arxiv.org/abs/2604.02340. 2

[62] Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and Generalized Masked Diffusion for Discrete Data. *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL http://arxiv.org/abs/2406.04329. 2, 4

[63] Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, and Dumitru Erhan. Phenaki: Variable Length Video Generation From Open Domain Textual Description. *arXiv preprint arXiv:2210.02399*, 2022. URL http://arxiv.org/abs/2210.02399. 2

[64] Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Remasking discrete diffusion models with inference-time scaling. *Advances in Neural Information Processing Systems (NeurIPS)*, 2025. URL https://arxiv.org/abs/2503.00307. 2, 37, 41

[65] Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion Large Language Models. *arXiv preprint arXiv:2508.15487*, 2025. URL http://arxiv.org/abs/2508.15487. 10

[66] Zebin You, Jingyang Ou, Xiaolu Zhang, Jun Hu, Jun Zhou, and Chongxuan Li. Effective and Efficient Masked Image Generation Models. *International Conference on Machine Learning (ICML)*, 2025. URL http://arxiv.org/abs/2503.07197. 2

[67] Chengting Yu, Xiaobo Shu, Yadao Wang, Yizhen Zhang, Haoyi Wu, You Wu, Rujiao Long, Ziheng Chen, Yuchi Xu, Wenbo Su, and Bo Zheng. SpiralFormer: Looped Transformers Can Learn Hierarchical Dependencies via Multi-Resolution Recursion. *arXiv preprint arXiv:2602.11698*, 2026. URL http://arxiv.org/abs/2602.11698. 10

[68] Lijun Yu, Yong Cheng, Kihyuk Sohn, José Lezama, Han Zhang, Huiwen Chang, Alexander G. Hauptmann, Ming-Hsuan Yang, Yuan Hao, Irfan Essa, and Lu Jiang. MAGVIT: Masked Generative Video Transformer. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2023. URL https://openaccess.thecvf.com/content/CVPR2023/papers/Yu_MAGVIT_Masked_Generative_Video_Transformer_CVPR_2023_paper.pdf. 2

[69] Shuibai Zhang, Caspian Zhuang, Chihan Cui, Zhihan Yang, Fred Zhangzhi Peng, Yanxin Zhang, Haoyue Bai, Zack Jia, Yang Zhou, Guanhua Chen, and Ming Liu. Expert-choice routing enables adaptive computation in diffusion language models. *arxiv preprint arXiv:2604.01622*, 2026. URL https://arxiv.org/abs/2604.01622. 2

[70] 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. *International Conference on Learning Representations (ICLR)*, 2025. URL http://arxiv.org/abs/2409.02908. 30, 31, 32, 37

[71] Yuanzhi Zhu, Xi Wang, Stéphane Lathuilière, and Vicky Kalogeiton. Di$\mathtt{[M]}$O: Distilling Masked Diffusion Models into One-step Generator. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2025. URL http://arxiv.org/abs/2503.15457. 10

Contents

1 Introduction **2**

2 Background **3**

2.1 Discrete generative models and masked generative models 3

2.2 Deep equilibrium models, and fixed-point diffusion models 4

3 Fixed-Point Denoising Networks for Masked Sequence Modeling **4**

3.1 Fixed-point MGMs 4

3.2 Training a FP-MGM 5

3.3 Sampling with three-state reuse 6

3.4 Pretrained model conversion and short adaptation 7

4 Experiments **7**

4.1 CoFRe improves the quality-cost trade-off 7

4.2 Adapting a pretrained MDLM checkpoint into FP-MDLM 8

4.3 Ablations 9

4.3.1 Three-state reuse 9

4.3.2 Adaptation initialization. 9

5 Related work **10**

6 Conclusion **10**

A Limitations and future work **20**

B Ethics statement **20**

C Additional Method Details **20**

C.1 Stochastic Jacobian-Free Backpropagation 20

C.2 Three-State Reuse Details 21

C.3 A conditional fixed-point error bound 24

C.4 Controlled linear fixed-point surrogate 25

C.5 Pretrained MDLM Conversion Details 26

C.5.1 Analysis of the pretrained MDLM checkpoint 26

C.5.2 Layer mapping and initialization 27

C.5.3 Correlated mask construction 28

C.5.4 Effect on the training loss when using or not initialization 28

D Experimental Details **29**

D.1 Hyperparameter Tuning Protocol 29

D.1.1 Fixed-Point MGM hyperparameters 29

D.1.2 Learning-rate and solver tuning . . . 29
D.2 Language Modeling Setup . . . 29
D.2.1 OpenWebText . . . 30
D.2.2 Downstream evaluation . . . 30
D.3 Image Modeling Setup . . . 30
D.4 Sampling Precision . . . 30
D.5 Training Costs and Resources . . . 31

E Metrics details **31**
E.1 Generative perplexity . . . 31
E.2 Unigram Entropy . . . 31
E.3 Fréchet Inception Distance and Inception Score . . . 31
E.4 Throughput and Latency . . . 32
E.5 Training time and VRAM used . . . 32

F Additional Results **32**
F.1 Extended OWT Generation Results . . . 32
F.1.1 Samples . . . 32
F.1.2 Matched qualitative text samples . . . 33
F.1.3 MDLM vs FP-MDLM . . . 35
F.1.4 Initialized vs not initialized FP-MDLM adaptation . . . 36
F.1.5 Generation quality when adding different components of CoFRe . . . 36
F.1.6 Mauve score evaluation . . . 37
F.1.7 Entropy–perplexity frontiers . . . 38
F.1.8 Downstream evaluation . . . 41
F.1.9 Sampling with nucleus sampling . . . 41
F.2 Using consistency loss on MDLM . . . 41
F.3 Comparison against other distilled or accelerated diffusion language models . . . 42
F.4 Extended Image Modeling Results . . . 44
F.4.1 Samples . . . 44
F.4.2 Image precision and recall . . . 44
F.4.3 MaskGIT-Large vs MaskGIT-12 vs FP-MaskGIT . . . 47
F.4.4 Generation quality when adding different components of CoFRe . . . 47
F.5 Component Ablation for FP-MGM . . . 47
F.6 Cross-step consistency recovers low-budget generation quality . . . 49
F.7 Consistency Loss Training Dynamics . . . 50
F.8 Ablations on the Consistency Loss Type . . . 51
F.9 Tradeoff Between Denoising Steps and Fixed-Point Iterations . . . 53
F.10 Different budget allocation strategies . . . 54
F.11 Latency and generation quality for language modeling . . . 55

F.12 Fixed-point residual analysis. 56
F.13 Effect of Training Duration on Generation Quality 56

A Limitations and future work

While this work advances masked generative modeling by introducing fixed-point denoisers, cross-step consistency regularization, and three-state reuse, several limitations remain. We discuss these limitations below, both to clarify the scope of our current results and to highlight directions for future research.

Scale and scope. Our experiments are limited to OWT-scale language modeling and ImageNette image generation. These settings allow controlled comparisons, but do not yet show whether FP-MGMs scale to larger language models, larger image datasets, or multimodal generation. Evaluating FP-MGMs at larger model and data scales is an important direction for future work.

Additional tuning and adaptation. FP-MGMs introduce extra design choices, including where to place the fixed-point block, how many solver iterations to use, how to set reuse coefficients, and how to allocate the sampling budget across denoising steps. We ablate these choices, but the current recipe remains partly heuristic. In addition, the best FP-MDLM results require a short consistency post-training stage, whose duration must be chosen carefully to avoid over-sharpening and entropy collapse.

Generality and practical speedups. Three-state reuse is designed for monotonic masked decoding, where tokens are revealed and then remain fixed; samplers that remask or revise visible tokens may require different reuse rules. Moreover, our main compute metric is transformer-block forward passes, which is hardware-independent but does not always translate directly into wall-clock gains because fixed-point solvers add control-flow overhead. Future work should develop adaptive stopping rules, optimized implementations, and reuse strategies for more general masked-generation trajectories.

B Ethics statement

The objective of this work is to improve the efficiency of masked generative models by introducing fixed-point denoisers, cross-step consistency regularization, and three-state reuse. Masked generative models are relevant to a broad range of applications, including language generation, image synthesis, video, audio, and multimodal modeling. Improvements in their training and sampling efficiency may therefore reduce the computational cost of developing and deploying generative models, making such models more accessible to researchers with limited compute resources.

At the same time, FP-MGMs inherit the broader risks of generative models. More efficient generation can lower the cost of producing synthetic text or images, which may amplify existing concerns around misinformation, spam, copyright misuse, or the generation of biased and harmful content. Our experiments are limited to moderate-scale language and image benchmarks, and the generated samples still exhibit failure modes such as repetition, factual inconsistency, and topic drift. As a result, we do not view the current models as directly suitable for high-stakes applications such as medical, legal, or policy decision-making.

Overall, this work primarily contributes an architectural and algorithmic efficiency improvement. We expect its main near-term impact to be methodological, by providing a route toward cheaper training and stronger low-budget masked generation. Future work should evaluate FP-MGMs at larger scales and study how efficiency gains interact with safety, bias, memorization, and misuse risks in practical deployments.

C Additional Method Details

C.1 Stochastic Jacobian-Free Backpropagation

Training an FP-MGM requires differentiating through the implicit fixed-point block, whose output at denoising state t is defined as the hidden-state solution

$$\mathbf{h}_t^\star = F_{\theta_F}(\mathbf{h}_t^\star; \tilde{\mathbf{h}}_t, t),$$

where $\tilde{\mathbf{h}}_t = G_{\theta_G}(\mathbf{h}_{\mathrm{pre},t})$ is the input-conditioning signal produced by the preceding explicit layers. In principle, one can backpropagate through this equilibrium using implicit differentiation, which

gives a gradient involving the inverse Jacobian term

$$\left(I - \frac{\partial F_{\theta_F}(\mathbf{h}_t^\star; \tilde{\mathbf{h}}_t, t)}{\partial \mathbf{h}_t^\star}\right)^{-1}.$$

However, explicitly forming or solving this Jacobian system is computationally expensive and can be unstable at scale. Jacobian-Free Backpropagation (JFB) [19] avoids this cost by first computing an approximate fixed point without storing intermediate activations, and then applying one additional fixed-point iteration with gradients enabled; the backward pass is therefore performed only through this final step, giving an approximate gradient of the form:

$$\frac{\partial \mathcal{L}}{\partial \theta_F} \approx \frac{\partial \mathcal{L}}{\partial \mathbf{h}_t^\star} \frac{\partial F_{\theta_F}(\mathbf{h}_t^\star; \tilde{\mathbf{h}}_t, t)}{\partial \theta_F}.$$

Stochastic Jacobian-Free Backpropagation (S-JFB) [4] generalizes this idea by unrolling a random number of fixed-point iterations during training. At each training step, S-JFB samples two integers $n \sim \mathcal{U}\{0, \ldots, N\}$ and $m \sim \mathcal{U}\{1, \ldots, M\}$. It first performs n fixed-point iterations under a stop-gradient/no-gradient context, producing an approximate equilibrium while avoiding the memory cost of storing these intermediate states. It then performs m additional iterations with gradient tracking enabled, and the loss is backpropagated only through these last m unrolled iterations. The hyperparameters N and M therefore control the maximum number of fixed-point iterations used without and with gradients, respectively. Compared with standard one-step JFB, S-JFB is slightly more expensive because it backpropagates through multiple final iterations rather than only one, but it remains much cheaper than full implicit differentiation or fully unrolled explicit networks. Its stochasticity also exposes the model to different approximation depths during training, which makes the fixed-point layer more robust and empirically improves optimization compared with the deterministic one-step JFB baseline.

C.2 Three-State Reuse Details

Three-state reuse is the inference-time mechanism we propose to warm-start the fixed-point solver across masked denoising steps. Unlike full reuse, which initializes every token from the previous fixed-point solution, 3SR accounts for the fact that token states evolve non-uniformly during sampling: some tokens remain visible and unchanged, some remain masked but receive updated context, and others are newly revealed. This appendix gives the exact token-wise interpolation rule used to initialize the solver, the visible-fraction-dependent coefficient schedule, and the complete sampling procedure.

Three-state reuse schedule. We detail here the schedule used by 3SR. We base this schedule on the results obtained when looking at the distance between $\mathbf{h}_t^\star$ and $\mathbf{h}_{t+1}^\star$, as plotting on Figure 3. We analyze this distance as it shows, for each type of transition, how far the previous fixed-point solution is from the current fixed-point solution, when the initialization is made without reuse. We therefore use the following schedule: Unchanged visible tokens use full reuse, with $\gamma_t = 1.0$. Still-masked tokens use $\gamma_t = \gamma_{\text{masked}} = \gamma_{\text{mask,min}} + \left(\gamma_{\text{mask,max}} - \gamma_{\text{mask,min}}\right) v_t$, while newly revealed tokens use $\gamma_t = \gamma_{\text{changed}} = 0.2$ where $v_t = \frac{1}{d}\sum_{i=1}^{d} \mathbb{1}[z_t^i \neq [\text{MASK}]\,]$ is the fraction of visible tokens at step t. In our base setting, we use $\gamma_{\text{mask,min}} = 0.75$, $\gamma_{\text{mask,max}} = 0.90$, $\gamma_{\text{changed}} = 0.2$. We tune these hyperparameters using a grid search.

To select the reuse coefficients, we run a sweep over the masked-token and newly revealed-token interpolation parameters. Tables 5 and 6 report these sweeps across sampling budgets. Table 5 varies the reuse range for still-masked tokens and the maximum reuse assigned to newly revealed tokens, while Table 6 fixes the masked-token range and varies the newly revealed-token reuse coefficient more finely. Across these sweeps, performance is relatively robust within a moderate range of coefficients, supporting the use of a simple hand-tuned 3SR schedule rather than a learned or highly budget-specific policy.

Algorithm 1 FP-MGM sampling with three-state reuse

Require: Schedule $1 = \tau_T > \cdots > \tau_0 = 0$, initial state $\mathbf{z}_{\tau_T} = [\text{MASK}]^{\,d}$
Require: Preprocessing stack P_{θ_P}, input-conditioning projection G_{θ_G}, fixed-point block F_{θ_F}, post-processing stack H_{θ_H}
Require: Per-step solver iterations N_i
Require: Reuse parameters $\gamma_{\text{mask,min}}, \gamma_{\text{mask,max}}, \gamma_{\text{changed}}$

$\mathbf{h}^{\star}_{\text{prev}} \leftarrow \emptyset$
$\mathbf{z}_{\text{prev}} \leftarrow \emptyset$
for $i = T, T-1, \ldots, 1$ **do**
  $\mathbf{h}_{\text{pre},\tau_i} \leftarrow P_{\theta_P}(\mathbf{z}_{\tau_i}, \tau_i)$
  $\tilde{\mathbf{h}}_{\tau_i} \leftarrow G_{\theta_G}(\mathbf{h}_{\text{pre},\tau_i})$
  if $\mathbf{h}^{\star}_{\text{prev}} = \emptyset$ **then**
    $\mathbf{h}^{0}_{\tau_i} \leftarrow \mathbf{h}_{\text{pre},\tau_i}$
  else
    $v_{\tau_i} \leftarrow \frac{1}{d}\sum_{j=1}^{d} \mathbb{1}[z^{j}_{\tau_i} \neq [\text{MASK}]\,]$
    $\gamma_{\text{mask}} \leftarrow \gamma_{\text{mask,min}} + (\gamma_{\text{mask,max}} - \gamma_{\text{mask,min}})v_{\tau_i}$
    Define token-wise reuse coefficients $\gamma^{j}_{\tau_i}$ as

$$\gamma^{j}_{\tau_i} = \begin{cases} 1, & \text{if } z^{j}_{\tau_i} = z^{j}_{\text{prev}} \neq [\text{MASK}]\,, \\ \gamma_{\text{mask}}, & \text{if } z^{j}_{\tau_i} = z^{j}_{\text{prev}} = [\text{MASK}]\,, \\ \gamma_{\text{changed}}, & \text{otherwise.} \end{cases}$$

    $\mathbf{h}^{0}_{\tau_i} \leftarrow \boldsymbol{\gamma}_{\tau_i} \odot \mathbf{h}^{\star}_{\text{prev}} + (1 - \boldsymbol{\gamma}_{\tau_i}) \odot \mathbf{h}_{\text{pre},\tau_i}$
  end if
  for $n = 0, \ldots, N_i - 1$ **do**
    $\mathbf{h}^{n+1}_{\tau_i} \leftarrow F_{\theta_F}(\mathbf{h}^{n}_{\tau_i}; \tilde{\mathbf{h}}_{\tau_i}, \tau_i)$
  end for
  $\mathbf{h}^{\star}_{\tau_i} \leftarrow \mathbf{h}^{N_i}_{\tau_i}$
  $\boldsymbol{\ell}_{\theta}(\mathbf{z}_{\tau_i}, \tau_i) \leftarrow H_{\theta_H}(\mathbf{h}^{\star}_{\tau_i}, \tau_i)$
  Sample $\mathbf{z}_{\tau_{i-1}}$ using the original MGM transition rule and logits $\boldsymbol{\ell}_{\theta}(\mathbf{z}_{\tau_i}, \tau_i)$
  $\mathbf{h}^{\star}_{\text{prev}} \leftarrow \mathbf{h}^{\star}_{\tau_i}$
  $\mathbf{z}_{\text{prev}} \leftarrow \mathbf{z}_{\tau_i}$
end for
return $\mathbf{z}_{\tau_0}$

Table 5: Results for `fp_mdlm` with fixed strategy across budgets. Constant settings for all runs: visible $\gamma = 1.0$, changed $\gamma_{\min} = 0.0$. Values are shown to four decimal places. These sweeps use the fixed schedule for fixed-point iterations, whereas the main results use the decreasing schedule (Appendix F.10), so values differ from Table 2.

Masked γ range	Changed $\gamma_{\max}$	Metric	Budget 96	192	384	768
$[0.60, 0.90]$	0.1	Gen PPL ↓	94.1776	83.0655	47.4160	37.0117
		Entropy ↑	5.5104	5.4820	5.3554	5.2280
	0.2	Gen PPL ↓	95.3269	81.8788	47.2272	36.9850
		Entropy ↑	5.5110	5.4766	5.3562	5.2260
	0.3	Gen PPL ↓	96.1154	81.8405	46.8567	36.9900
		Entropy ↑	5.5181	5.4817	5.3548	5.2394
$[0.60, 0.95]$	0.1	Gen PPL ↓	94.7594	82.7298	47.1981	36.7854
		Entropy ↑	5.5115	5.4771	5.3496	5.2242
	0.2	Gen PPL ↓	95.0930	81.3275	47.4469	37.2802
		Entropy ↑	5.5126	5.4770	5.3526	5.2504
	0.3	Gen PPL ↓	95.4836	83.2949	47.0793	36.8499
		Entropy ↑	5.5122	5.4893	5.3624	5.2257
$[0.75, 0.90]$	0.1	Gen PPL ↓	97.2015	81.5204	45.9489	36.8647
		Entropy ↑	5.5201	5.4752	5.3500	5.2429
	0.2	Gen PPL ↓	97.8571	80.2788	46.4174	36.0494
		Entropy ↑	5.5267	5.4656	5.3382	5.2489
	0.3	Gen PPL ↓	96.8045	80.8193	46.2536	36.4045
		Entropy ↑	5.5187	5.4681	5.3523	5.2345
$[0.75, 0.95]$	0.1	Gen PPL ↓	98.9172	80.3999	46.2176	36.8800
		Entropy ↑	5.5264	5.4639	5.3496	5.2355
	0.2	Gen PPL ↓	98.4236	81.4000	46.7018	37.2678
		Entropy ↑	5.5262	5.4768	5.3453	5.2457
	0.3	Gen PPL ↓	97.5960	80.5485	46.8595	36.5579
		Entropy ↑	5.5210	5.4687	5.3531	5.2423
$[0.85, 0.90]$	0.1	Gen PPL ↓	98.6561	80.5434	47.7492	36.8783
		Entropy ↑	5.5192	5.4795	5.3718	5.2484
	0.2	Gen PPL ↓	99.3588	79.7671	47.5099	36.3204
		Entropy ↑	5.5297	5.4734	5.3559	5.2056
	0.3	Gen PPL ↓	98.3399	79.6512	47.7224	37.3283
		Entropy ↑	5.5244	5.4711	5.3672	5.2439
$[0.85, 0.95]$	0.1	Gen PPL ↓	98.4593	79.7479	47.3798	37.1501
		Entropy ↑	5.5246	5.4711	5.3686	5.2551
	0.2	Gen PPL ↓	99.1255	78.1578	46.5278	36.6537
		Entropy ↑	5.5231	5.4590	5.3563	5.2321
	0.3	Gen PPL ↓	98.2182	79.6038	47.8894	36.5796
		Entropy ↑	5.5240	5.4698	5.3720	5.2358

Table 6: Results for `fp_mdlm` with fixed strategy for masked γ range $[0.75, 0.90]$. Constant settings for all runs: visible $\gamma = 1.0$. Values are shown to four decimal places. As in Table 5, these runs use the fixed iteration schedule rather than the decreasing schedule of the main results.

Masked γ range	$\gamma_{\text{changed,min}}$	$\gamma_{\text{changed,max}}$	Metric	Budget 96	192	384	768
$[0.75, 0.90]$	0.00	0.00	Gen PPL ↓	101.4907	62.5693	42.3873	37.8086
			Entropy ↑	5.4348	5.3472	5.1515	5.1822
	0.10	0.10	Gen PPL ↓	100.9282	62.0509	41.9018	37.2647
			Entropy ↑	5.4300	5.3373	5.1356	5.1530
	0.20	0.20	Gen PPL ↓	101.1447	60.5738	40.7417	37.1662
			Entropy ↑	5.4351	5.3447	5.0733	5.1485
	0.25	0.25	Gen PPL ↓	99.7740	61.9484	41.2334	37.7585
			Entropy ↑	5.4340	5.3493	5.1347	5.1718
	0.30	0.30	Gen PPL ↓	100.9554	61.3730	42.3543	36.9675
			Entropy ↑	5.4388	5.3411	5.0945	5.1459
	0.00	0.10	Gen PPL ↓	97.2015	81.5204	45.9489	36.8647
			Entropy ↑	5.5201	5.4752	5.3500	5.2429
	0.00	0.20	Gen PPL ↓	97.8571	80.2788	46.4174	36.0494
			Entropy ↑	5.5267	5.4656	5.3382	5.2489
	0.00	0.30	Gen PPL ↓	96.8045	80.8193	46.2536	36.4045
			Entropy ↑	5.5187	5.4681	5.3523	5.2345

C.3 A conditional fixed-point error bound

We separate the effects of solver depth, cross-step displacement, and state reuse. Let sampling visit states $k = 1, \dots, K$ in decreasing-noise order. Write $F_k(\mathbf{h}) = F_{\theta_F}(\mathbf{h}; \widetilde{\mathbf{h}}_k, t_k)$ for the fixed-point block at step k, with fixed point $\mathbf{h}_k^\star$. The solver starts from a tokenwise interpolation between its previous computed state $\widehat{\mathbf{h}}_{k-1}$ and the current cold state $\mathbf{h}_{\mathrm{pre},k}$:

$$\mathbf{h}_k^0 = A_k \widehat{\mathbf{h}}_{k-1} + (I - A_k)\mathbf{h}_{\mathrm{pre},k}, \qquad A_k = \mathrm{Diag}(\boldsymbol{\alpha}_k) \otimes I_d, \qquad 0 \le \alpha_{k,j} \le 1. \tag{4}$$

After N_k Picard iterations, $\widehat{\mathbf{h}}_k = F_k^{N_k}(\mathbf{h}_k^0)$. Define the solver error $e_k = \|\widehat{\mathbf{h}}_k - \mathbf{h}_k^\star\|$, previous error vector $\mathbf{r}_{k-1} = \widehat{\mathbf{h}}_{k-1} - \mathbf{h}_{k-1}^\star$, adjacent fixed-point displacement $\boldsymbol{\delta}_k = \mathbf{h}_{k-1}^\star - \mathbf{h}_k^\star$, and cold-start error vector $\mathbf{c}_k = \mathbf{h}_{\mathrm{pre},k} - \mathbf{h}_k^\star$.

Assumption (invariant local contraction). For each k, there is a closed set U_k containing $\mathbf{h}_k^0$ and $\mathbf{h}_k^\star$ such that $F_k(U_k) \subseteq U_k$ and

$$\|F_k(\mathbf{x}) - F_k(\mathbf{y})\| \le L_k \|\mathbf{x} - \mathbf{y}\|, \qquad 0 \le L_k < 1, \qquad \mathbf{x}, \mathbf{y} \in U_k. \tag{5}$$

The set invariance ensures that the Picard trajectory remains where the contraction inequality holds. Geometric decay of the measured operator residual is evidence compatible with this assumption, but is not a proof of a neighborhood-wide Lipschitz bound.

Proposition (aggregate solver bound). Under the assumption above, the fixed point in U_k is unique, and

$$\begin{aligned} e_k &\le L_k^{N_k} \|\mathbf{h}_k^0 - \mathbf{h}_k^\star\| \\ &\le L_k^{N_k} \big(\|A_k \mathbf{r}_{k-1}\| + \|A_k \boldsymbol{\delta}_k\| + \|(I - A_k)\mathbf{c}_k\| \big). \end{aligned} \tag{6}$$

These are, respectively, propagated previous solver error, displacement of the target fixed point, and the cold-start contribution.

Proof. Repeated application of (5) gives $\|F_k^{N_k}(\mathbf{h}_k^0) - F_k^{N_k}(\mathbf{h}_k^\star)\| \le L_k^{N_k} \|\mathbf{h}_k^0 - \mathbf{h}_k^\star\|$. The initialization error decomposes exactly as

$$\mathbf{h}_k^0 - \mathbf{h}_k^\star = A_k \mathbf{r}_{k-1} + A_k \boldsymbol{\delta}_k + (I - A_k)\mathbf{c}_k. \tag{7}$$

Applying the triangle inequality proves (6). The statement concerns the full coupled hidden state: bidirectional attention can move errors between tokens, so an independent per-token contraction bound would require an additional separability or influence-matrix assumption.

For scalar reuse, $A_k = \alpha_k I$, let $\Delta_k = \|\boldsymbol{\delta}_k\|$ and $c_k = \|\mathbf{c}_k\|$. Equation (6) yields

$$e_k \le L_k^{N_k} \left(\alpha_k e_{k-1} + \alpha_k \Delta_k + (1 - \alpha_k) c_k \right). \tag{8}$$

Thus extra solver iterations reduce the bound geometrically; reuse helps when the previous state, including its accumulated error and cross-step displacement, is a better initialization than the cold state. This is conditional on contraction and does not by itself prescribe a universally optimal iteration schedule.

Oracle soft reuse and three-state approximation. For token j, abbreviate $\mathbf{a} = \widehat{\mathbf{h}}_{k-1,j}$, $\mathbf{b} = \mathbf{h}_{\mathrm{pre},k,j}$, and $\mathbf{y} = \mathbf{h}_{k,j}^\star$. Minimizing the exact squared initialization distance over $\alpha \in [0, 1]$ gives

$$\alpha_{k,j}^{\mathrm{oracle}} = \Pi_{[0,1]} \left(\frac{\langle \mathbf{y} - \mathbf{b}, \mathbf{a} - \mathbf{b} \rangle}{\|\mathbf{a} - \mathbf{b}\|^2} \right), \tag{9}$$

when $\mathbf{a} \neq \mathbf{b}$; otherwise every coefficient gives the same initialization. To prove this, expand $\|\alpha \mathbf{a} + (1 - \alpha)\mathbf{b} - \mathbf{y}\|^2$ as a quadratic in α, set its derivative to zero, and project the minimizer onto $[0, 1]$. Since squared Euclidean error sums across token blocks, these tokenwise coefficients jointly minimize the exact initialization norm. Three-state reuse approximates this unavailable oracle with a shared coefficient for each of the visible, still-masked, and newly revealed transition classes. A looser triangle-inequality bound is affine in α and cannot justify partial reuse coefficients by itself.

Relation to cross-step consistency. The training loss in Section 3.2 acts on final pre-logit states $\mathbf{u}_k$ after the fixed-point and postprocessing blocks. It directly penalizes a sampled version of their adjacent-step change $\Gamma_k = \|\mathbf{u}_{k-1} - \mathbf{u}_k\|$. The solver bound instead contains $\Delta_k = \|\mathbf{h}^\star_{k-1} - \mathbf{h}^\star_k\|$. Reducing Γ_k does not automatically reduce Δ_k; a sufficient additional relation would be $\Delta_k \leq \beta_k \Gamma_k + \eta_k$, with constants capturing inverse stability and explicit step dependence of the postprocessing map. Without such a relation, the correct interpretation is empirical: hidden-state consistency targets a representation-space proxy for fixed-point displacement. Its possible effect on L_k and the entropy cost of excessive training are separate questions.

C.4 Controlled linear fixed-point surrogate

We use a controlled linear surrogate of the fixed-point block to isolate solver depth and warm-start quality from linguistic content. At sampling step k, its update is

$$F_k(Z) = \rho A Z W_c + H_k, \tag{10}$$

where $Z \in \mathbb{R}^{T \times d}$ is the hidden state, $A \in \mathbb{R}^{T \times T}$ mixes tokens, $W_c \in \mathbb{R}^{d \times d}$ mixes channels, ρ controls contraction, and H_k encodes the conditioning from revealed tokens. When $|\rho| \|A\|_2 \|W_c\|_2 < 1$, this update is contractive in the Frobenius norm. Its exact fixed point is obtained by solving the linear system

$$\mathrm{vec}(Z^\star_k) = \left[I - \rho(W_c^\top \otimes A)\right]^{-1} \mathrm{vec}(H_k). \tag{11}$$

We apply Picard iteration and the same 3SR scheme over a progressive masking schedule. Each token corresponds to one row of Z and is classified as unchanged visible, still masked, or newly revealed. We compare cold initialization, indiscriminate full reuse, 3SR, and an oracle initialization against the exact solution.

Figure 6 summarizes the controlled results. Error decreases as Picard depth grows. Newly revealed tokens undergo much larger adjacent-step changes than the other two classes. At small iteration budgets, 3SR reduces error relative to cold starts and full reuse and approaches the oracle. The benefit depends on both the contraction factor and the fraction of tokens revealed per step: reuse is most useful when relatively few tokens change and convergence is slow. Full reuse is harmful in about 40% of the tested settings. The final panel associates low-budget error with adjacent-step fixed-point displacement $\Delta_k = \|Z^\star_k - Z^\star_{k-1}\|$; it does not establish that the training loss mathematically controls this term in (6).

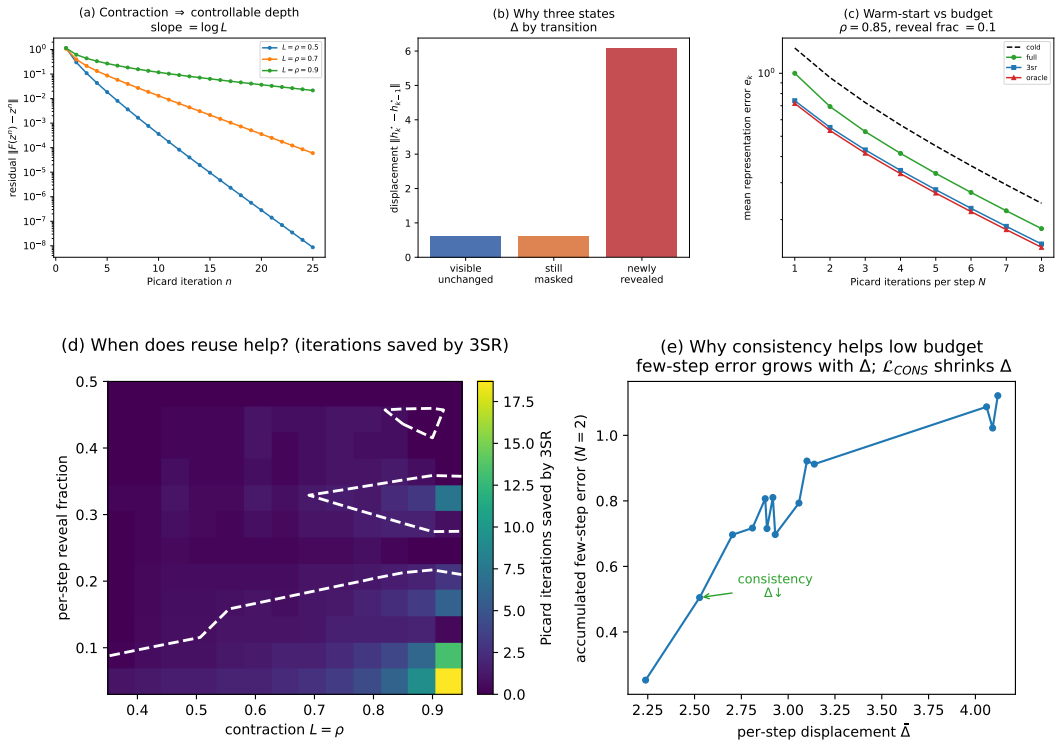


Figure 6: **Controlled fixed-point toy diagnostics.** Top: solver residual versus depth, displacement by token transition, and warm-start error across iteration budgets. Bottom: (d) Picard iterations saved by 3SR as a function of contraction and per-step reveal fraction, where dashed contours mark where the bound predicts that full reuse and cold initialization break even; (e) the association between per-step displacement and low-budget error. The applicable error bound is for the full coupled hidden state, as stated in (6).

C.5 Pretrained MDLM Conversion Details

In this part, we give further details and context on how we initialize the FP-MDLM model when we adapt it, as presented in Section 4.2, and why these design choices are motivated.

C.5.1 Analysis of the pretrained MDLM checkpoint

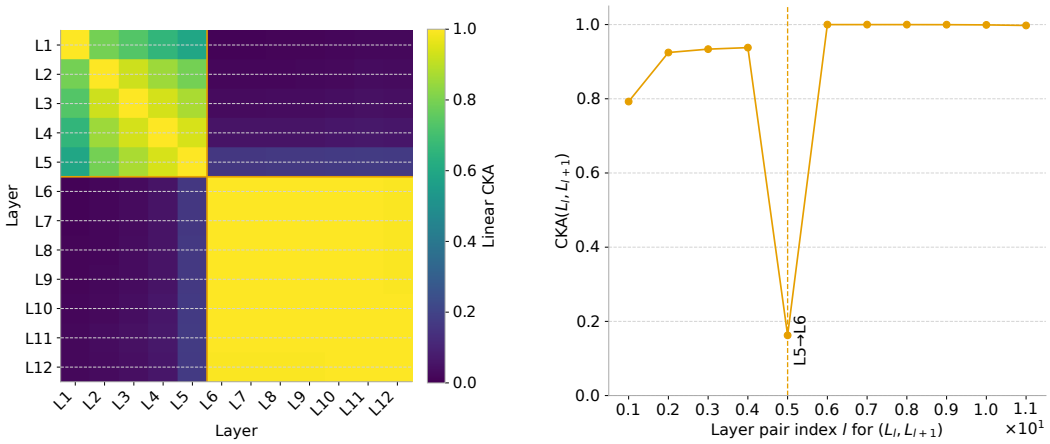
Layer similarity analysis with CKA. To better understand how to convert a pretrained MDLM into a FP model, we analyze the similarity of hidden representations across transformer layers. We use Linear Centered Kernel Alignment (CKA) [39], a standard representation-similarity measure that compares whether two layers encode examples with similar geometry. For each timestep t, we collect the residual-stream activations of every transformer layer on held-out OWT batches. For layer l, we flatten batch and sequence dimensions to obtain a feature matrix

$$X_l^{(t)} \in \mathbb{R}^{N \times d},$$

where each row corresponds to one token representation. Given two centered feature matrices X and Y, Linear CKA is defined as

$$\mathrm{CKA}(X, Y) = \frac{\left\|X^\top Y\right\|_F^2}{\left\|X^\top X\right\|_F \left\|Y^\top Y\right\|_F}.$$

CKA is close to 1 when two layers induce very similar representation geometry over the same token samples, and close to 0 when their representations are largely unrelated. We therefore use CKA to identify redundant groups of layers and potential boundaries between qualitatively different representation regimes.



(a) Mean Linear CKA across timesteps.

(b) Consecutive-layer CKA.

Figure 7: **Representation similarity in a pretrained MDLM.** We compute Linear CKA between residual-stream activations of all transformer layers at timesteps $t \in \{0.1, 0.3, 0.5, 0.7, 0.9\}$, then average the similarities across timesteps. The heatmap shows a clear two-stage structure: layers 1–5 form an early block, layers 6–12 form a highly self-similar late block, and cross-block similarity is low. The consecutive-layer plot shows a sharp drop between layers 5 and 6, followed by near-saturated similarity among the later layers.

Results and interpretation. Figure 7 shows that the pretrained MDLM has a pronounced block structure in depth. Averaged across timesteps, Linear CKA identifies the strongest and most stable boundary between layers 5 and 6; the same boundary is recovered at all analyzed timesteps. Layers 1–5 form a moderately coherent early stage, with average within-block similarity 0.808, while layers 6–12 form an extremely tight late stage, with average within-block similarity 0.998. In contrast, similarity across the two blocks is very low, 0.060, indicating that the later layers operate in a representation regime that is strongly separated from the earlier layers. The consecutive-layer profile makes this transition especially visible: similarity increases gradually through the early stack, drops sharply from 0.938 between layers 4 and 5 to 0.162 between layers 5 and 6, and then remains nearly saturated from layer 6 onward. This suggests that the deeper part of the pretrained MDLM is highly redundant and already behaves like repeated refinement in a shared representation space. This provides empirical support for replacing the later transformer stack with a shared fixed-point block.

Preliminary image-checkpoint redundancy. We also computed linear CKA for a pretrained 24-layer MaskGIT-Base 512×512 checkpoint, averaging across mask ratios. Layers 1–17 and 19–24 show high within-group similarity, with a transition near layer 18. This resembles the redundancy pattern observed in the MDLM checkpoint and suggests that some image-denoiser blocks may be candidates for weight sharing. This is preliminary architectural evidence: we have not performed a controlled conversion and generation evaluation for this pretrained MaskGIT checkpoint, so the MDLM conversion result should not be read as demonstrated transfer to image models.

C.5.2 Layer mapping and initialization

Motivated by the CKA analysis above, we initialize the converted FP-MDLM by mapping representative layers from the pretrained MDLM into the fixed-point architecture. Since layers 6-12 form a highly coherent late-stage block, we use layer 6 to initialize the shared fixed-point block. We keep the boundary layers explicit by mapping layer 1 to the preprocessing block and layer 12 to the postprocessing block (see Figure 8, Left).

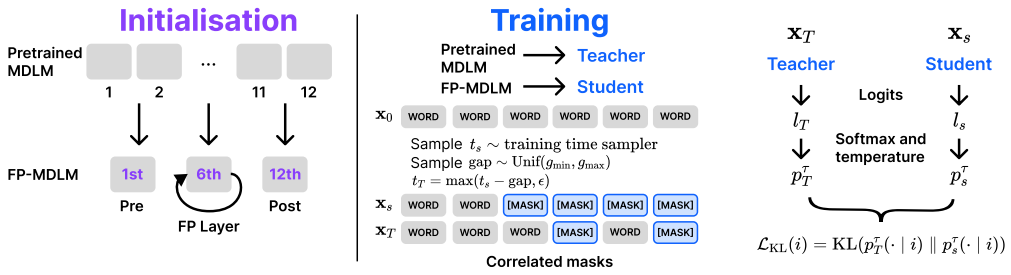


Figure 8: **Adapting an MDLM checkpoint into FP-MDLM.** (Left) FP-MDLM is initialized by mapping layers from a pretrained MDLM checkpoint to the preprocessing, fixed-point, and postprocessing blocks (see Appendix C.5.1). (Right) we then run a short adaptation stage with a teacher–student KL loss on logits, using correlated masks at two nearby noise levels, where the teacher input is less noisy than the student input.

C.5.3 Correlated mask construction

For both logits-KL adaptation against a teacher checkpoint and the consistency loss $\mathcal{L}_{\mathrm{CONS}}$, we construct the student and cleaner masks in a correlated rather than independent way. We first sample a student noise level t_s, then obtain a cleaner level t_c by subtracting a random gap $\Delta \sim \mathcal{U}[\mathrm{gap}_{\min}, \mathrm{gap}_{\max}]$. These two times are converted into keep probabilities α_s and α_c, with $\alpha_c \geq \alpha_s$. For each token position, we draw a single uniform random value u_i and reuse it for both branches:

$$z_s^i = \begin{cases} [\mathrm{MASK}], & u_i < 1 - \alpha_s, \\ x^i, & \text{otherwise}, \end{cases} \qquad z_c^i = \begin{cases} [\mathrm{MASK}], & u_i < 1 - \alpha_c, \\ x^i, & \text{otherwise}. \end{cases}$$

Because the same random numbers are reused, the two masks are nested: the cleaner mask is a subset of the student mask. Thus, the cleaner branch always sees an equally clean or cleaner context. This avoids extra variance from unrelated masking patterns and makes the consistency comparison focus on a controlled change in corruption level. In the rare case where both masks would otherwise be identical, we unmask one cleaner position so that the cleaner branch is strictly less noisy. The consistency term is evaluated only on positions masked in the student input.

C.5.4 Effect on the training loss when using or not initialization

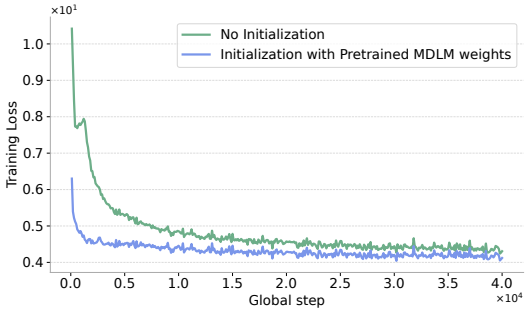


Figure 9: Comparison of the training loss between no initialization and initialization with pretrained MDLM weights when distilling the MDLM model into a FP model.

Figure 9 shows that initializing the FP model from pretrained MDLM weights provides a better starting point for short adaptation. The initialized run starts with a substantially lower training loss and remains below the non-initialized run throughout training. Both runs eventually approach similar loss values, but pretrained initialization reaches the low-loss regime much faster, which supports using layer mapping before the KL adaptation stage.

D Experimental Details

D.1 Hyperparameter Tuning Protocol

We detail in this section how we tune the majority of the hyperparameters introduced by our methods.

D.1.1 Fixed-Point MGM hyperparameters

In order to tune the hyperparameters of our fixed-point models, we use a simple staged protocol. For language modeling, all tuning is done on OWT for 100k training steps with sequence length 128, which provides a fast and inexpensive proxy before running the full setup. We first tune the architecture of the fixed-point backbone, namely the number of preprocessing and postprocessing layers, while keeping the solver settings at the default values from Bai and Melas-Kyriazi [4]. We then tune the solver iteration budget, and finally the learning rate. The full procedure is as follows:

1. **Choice of tuning setup.** We first select the target setting for hyperparameter tuning. For language modeling, we tune on OWT for 100k steps with sequence length 128 in order to obtain fast and cheap comparisons.
2. **Architecture tuning.** We tune the number of preprocessing and postprocessing layers in the fixed-point backbone, while keeping the fixed-point solver hyperparameters at the default values of Bai and Melas-Kyriazi [4]. We evaluate a small grid of candidate architectures and retain the best-performing one.
3. **Solver tuning.** With the architecture fixed, we tune the fixed-point solver budget, including the number of no-gradient and with-gradient iterations. This isolates the effect of the implicit solver from that of the backbone architecture.
4. **Learning-rate tuning.** With both the architecture and solver settings fixed, we tune the learning rate over a logarithmic grid and select the value that gives the best validation performance.
5. **Boundary check.** Whenever the best hyperparameter lies at the edge of the tested range, we extend the search range and repeat the evaluation until the selected value is not on the boundary.
6. **Final selection.** Finally, we choose the configuration that performs best under this tuning protocol and use it for the full training runs.

D.1.2 Learning-rate and solver tuning

We tune the main optimization and solver hyperparameters through small-scale experiments before running the full training jobs. We first test the base learning rate used for MDLM. However, with the original learning-rate setting, the FP-MDLM training does not converge, as seen in previous work [54]. We therefore progressively lower the learning rate until the training dynamics become stable and the validation loss shows consistent convergence. We use the same gradient clipping as MDLM, as detailed in [58].

Adaptation hyperparameters For adaptation, we keep the optimizer and model hyperparameters fixed to the base FP-MDLM setup and tune only the adaptation-specific hyperparameters. In particular, we tune the consistency weight λ, the distillation temperature τ, and the teacher-student noise-gap range $[\mathrm{gap}_{\min}, \mathrm{gap}_{\max}]$. In our base adaptation setting, we use logits-KL with $\lambda = 0.1$, $\tau = 1.5$, $\mathrm{gap}_{\min} = 0.05$, and $\mathrm{gap}_{\max} = 0.30$, apply the loss only on positions masked in the student input, and linearly warm up the consistency weight over the first 5k steps. We select these hyperparameters using a small validation sweep, choosing the setting with the best generative perplexity at matched training compute.

D.2 Language Modeling Setup

For FP-MDLM, each denoising step applies the explicit preprocessing block, then solves the implicit fixed-point layer, and finally applies the explicit postprocessing block.

We evaluate sampling quality under different compute budgets by sweeping the total forward-pass budget and the number of denoising steps. Unless stated otherwise, we use the ancestral sampler with

no additional noise-removal step for MDLM , `float64` logits before sampling, and compute sample quality with GPT-2 Large generative perplexity and sample entropy for text. We report both sampling quality and generation speed in tokens per second.

We do not include LM1B [8] because our primary evaluation target is generation quality under limited sampling budgets, rather than likelihood modelling alone. LM1B is most commonly used to report validation/test perplexity, which would mainly evaluate likelihood estimation and would not directly exercise the core advantages of CoFRe: adaptive-depth iterative masked generation and reuse across denoising steps. We therefore focus on settings such as OWT generation, where sample quality and quality–cost trade-offs can be measured more directly.

D.2.1 OpenWebText

For language modeling, we evaluate on OWT with context length 1024, sentence packing, and the GPT-2 tokenizer. We reserve the last 100k documents for validation. The MDLM baseline follows Sahoo et al. [58]: a 12-layer Diffusion Transformer with hidden size 768, RoPE, dropout 0.1, Adam with learning rate 3×10^{-4}, global batch size 512, EMA decay 0.9999, and 1M training steps. FP-MDLM uses the same data, tokenizer, and objective, but replaces the middle transformer stack with a fixed-point block. We report generative perplexity using GPT-2 Large and unigram entropy across fixed transformer-block budgets. We use a decreasing order of number of sampling steps as we found that it performs best.

D.2.2 Downstream evaluation

We evaluate downstream performance with `lm-eval-harness` [20], following the masked-model evaluation protocol of Deschenaux and Gulcehre [14] and Nie et al. [49]. Because the harness is designed for autoregressive models, we adapt its scoring rule to masked generative models: each answer choice is scored using the variational likelihood bound available for MDLM and FP-MDLM/CoFRe, and the highest-scoring choice is selected.

D.3 Image Modeling Setup

For image generation, we evaluate on ImageNette at 256×256 resolution. Images are center-cropped, resized, and tokenized into a 16×16 grid of discrete latent codes, giving a sequence length of 256. We use the ImageFolder VQ-4096/XQGAN-4096 tokenizer [42, 43] and follow the MaskGIT/Halton setup of Besnier et al. [5]. We compare MaskGIT-Large and FP-MaskGIT under the same training and sampling protocol: AdamW optimizer, learning rate 5×10^{-4}, weight decay 0.03, cosine learning-rate schedule, 1500 warmup steps, batch size 128, classifier-free guidance, and Halton sampling. We use Transformer dropout 0.1 and class-label dropout 0.1 for classifier-free guidance. Following Besnier et al. [5], the MaskGIT baseline uses one register [11]. FP-MaskGIT keeps the same tokenizer, masked-token prediction objective, and decoding procedure as MaskGIT-Large, but replaces the middle transformer stack with a fixed-point block. We report FID, IS, training time, and peak VRAM usage.

Halton sampler While effective, Besnier et al. [5] identified a notable drawback: confidence-guided sampling typically leads to spatially clustered token decoding, as the denoiser is inherently more confident near already-populated regions. Because MGMs sample positions independently from the product of marginals $\prod_{\ell \in \mathcal{S}} p_\theta(x^\ell \mid \mathbf{z}_\tau)$ instead of the exact joint distribution, clusters of neighboring tokens are more likely to produce spatial inconsistencies. To address this, Besnier et al. [5] propose to use low-discrepancy sequences [25] to guarantee more uniform spatial coverage during decoding. This modification prevents extreme clustering and leads to improved FID and IS compared to standard confidence sampling.

D.4 Sampling Precision

Zheng et al. [70] found that when Masked Diffusion Models sample with low-precision logits, some logits can underflow. This can reduce the variety of sampled tokens and make Generative Perplexity look better than it really is. Because of this, we cast all logits to FP64 before sampling.

D.5 Training Costs and Resources

All sampling experiments, for both text and images, are run on NVIDIA A100 GPUs with either 40GB or 80GB of memory. FP-MaskGIT training is also performed on A100 GPUs. For the text experiments, both MDLM and FP-MDLM are trained on 8 NVIDIA H200 GPUs with 141GB of memory per GPU. Unless otherwise stated, all reported latency, throughput, training-time, and VRAM measurements use these hardware settings.

E Metrics details

In this section, we detail the main metrics used to monitor the performance of the baseline and proposed models and methods of this paper.

E.1 Generative perplexity

We evaluate generated text using *generative perplexity*, following prior work on discrete diffusion language models [15, 46, 58]. This metric measures how well a strong autoregressive reference model predicts samples generated by our model. Concretely, we generate N_{samp} samples and score them with GPT-2 Large:

$$\text{GenPPL} = \exp\left(-\frac{1}{N_{\text{samp}}}\sum_{n=1}^{N_{\text{samp}}}\frac{1}{L_n}\sum_{i=1}^{L_n}\log p_{\text{GPT-2 Large}}(y_i^{(n)} \mid \mathbf{y}_{<i}^{(n)})\right), \tag{12}$$

where L_n is the length of generated sample $\mathbf{y}^{(n)}$, and $p_{\text{GPT-2 Large}}(y_i^{(n)} \mid \mathbf{y}_{<i}^{(n)})$ is the probability assigned by GPT-2 Large [56] to token $y_i^{(n)}$ given its prefix. Lower values indicate that generated text is more predictable under the reference language model and are therefore better.

As noted in prior work, sampling precision can affect this metric substantially. In particular, low-precision logits may artificially reduce token diversity and make generative perplexity appear better than it truly is. To avoid this issue, we cast logits to `float64` before sampling in all generative-perplexity evaluations [70].

E.2 Unigram Entropy

Generative perplexity alone can reward degenerate text, for example if a model produces repetitive or low-diversity samples. To detect such failures, we also report the *unigram entropy* of generated text, following prior work [16, 17]. For a generated sequence $\mathbf{y}^{(n)}$ of length L_n, let $c(v, \mathbf{y}^{(n)})$ denote the number of occurrences of token v. The unigram entropy is

$$H_{\text{uni}} = -\frac{1}{N_{\text{samp}}}\sum_{n=1}^{N_{\text{samp}}}\sum_{v\in\mathcal{V}}\frac{c(v,\mathbf{y}^{(n)})}{L_n}\log\frac{c(v,\mathbf{y}^{(n)})}{L_n}. \tag{13}$$

Higher values indicate more diverse token usage, while unusually low entropy can reveal collapse or repetitive generation. We therefore interpret unigram entropy jointly with generative perplexity rather than in isolation.

E.3 Fréchet Inception Distance and Inception Score

FID embeds real and generated images using a pretrained Inception network, fits a Gaussian distribution to each set of features, and computes the Fréchet distance between the two Gaussians:

$$\text{FID} = \|\mu_r - \mu_g\|_2^2 + \text{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\right),$$

where (μ_r, Σ_r) and (μ_g, Σ_g) are the empirical feature mean and covariance of real and generated images. Lower FID indicates that generated images better match the real data distribution in Inception feature space. IS instead evaluates the class predictions of the Inception network on generated images:

$$\text{IS} = \exp\left(\mathbb{E}_x \text{KL}\left(p(y \mid x) \,\|\, p(y)\right)\right).$$

It is high when individual generated images produce confident class predictions while the marginal class distribution remains diverse.

Following the protocol used in the PGM paper (except the number of generated images), we compute these metrics on 10,000 generated images for efficiency, rather than the more common 50,000 [16]. When comparing models, we keep the evaluation protocol fixed across all methods.

E.4 Throughput and Latency

We measure inference efficiency using both *latency* and *throughput*. Latency is the wall-clock time required to generate a batch of samples under a fixed sampling budget. Throughput is the amount of generated output per second, reported as tokens/s for language and images/s for image generation. Lower latency and higher throughput are better. We report these metrics after a short warmup phase and average them over repeated runs on a single device. Unless stated otherwise, all models are evaluated with the same batch size, numerical precision, and hardware setup to ensure a fair comparison (which is A100 40 or 80GB and `float64` for sampling text, following [70]). More details regarding these results in Appendix F.11

E.5 Training time and VRAM used

We report training time as the wall-clock time required to complete the corresponding training run, measured from the first optimization step to the final checkpoint. Unless stated otherwise, this does not include offline preprocessing, dataset download, evaluation, or sampling. For post-training stages such as $\mathcal{L}_{\text{CONS}}$, SDTT, or checkpoint adaptation, we report the additional number of training steps separately when relevant, since these stages are short compared to full pretraining.

We report VRAM as the peak GPU memory used during training, measured per GPU and expressed in GiB/GPU. For language modeling, MDLM and FP-MDLM are trained on 8 NVIDIA H200 GPUs. For image modeling, FP-MaskGIT is trained on NVIDIA A100 GPUs. Sampling experiments for both text and images are run on NVIDIA A100 GPUs with either 40GB or 80GB of memory. All training-time and VRAM comparisons are therefore intended to compare models within the same experimental setting and hardware configuration.

F Additional Results

F.1 Extended OWT Generation Results

In this section we detail all the results for FP-MDLM, MDLM and CoFRe on OpenWebText.

F.1.1 Samples

We provide short uncurated samples to complement Gen. PPL and entropy. They illustrate typical low-budget failure modes: local fluency can be reasonable, but generations may drift semantically, repeat entities or phrases, and lose long-range coherence.

Sample 1: repetition and semantic drift

Researchers from the Medical Research Institute of Germany's Centre for Dermatology and Research Center for Brain Research, in Berlin recently, unearthed evidence from more than 50 decades in Berlin, when ARTISTS ate meals on seven different occasions – dishes in toilet tissue drippings; scraps of 22-ounce paper bags and towels; glove boxes, and perfumed socks, wrapped around their bedsits.
Some say that while the press claims that Germans have forged a legend in the Hollywood musical *Waters on the White Strip*, a portrait of Rolling Nirvana frontman Billie got published in *Times Magazine*. Perhaps most astonishing is Jonathan Weber, a native German born to immigrant parents who grew up in Berlin, Germany, earning only $185 for his meals.
The biscuit consists of a sparge of rare, handcrafted alloy of peppers, beef kidneys, pork, pork and oil. "I have got the biscuits from German kitchens," he says. "I also got the food from German restaurants. I even got ramen noodles from my own supermarket."

Failure mode: the sample remains locally grammatical, but drifts between unrelated topics and repeats food/Germany motifs with weak semantic consistency.

Sample 2: fluent fragments with topic drift

<|endoftext|>'s website, wellness.com.com, provides a nurturing health and wellness platform for professionals in the business, health and wellness industries. In the last years we have expanded our wellness Topics Page, post wellness information page on our website and Facebook page and has created a dedicated 24 hour monthly wellness page.
<|endoftext|>Schedstein's favorite musical savior
This Hammerstein's was the most memorable year of the millennium. The celebration of Hammerstein's included an "open mic" festival, performances, movie screenings, and a parade of reviews, all musical from around the world, from MTV to music festivals.
Hopefully this is a forever, this is a this. This is a red. Hopefully it will always be a red. Call it. I can call it that but that warm smile is on. The color of smile. That warm breath is the bottom of "cheeks."

Failure mode: the generation contains fluent local fragments, but abruptly switches topics and degenerates into repetitive, low-information phrasing.

F.1.2 Matched qualitative text samples

We compare generations from MDLM, MDLM+SDTT, FP-MDLM without consistency, FP-MDLM with consistency, and CoFRe at matched transformer-block budgets of 96, 192, 384, and 768. Table 7 shows one sample per configuration and budget, chosen from ten generated samples with a fixed selection procedure and truncated for space. It includes both successful outputs and failure cases. At budget 96, MDLM often loses grammatical and topical coherence, whereas CoFRe is more locally fluent but can still become generic and repetitive (for example, repeated newsletter invitations). At budget 768, generations are more coherent overall, yet individual samples still exhibit repetition, invented entities, or topic drift. These examples motivate reporting entropy and MAUVE alongside Gen PPL rather than treating perplexity as a complete quality measure.

Table 7: **Matched qualitative text samples on OWT.** For each configuration and transformer-block budget, we show one sample out of ten generated with a fixed selection procedure, truncated with [...] for space. PPL is the sample's generative perplexity under GPT-2 Large, and H is its unigram entropy in nats.

Configuration	Sample
MDLM	
Budget 96 H=5.88, PPL=797.6	<|endoftext|> gods. There Articles shouldn't improve. Truly, it is. worse, because those papers are the other half of too the taxpayers U... greatly increased along the growth of funding political interests crisis Darpanas may is born as a virus, but Report was more released it, not the Dan Molle is a random Repair Signarian Reagan He told sort, that if somebody hasa what its 600 max and you run out. He had gone to allow people to hide at the family properties allowing a [...]
Budget 192 H=5.81, PPL=360.0	<|endoftext|> gods. History Articles include: Growing bored $6 a week, moving to New York City. Off the Train... Other topics Kitchen interests You can find me featured speaking... Join a Podcast Report! On this blog, members of the M Anokle Join Newsletter Sign up to a site schedule, registration information. Comments Join Newsletter Subscribe and you agree to web terms provider subscribing to MITEANOKOKER BEATS.NET. The Real Thing Feels the classic Evil witch factor Evil [...]
Budget 384 H=5.68, PPL=158.5	<|endoftext|> common. The vast majority of residents, though bored out with their problems, lack ofers, are now completely involved in the political game. Why the changing focus on political interests? It affects me that speaking to me in my native English speaking and national nationalities, while by to share their own stories, suddenly become part of the political debate. I share more of my stands and what it is coming on you all give me a long way to think about the [...]
Budget 768 H=5.63, PPL=126.1	<|endoftext|> with the Chinese block symbols. So, to be a one side of the European language continuum, but now we all look the same one. Doubt is also the kind of condescension that suggests to me that speaking German and French, like English-words doesn't make it easier to speak in original German language. I am almost think that Europeans sort of respect their European language for what it is. If you were a Zip customer provider you would have all these various languages. [...]
MDLM + SDTT	
Budget 96 H=5.42, PPL=116.3	by peak surprise. What we do do is that someos a background in training, practice and haveate experience, and by radiofrequency radio programs are using professionals and systems for the better than the pros because they are developing, teaching and researching – especially right now all those who really have like an on-set mind. where the researchers in the report was called a room, some of whom are both sides of the country and offered their price are in the form of trade [...]

Configuration	Sample
Budget 192 H=5.36, PPL=58.4	<|endoftext|> The associations between unemotional failure and on-label psychos apt are small, both by satisfaction and outcome, and by assessments of similar, independent health professionals, likely because the sizes of the prosocial failure are small, individual and complex, especially when compared with those who are not with an on-label condition condition, the researchers wrote a report. The women, 80 women from both Massachusettsers' Choice and Medicaid, who are in 95 [. . .]
Budget 384 H=5.37, PPL=43.6	<|endoftext|> The relationship between those delays and failure to re-evaluate is still a mystery for researchers, because the rate of policy change affected by assessments is similar to estimating. That is likely because the assessments are delayed or because they are too late, and the individual is the subject of those assessments, as are others on the PRISM program. The researchers said the agencies should get a policy updated frequently before possible, both offline and [. . .]
Budget 768 H=5.28, PPL=31.3	<|endoftext|> assault, resisting arrest, and failure to provide a false report to a police officer, a police report said. Original story: 10:30 p.m. Later in the night When two police officers saw the individual and picked him up from behind, police said, they saw others on the street who started chasing the officers to gather evidence. The suspect cautiously identified himself, and took him off into the woods where he was in St. Luke's Hospital to be assessed, according the [. . .]
FP-MDLM (no $\mathcal{L}_{\text{CONS}}$)	
Budget 96 H=5.80, PPL=381.9	<|endoftext|> gods. We have to. Yes, to bored deep one, forever, anyway those clues, but so this is gets the family trust... Even then changing focus on this grenade can then make time for me, speaking to me that as a very long fan for basic genres, going Sap the scent of Tricklead has become one of the best shooters. Well, you need to understand that what's going wrong when you lean very closely to provider way to dealing with the various aspects of discovery of defeat. We [. . .]
Budget 192 H=5.78, PPL=282.0	<|endoftext|> gods. We have friends. And, though bored, one of them, right before him, shall now talk most of the world about. The ever changing focus on this issue can then make people like me who speaking of me in my native English buzzwords are apparently supportive, while by reading or reading my stories, suddenly become patron of me from logging. I am more and to understand that what it is coming on you elsewhere very well will compel you to engage with the various [. . .]
Budget 384 H=5.74, PPL=218.0	<|endoftext|> resisted. Women in 1997. Or, to chuckle on the page of the Senate. New York, now the U2 loft in Manhattan. "I mean I don't really think it happened to me that speaking with me or speaking in English is needed even in Africa, but I mean - young people everywhere." I was almost hoping that Europeans sort of understood where this stands and what it means. Did you believe that? I suspect you would have - for various reasons. I don't think museums continue to be [. . .]
Budget 768 H=5.66, PPL=176.5	<|endoftext|> arrived in Ontario on Nov. 29, 1916.Then I have someone who lived in New York and now the U.K. also. Duff and pale, kind of condescending for someone to me who thus escapes me as like poor English wasted people even because it is Ber to say they are such people. I knew that in the Europeans sort of uneasy and skepticism in attitude and it was, if you mean that in those cultures people would not work for various opportunities. I don't know what the society they [. . .]
FP-MDLM + $\mathcal{L}_{\text{CONS}}$	
Budget 96 H=5.43, PPL=97.5	<|endoftext|> reader. Related Articles<|endoftext|>Whether you are becoming bored friends with a psychologist, a psychiatric psychiatrist, or an intelligent person on the autism spectrum. You can now focus on niche interests You can join me in speaking with me with Autism Hygiene Report and for this blog, you the latest news and breakthrough stories about your niche. Sign up to subscribe. Story Break Your Huge Connection! Join Gum, and you will be a great provider way to [. . .]
Budget 192 H=5.39, PPL=69.3	<|endoftext|> industry. The overwhelming majority of residents, to a small percentage of workers, are those who are currently working as part of the industry. Will we continue to focus on political interests? Now let me be speaking with an entertainment industry. I am grateful for this opportunity, seeing that the interests of an industry has suddenly become part of a larger debate. What is this important to discuss and what are the questions that you ask? What do you want [. . .]
Budget 384 H=5.28, PPL=48.9	<|endoftext|> play. Competing causes. Or, to learn from this new generation, to do this, right now, all over the world. For me, as an audience member, I think what happened to me, speaking of me, is that I've been told, consistently, since a young age that this has become patronized and trivialized. I was told that this stands for what it is, and you're doing what you would with all these various aspects of the culture that is never going to be part of the culture. For me it [. . .]
Budget 768 H=5.18, PPL=38.6	<|endoftext|> resisted conventional anarchists and anarchists. Well, to begin with the idea of the free-market, which is exactly what the Anarchists advocated. HERBERGER: I't think it occurred to me that the idea of Anarchism was that anarchism was sort of an anarchist and an anarchist anarchist If anarchism had enough of this idea that there would be a sort of anarchists and anarchist anarchists, and it was only if there was a sort of anarchism that would represent [. . .]

Configuration	Sample
CoFRe (FP-MDLM + $\mathcal{L}_{\text{CONS}}$ + 3SR)	
Budget 96 H=5.47, PPL=101.2	<|endoftext|> gods. Related Articles: Talking to bored friends about mental health, politics, depression, etc. Off the world List... Join us to focus on your interests Become a me of the readers Join a weekly Report for national exclusives, but don't miss our breakthrough stories New Newsletter Sign up to join our Developer's Market Break Newsletter every morning! Join us today and you're be web content provider empowered to download and cover various tips and tricks! Learn [. . .]
Budget 192 H=5.36, PPL=61.9	<|endoftext|> way. We care about handing the keys to the economy to to workers, knowing that the future will forever be changed by the music industry. S: The focus on political interests and thinks prevented me from speaking of the music industry a bit longer. It too early, afraid of the future. The industry has suddenly become one of the best companies. If you don't know what to think about, you'll have to listen to think about the various aspects of the accumulation of [. . .]
Budget 384 H=5.17, PPL=42.3	<|endoftext|> resisted." Mr. Brooks, to present the interview this week, also in New York.<|endoftext|>This post contains the original post. Erm: Because I hear some people thinks of me for speaking of me personally, and I'm telling you, I'm not advocating that you would become a real citizen among Americans. I don't entirely understand what it means that if you're a Muslim you would be a real citizen. I don't think you would be successful in the West. If you're a Christian, [. . .]
Budget 768 H=5.17, PPL=31.5	<|endoftext|> the Communists in New York. Now, tell us about the history of the movement in New York and now the U.S. State. DANIELINE WARNEN: I think the message to me was the idea that New York was revolutionary and was revolutionary as early as the 1930s I think it had a very individualized basis and it was a sort of working-class movement, and it was revolutionary. It was anarchic, and people would look up to various elements of bourgeois politics in the streets, and [. . .]

F.1.3 MDLM vs FP-MDLM

Table 8 compares MDLM and FP-MDLM across fixed transformer-block budgets on OWT. FP-MDLM substantially improves the low-budget regime: at budget 96, generative perplexity drops from 830.82 to 375.63, and at budget 192 from 343.33 to 273.28, while maintaining similar unigram entropy. This shows that replacing part of the denoiser with a fixed-point block improves the quality–cost trade-off when sampling compute is limited. At larger budgets, however, the standard MDLM becomes stronger, suggesting that the base FP-MDLM architecture alone mainly benefits the low-budget regime and requires additional regularization for strong high-budget generation.

Table 9 reports the corresponding validation perplexity and efficiency metrics. FP-MDLM uses substantially fewer parameters than MDLM, reducing the model size from 170M to 104M parameters. It also lowers training time and VRAM usage, from approximately 139h and 112.4 GiB/GPU to approximately 123h and 93.44 GiB/GPU. At equal transformer-block budget, fixed-point sampling is somewhat slower in our current implementation because of solver control flow (Appendix F.11). The validation perplexity is worse than MDLM, which is expected from the reduced parameter count and weight sharing, but the generation results in Table 8 show that this trade-off is favorable under low sampling budgets.

Table 8: Generation quality across compute budgets for MDLM and FP-MDLM on OWT. The budget counts the total number of transformer-block forward passes. Training time and training VRAM are reported in the column headers. For MDLM, the budget is obtained by multiplying the number of denoising steps by 12, corresponding to the 12-layer backbone. FP-MDLM results are reported without reuse.

Budget	**MDLM** 12 transformer layers		**FP-MDLM** no reuse	
	Gen. PPL ↓	**Entropy** ↑	**Gen. PPL** ↓	**Entropy** ↑
96	830.8200	5.9100	375.6314	5.8102
192	343.3300	5.8100	273.2752	5.7630
384	196.7900	5.7500	215.1965	5.7259
768	143.8800	5.7000	179.6546	5.7016
1536	120.7700	5.6700	158.5044	5.6859
3072	112.7000	5.6600	155.8161	5.6858

Table 9: Validation perplexity, training time and VRAM on OpenWebText.

Model	#Params	Val. PPL ↓	Training time (h)	VRAM
OWT (1024)				
MDLM	170M	**23.07**	≈ 139	**112.4** GiB/GPU
FP-MDLM	**104M**	27.45	≈ **123**	**93.44** GiB/GPU

F.1.4 Initialized vs not initialized FP-MDLM adaptation

Table 10 compares 40k-step FP-MDLM adaptation with and without initialization from a pretrained MDLM checkpoint. Pretrained initialization improves the no-reuse setting at every budget, reducing generative perplexity from 296.76 to 276.00 at budget 96 and from 149.63 to 139.00 at budget 768. This indicates that the layer-mapping initialization provides a better starting point for adapting the fixed-point architecture.

The effect is also visible when reuse is enabled. For both initialized and non-initialized models, reuse becomes more beneficial at medium and high budgets, and three-state reuse gives the best results in most of these regimes. With initialization and 3SR, the adapted FP-MDLM reaches the best overall perplexities at budgets 192, 384, and 768. These results suggest that pretrained initialization not only improves the adapted checkpoint itself, but also makes the resulting fixed-point states more reusable across denoising steps.

Table 10: **Pretrained initialization improves short FP-MDLM adaptation.** We compare 40k-step FP-MDLM adaptation with and without initialization from a pretrained MDLM checkpoint. For each sampling budget, we report the best generative perplexity and entropy across denoising-step sweeps under no reuse (NR), full reuse (R), and three-state reuse (3SR). Initialization improves generation quality at every budget and gives the best results with 3SR at medium and high budgets.

			Budget			
Model	**Reuse**	**Metric**	**96**	**192**	**384**	**768**
No init FP-MDLM @40k	NR	Gen PPL ↓	296.764	204.787	170.835	149.626
		Entropy ↑	5.690	5.661	5.629	5.603
	R	Gen PPL ↓	336.057	201.521	161.845	131.800
		Entropy ↑	5.705	5.662	5.637	5.590
	3SR	Gen PPL ↓	298.643	192.227	149.427	131.291
		Entropy ↑	5.722	5.658	5.611	5.597
Init FP-MDLM @40k	NR	Gen PPL ↓	276.003	194.355	163.424	139.003
		Entropy ↑	5.696	5.649	5.622	5.578
	R	Gen PPL ↓	312.912	195.589	149.934	130.443
		Entropy ↑	5.702	5.656	5.617	5.588
	3SR	Gen PPL ↓	286.403	184.708	147.497	126.872
		Entropy ↑	5.695	5.649	5.619	5.572

F.1.5 Generation quality when adding different components of CoFRe

In this section, we ablate the components of CoFRe and analyze how each one affects OWT generation quality. Detailed results are reported in Table 11.

Table 11: **Component ablation for FP-MDLM generation on OpenWebText.** We report generative perplexity and entropy across sampling budgets. The table compares MDLM, the base FP-MDLM checkpoint trained for 1M steps, the same model with inference-time reuse or three-state reuse, the checkpoint after cross-step consistency regularization, and FP-MDLM checkpoints obtained by adapting a pretrained MDLM with a 40k-step teacher-student KL loss on logits.

		Budget			
Method	**Metric**	**96**	**192**	**384**	**768**
MDLM	Gen PPL ↓	830.8200	343.3300	196.7900	143.8800
	Entropy ↑	5.9100	5.8100	5.7500	5.7000
FP-MDLM	Gen PPL ↓	375.6314	273.2752	215.1965	179.6546
	Entropy ↑	5.8102	5.7630	5.7259	5.7016
FP-MDLM + reuse	Gen PPL ↓	516.677	253.210	229.409	269.007
	Entropy ↑	5.815	5.729	5.728	2.209
FP-MDLM + 3SR	Gen PPL ↓	454.100	249.322	196.307	254.258
	Entropy ↑	5.795	5.736	5.663	5.384
FP-MDLM + $\mathcal{L}_{\text{CONS}}$	Gen PPL ↓	104.153	70.275	54.927	41.673
	Entropy ↑	5.447	5.388	5.343	5.156
FP-MDLM + $\mathcal{L}_{\text{CONS}}$+ reuse	Gen PPL ↓	117.592	68.550	50.195	37.567
	Entropy ↑	5.438	5.387	5.283	5.142
FP-MDLM + $\mathcal{L}_{\text{CONS}}$+ 3SR	Gen PPL ↓	101.791	65.182	48.755	37.846
	Entropy ↑	5.434	5.380	5.283	5.142
Adapted FP-MDLM	Gen PPL ↓	276.003	194.355	163.424	139.003
	Entropy ↑	5.696	5.649	5.622	5.578
Adapted FP-MDLM+ reuse	Gen PPL ↓	312.912	195.589	149.934	130.443
	Entropy ↑	5.702	5.656	5.617	5.588
Adapted FP-MDLM+ 3SR	Gen PPL ↓	286.403	184.708	147.497	126.872
	Entropy ↑	5.695	5.649	5.619	5.572

Table 11 decomposes the effect of the main CoFRe components. The base FP-MDLM substantially improves over MDLM in the very low-budget regime, reducing generative perplexity from 830.82 to 375.63 at budget 96. However, this advantage decreases at larger budgets, where the unregularized FP-MDLM remains worse than MDLM. This confirms that the fixed-point architecture alone improves the low-budget quality–cost trade-off, but is not sufficient for strong generation across all budgets.

Reuse alone is unstable on the base FP-MDLM checkpoint. Full reuse and 3SR improve some medium-budget results, but can hurt in other regimes; in particular, full reuse at budget 768 leads to severe degeneration, as reflected by both high generative perplexity and very low entropy. This supports the motivation for adding a regularization mechanism that makes representations smoother and more reusable across denoising steps.

Cross-step consistency is the largest contributor to generation quality. Adding $\mathcal{L}_{\text{CONS}}$ reduces generative perplexity from 375.63 to 104.15 at budget 96 and from 179.65 to 41.67 at budget 768. Once the model is regularized with $\mathcal{L}_{\text{CONS}}$, reuse becomes much more effective: 3SR gives the best results at budgets 96, 192, and 384, while full reuse is slightly better at budget 768. Finally, the adapted FP-MDLM rows show a complementary path to obtaining useful fixed-point denoisers from pretrained MDLM checkpoints with only a short teacher-student adaptation stage.

F.1.6 Mauve score evaluation

Following [53, 64], we evaluate generation quality with MAUVE, which measures the distributional gap between model-generated and human-written text using divergence frontiers [44, 53]. For each model and sampling budget, we compare $5{,}000$ generations against $5{,}000$ OpenWebText reference samples. MAUVE complements generative perplexity because low perplexity alone need not indicate broad distributional coverage [70]. Table 12 reports all six model variants. CoFRe has the highest MAUVE at every budget, including 0.440 at budget 96 and 0.738 at budget 768. This supports a quality–diversity improvement at the selected operating points; the entropy–perplexity frontiers in Appendix F.1.7 give a more detailed comparison.

Table 12: **MAUVE scores for six model variants on OpenWebText.** CoFRe denotes FP-MDLM with cross-step consistency regularization and three-state reuse.

		Budget			
Method	**Metric**	**96**	**192**	**384**	**768**
MDLM	MAUVE ↑	0.022	0.154	0.448	0.649
MDLM + SDTT	MAUVE ↑	0.273	0.592	0.631	0.613
MDLM + $\mathcal{L}_{\text{CONS}}$	MAUVE ↑	0.018	0.084	0.284	0.453
FP-MDLM	MAUVE ↑	0.144	0.224	0.369	0.480
FP-MDLM + $\mathcal{L}_{\text{CONS}}$	MAUVE ↑	0.405	0.667	0.753	0.734
CoFRe	MAUVE ↑	**0.440**	**0.720**	**0.787**	**0.738**

F.1.7 Entropy–perplexity frontiers

Following Pynadath et al. [55], we compare models along entropy–Gen-PPL frontiers rather than at a single operating point. We sweep sampling temperature from 0.80 to 1.20 at budgets 96, 192, 384, and 768 to examine whether low Gen PPL is obtained only by narrowing the output distribution. Figures 10 and 11 show the resulting entropy–perplexity curves for MDLM, MDLM+SDTT, and CoFRe (FP-MDLM with $\mathcal{L}_{\text{CONS}}$ and 3SR). Lower curves indicate better Gen PPL at a given entropy. The fixed-point method improves over MDLM throughout the plotted ranges and is competitive with SDTT at low and medium budgets; at budget 768, SDTT is better at the entropy matches summarized below.

Table 13: **Gen PPL at matched unigram entropy.** We compare the three models at $H = 5.4$ and $H = 5.6$ nats within each budget. Figures 10 and 11 show the full temperature-sweep frontiers.

H **(nats)**	**Budget**	**MDLM**	**MDLM+SDTT**	**CoFRe**
5.4	96	263.6	110.1	**95.5**
5.4	192	117.3	61.5	**59.2**
5.4	384	73.0	55.1	**52.1**
5.4	768	60.2	**40.3**	53.8
5.6	96	392.6	176.3	**160.3**
5.6	192	187.7	105.0	**97.0**
5.6	384	126.4	**89.0**	97.0
5.6	768	108.9	**68.2**	100.4

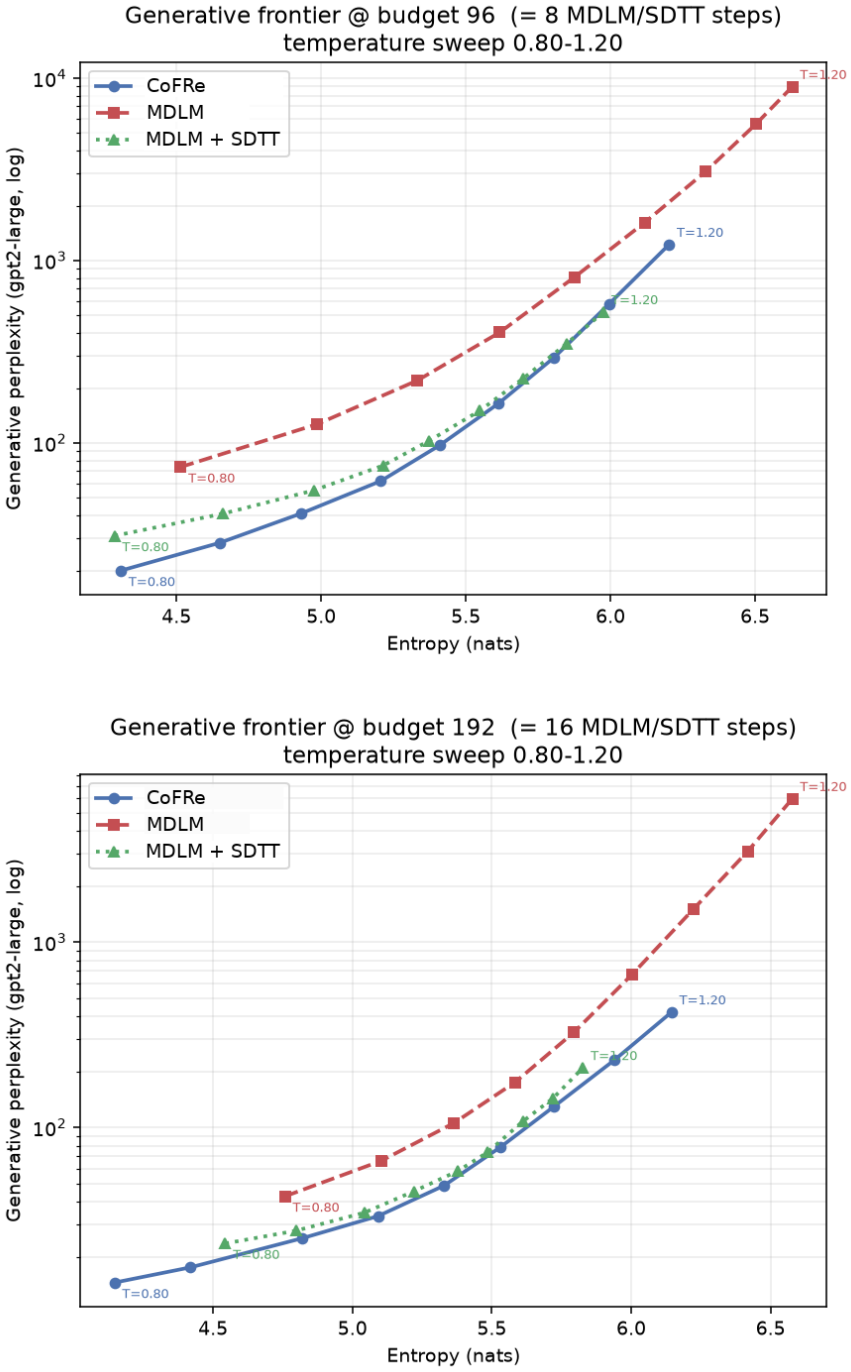


Figure 10: **Entropy–Gen-PPL frontiers at budgets 96 and 192.** Temperature varies from 0.80 to 1.20. The horizontal axis is unigram entropy in nats; lower Gen PPL at a matched entropy is preferable.

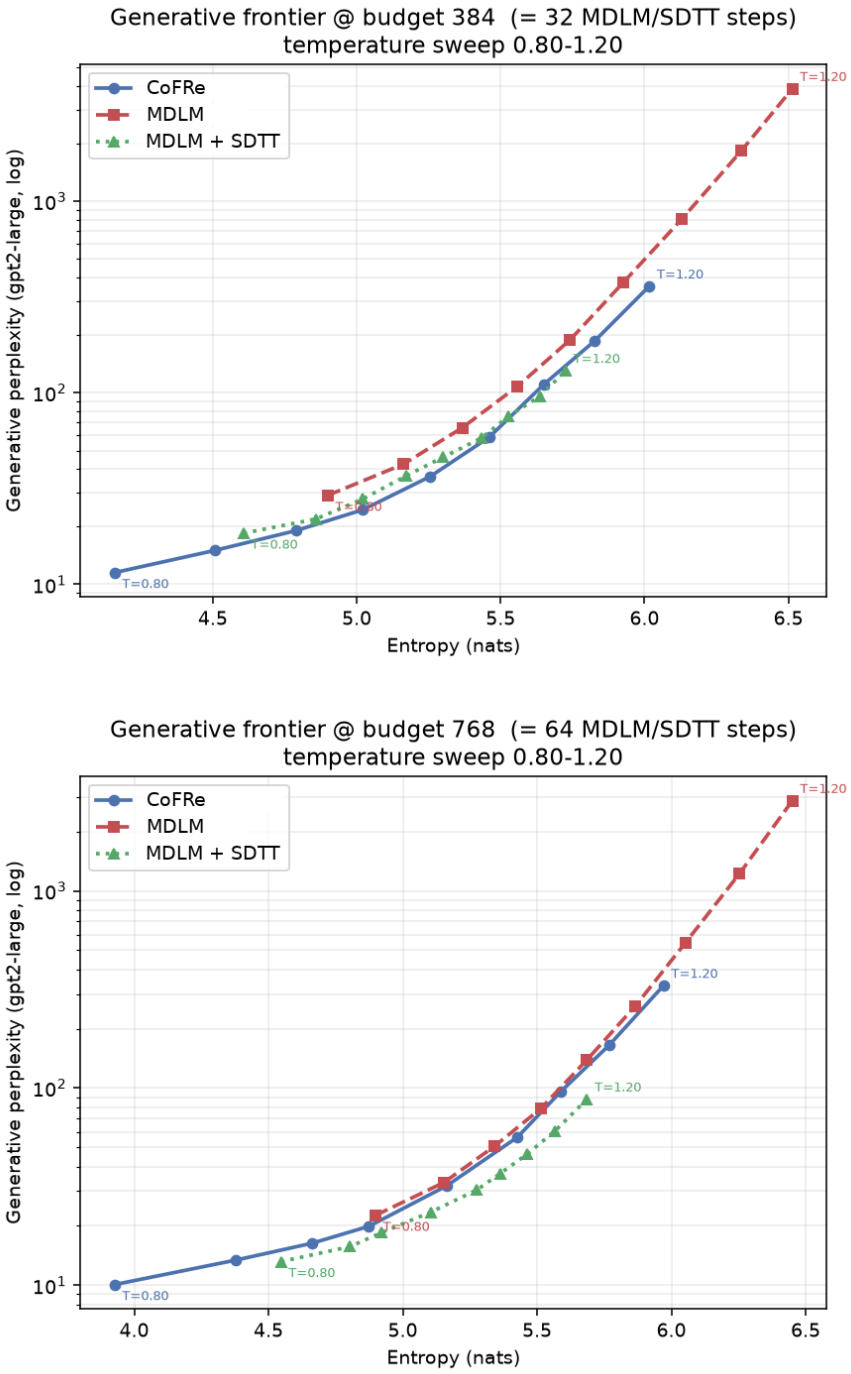


Figure 11: **Entropy–Gen-PPL frontiers at budgets 384 and 768.** The relative advantage over MDLM+SDTT depends on the entropy operating point and does not persist uniformly at the highest budget.

At both entropy targets, CoFRe has lower Gen PPL than MDLM and MDLM+SDTT at budgets 96 and 192. At budget 384 it leads at $H = 5.4$ but not $H = 5.6$, while MDLM+SDTT leads at budget 768. Thus the selected-point Gen-PPL results in Table 2 should be read alongside the full quality–diversity frontier rather than as uniform dominance at every entropy.

F.1.8 Downstream evaluation

Table 14 shows that CoFRe is competitive with MDLM on downstream multiple-choice tasks. It improves on LAMBADA (39.45 vs. 38.52), ARC-easy (36.70 vs. 34.26), and BoolQ (60.21 vs. 49.42), while underperforming on ARC-challenge, OpenBookQA, PIQA, RACE, and SIQA. This mixed behaviour is expected: these tasks are evaluated through likelihood-based answer scoring, so CoFRe does not directly benefit from its main advantage, which appears during low-budget generation. Overall, the results suggest that CoFRe preserves reasonable likelihood-based downstream performance, while its strongest gains remain in the sampling regime.

Table 14: Results on downstream evaluation tasks.

	LAMBADA	**ARC-e**	**ARC-c**	**BoolQ**	**OBQA**	**PIQA**	**RACE**	**SIQA**
MDLM	38.52	34.26	24.66	49.42	28.60	58.27	28.04	38.84
CoFRe	39.45	36.70	22.87	60.21	25.20	55.33	27.27	35.82

F.1.9 Sampling with nucleus sampling

As suggested by Deschenaux et al. [16], Wang et al. [64], nucleus sampling [29] can highly impact the generation of high-quality text sequences. We therefore use this method (with top-p = 0.9) on both MDLM and CoFRe and compare these results in Table 15.

Table 15: **The effect of using nucleus sampling on MDLM and CoFRe generation on OpenWebText.** We report generative perplexity and entropy across sampling budgets, comparing MDLM against CoFRe. Sampling is performed with nucleus sampling with $p = 0.9$, as in [16, 64].

		Budget			
Method	**Metric**	**96**	**192**	**384**	**768**
MDLM	Gen PPL ↓	830.8200	343.3300	196.7900	143.8800
	Entropy ↑	5.9100	5.8100	5.7500	5.7000
MDLM+nucleus	Gen PPL ↓	292.957	119.284	69.479	51.187
	Entropy ↑	5.609	5.538	5.481	5.433
FP-MDLM + $\mathcal{L}_{\text{CONS}}$+ 3SR	Gen PPL ↓	101.791	65.182	48.755	37.846
	Entropy ↑	5.434	5.380	5.283	5.142
FP-MDLM + $\mathcal{L}_{\text{CONS}}$+ 3SR + nucleus	Gen PPL ↓	39.518	30.113	29.199	28.111
	Entropy ↑	5.108	5.057	5.069	5.055

Table 15 shows that nucleus sampling substantially improves generation quality for both MDLM and CoFRe, but also introduces a clear tradeoff. For MDLM, using top-$p = 0.9$ reduces Gen PPL from 830.82 to 292.96 at budget 96, and from 143.88 to 51.19 at budget 768. However, this improvement comes with lower unigram entropy, which decreases from 5.91 to 5.61 at budget 96 and from 5.70 to 5.43 at budget 768. The same pattern is observed for CoFRe: adding nucleus sampling to CoFRe with $\mathcal{L}_{\text{CONS}}$ and 3SR further reduces Gen PPL across all budgets, reaching 28.11 at budget 768, while also lowering unigram entropy. Overall, nucleus sampling improves the Gen PPL-quality tradeoff, but does so by making generations less diverse according to unigram entropy.

F.2 Using consistency loss on MDLM

We trained a vanilla MDLM with the same cross-step hidden-state consistency protocol used for FP-MDLM: the same loss coefficient, training schedule, data, sampling setup, and first-checkpoint rule based on a 15% increase in validation perplexity (Appendix F.7). Table 16 reports three-seed means and standard deviations for this control and the corresponding model variants. The main

quality–cost table (Table 2) instead reports a single run with a different sampling seed, so its values need not coincide with these three-seed means.

Table 16: **Component controls on OWT across three seeds.** Each cell reports mean ± standard deviation for Gen PPL or unigram entropy at a fixed transformer-block budget. The MDLM consistency control uses the same training and checkpoint-selection protocol as the fixed-point models.

Method	Metric	96	192	384	768
MDLM	Gen PPL ↓	839.1 ± 27.4	346.9 ± 11.9	196.1 ± 8.5	143.6 ± 4.4
	Entropy ↑	$5.886 \pm .012$	$5.803 \pm .005$	$5.735 \pm .011$	$5.698 \pm .006$
MDLM + SDTT	Gen PPL ↓	193.05 ± 1.9	$89.2 \pm .1$	$62.3 \pm .4$	$47.0 \pm .1$
	Entropy ↑	$5.588 \pm .011$	$5.538 \pm .016$	$5.499 \pm .015$	$5.459 \pm .009$
MDLM + $\mathcal{L}_{\text{CONS}}$	Gen PPL ↓	979.8 ± 22.9	429.4 ± 17.2	247.6 ± 8.2	181.0 ± 2.9
	Entropy ↑	$5.912 \pm .011$	$5.823 \pm .009$	$5.764 \pm .009$	$5.724 \pm .007$
FP-MDLM	Gen PPL ↓	375.0 ± 6.8	273.5 ± 2.1	216.6 ± 1.4	178.7 ± 8.2
	Entropy ↑	$5.850 \pm .013$	$5.764 \pm .004$	$5.720 \pm .010$	$5.700 \pm .008$
FP-MDLM + $\mathcal{L}_{\text{CONS}}$	Gen PPL ↓	$104.0 \pm .2$	$69.7 \pm .7$	53.4 ± 1.8	42.1 ± 2.2
	Entropy ↑	$5.454 \pm .006$	$5.379 \pm .019$	$5.298 \pm .009$	$5.201 \pm .025$
FP-MDLM + $\mathcal{L}_{\text{CONS}}$ + 3SR	Gen PPL ↓	99.6 ± 1.9	$63.9 \pm .4$	47.8 ± 2.8	$37.8 \pm .9$
	Entropy ↑	$5.490 \pm .012$	$5.438 \pm .018$	$5.342 \pm .031$	$5.208 \pm .018$

The vanilla MDLM consistency control worsens Gen PPL at every budget, from 839.1 to 979.8 at budget 96 and from 143.6 to 181.0 at budget 768. Its entropy is slightly higher, so the degradation is not explained by diversity collapse. The same loss on FP-MDLM reduces Gen PPL from 375.0 to 104.0 at budget 96 and from 178.7 to 42.1 at budget 768; 3SR then improves the seed-mean Gen PPL to 99.6 and 37.8. Thus the consistency objective is not a generic regularizer for a standard feed-forward MDLM. One possible explanation is that aligning independently computed MDLM representations over-constrains features that should change with revealed context, whereas smoothing the representation of an iterated fixed-point denoiser helps its limited-budget solver. The present ablation supports the interaction empirically; it does not prove this explanation.

F.3 Comparison against other distilled or accelerated diffusion language models

We next compare CoFRe against other distilled or accelerated diffusion language models. In particular, we evaluate PGM 6/6, PGM 6/6 with SDTT [16], IDLM-MDLM [41], and our full CoFRe method. For PGM and IDLM-MDLM, we report the effective sampling budget as $12\times$ the number of sampling steps. We emphasize that CoFRe is not primarily designed as a distillation or acceleration method: rather, its goal is to provide a general generation framework that improves sample quality while reducing the effective cost of training and the cost of generation, especially in low-budget regimes. This comparison allows us to assess how CoFRe compares not only to MDLM-based variants, but also to methods that explicitly target faster sampling or improved diffusion language model generation.

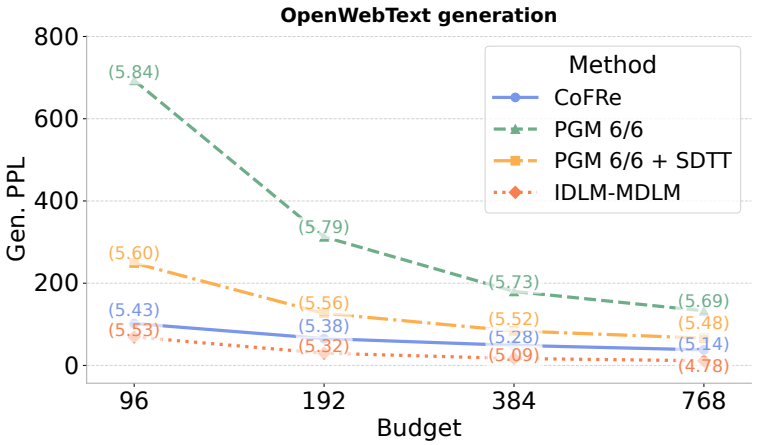


Figure 12: **Generative perplexity as a function of sampling budget on OpenWebText.** We compare CoFRe against PGM 6/6, PGM 6/6 with SDTT, and IDLM-MDLM.

Table 17: **Comparison of CoFRe, PGM, and IDLM-MDLM on OpenWebText.** We report generative perplexity and entropy across sampling budgets. For PGM and IDLM-MDLM, the budget is computed as $12\times$ the number of sampling steps.

		Budget			
Method	**Metric**	**96**	**192**	**384**	**768**
CoFRe FP-MDLM + $\mathcal{L}_{\text{CONS}}$ + 3SR	Gen PPL ↓ Entropy ↑	101.791 5.434	65.182 5.380	48.755 5.283	37.846 5.142
PGM 6/6	Gen PPL ↓ Entropy ↑	693.513 5.843	312.812 5.785	179.928 5.732	132.786 5.688
PGM 6/6 + SDTT	Gen PPL ↓ Entropy ↑	249.076 5.599	126.604 5.561	83.952 5.524	66.316 5.484
IDLM-MDLM	Gen PPL ↓ Entropy ↑	69.165 5.529	29.659 5.323	16.843 5.089	11.312 4.782

Table 17 and Figure 12 show that CoFRe substantially improves over the PGM baselines across all sampling budgets. Compared to PGM 6/6, CoFRe reduces Gen PPL across all budgets. Applying SDTT to PGM gives a strong improvement over the undistilled PGM baseline, reducing Gen PPL from 693.51 to 249.08 at budget 96 and from 132.79 to 66.32 at budget 768. Nevertheless, CoFRe remains better than PGM 6/6 + SDTT at every budget, with relative reductions of approximately 59.1%, 48.5%, 41.9%, and 42.9%.

IDLM-MDLM achieves the lowest Gen PPL across all budgets, reaching 69.17, 29.66, 16.84, and 11.31. Compared to CoFRe, this corresponds to additional reductions of approximately 32.1%, 54.5%, 65.5%, and 70.1%. However, these improvements are accompanied by a stronger reduction in unigram entropy at larger budgets: IDLM-MDLM entropy decreases from 5.53 at budget 96 to 4.78 at budget 768, whereas CoFRe remains between 5.43 and 5.14. Overall, CoFRe provides a strong improvement over accelerated PGM variants while preserving higher entropy than IDLM-MDLM at medium and large budgets, again highlighting the quality-diversity tradeoff among accelerated diffusion language model samplers.

F.4 Extended Image Modeling Results

F.4.1 Samples



Figure 13: Generated samples using CoFRe with a budget of 480.

Matched samples and failure cases. Figures 14 and 15 compare MaskGIT-Large, FP-MaskGIT, and CoFRe on all ten ImageNette classes with the same class label and the same budgets as in Table 2. Samples are not cherry-picked: we show three samples per model and setting without selecting for quality. At budget 48, all three models frequently fail. Samples are fragmented, collage-like scenes with texture artifacts and distorted hands and bodies, and some drift toward related but incorrect objects, such as car dashboards for the cassette-player class. These failures also occur in the fixed-point models, although their low-budget samples more often contain a recognizable object. As the budget grows, samples from all models become class-consistent and structurally coherent, and the visual differences between models become subtle; occasional failures remain at budget 384, such as samples in which the main object is very small or absent. This agrees with Table 18: the fixed-point models mainly gain precision, while recall at the lowest budgets is slightly lower, consistent with less varied low-budget samples.

F.4.2 Image precision and recall

FID and IS do not distinguish greater fidelity from reduced coverage. Table 18 reports precision and recall [40] on ImageNette. All fixed-point variants improve precision over MaskGIT at every budget. For full CoFRe, recall is modestly lower at budgets 48 and 96, but slightly higher at 192 and 384. Thus the low-budget FID gains include a fidelity–coverage trade-off, whereas the medium- and high-budget gains do not require reduced recall in this evaluation.

Table 18: **Precision and recall on ImageNette across sampling budgets.** Higher is better for both metrics.

Method	Metric	48	96	192	384
MaskGIT	Precision ↑	0.2059	0.3615	0.4273	0.4759
	Recall ↑	0.7338	0.6650	0.6637	0.6642
FP-MaskGIT	Precision ↑	0.2265	0.3888	0.4945	0.5422
	Recall ↑	0.7019	0.6703	0.6713	0.6767
FP-MaskGIT + $\mathcal{L}_{\text{CONS}}$	Precision ↑	0.2359	0.3888	0.5057	0.5567
	Recall ↑	0.7047	0.6609	0.6688	0.6773
CoFRe	Precision ↑	0.2339	0.3890	0.5070	0.5569
	Recall ↑	0.6994	0.6550	0.6762	0.6788



Figure 14: **Matched ImageNette samples for MaskGIT-Large, FP-MaskGIT, and CoFRe (classes 1–5).** Each row shows three samples (not cherry-picked) for a given model at budgets 48, 96, 192, and 384 transformer-block forward passes, with the same class label and sampling setup. Low-budget samples from all models show failure modes such as fragmented object shapes, texture artifacts, distorted bodies, and drift toward related but incorrect objects.



Figure 15: **Matched ImageNette samples for MaskGIT-Large, FP-MaskGIT, and CoFRe (classes 6–10).** Same layout as Figure 14.

F.4.3 MaskGIT-Large vs MaskGIT-12 vs FP-MaskGIT

Table 19: Generation quality vs. compute budget, where the budget counts the total number of transformer-block forward passes. Training time and training VRAM are reported in the column headers; percentages indicate reductions relative to MaskGIT-Large. Models are trained on Imagenette (10-class subset of ImageNet) for 50k training steps. MaskGIT-Large uses 24 transformer layers, MaskGIT-12 uses 12 transformer layers, and FP-MaskGIT uses 4 pre- and 4 post-transformer layers. All models use a global batch size (GBS) of 128. For 24-layer MaskGIT-Large, a budget of 48 corresponds to 2 decoding steps (2×24), while for 12-layer MaskGIT-12 it corresponds to 4 decoding steps (4×12).

Budget	**MaskGIT-Large** 24 transformer layers Train: 17h 46m VRAM: 72.45 GiB		**FP-MaskGIT** 4 pre / 4 post layers Train: 9h 08m (-48.6%) VRAM: 35.74 GiB (-50.7%)		**MaskGIT-12** 12 transformer layers Train: 7h 40m (-56.8%) VRAM: 28.14 GiB (-61.2%)	
	FID ↓	**IS** ↑	**FID** ↓	**IS** ↑	**FID** ↓	**IS** ↑
48	174.0856	9.2860	**100.0964**	**15.0196**	128.9871	12.6302
96	117.6439	13.3696	**57.5823**	15.8495	77.0653	**15.9587**
192	54.6172	**16.0220**	**32.0072**	15.0822	50.9830	15.3803
240	44.4930	**15.6458**	**27.6470**	14.6582	45.6384	15.2925
384	30.0202	14.6473	**24.1946**	14.4567	38.0946	**15.2823**
480	27.3975	**14.2319**	**22.1171**	13.8718	32.8731	14.1865

F.4.4 Generation quality when adding different components of CoFRe

Table 20: **Component ablation for FP-MaskGIT generation on ImageNette.** We report FID and Inception Score across sampling budgets. The table compares MaskGIT baselines, the FP-MaskGIT backbone with different reuse variants, and FP-MaskGIT with cross-step consistency regularization. CoFRe corresponds to the consistency-trained FP-MaskGIT model with reuse.

		Budget			
Method	**Metric**	**48**	**96**	**192**	**384**
MaskGIT-Large	FID ↓	174.0856	117.6439	54.6172	30.0202
	IS ↑	9.2860	13.3696	16.0220	14.6473
MaskGIT-12	FID ↓	128.9871	77.0653	50.9830	38.0946
	IS ↑	12.6302	15.9587	15.3803	15.2823
FP-MaskGIT no reuse	FID ↓	100.0964	57.5823	32.0072	24.1946
	IS ↑	15.0196	15.8495	15.0822	14.4567
FP-MaskGIT + reuse	FID ↓	101.5056	55.2582	31.7141	24.0396
	IS ↑	14.4204	15.5139	14.5876	13.9587
FP-MaskGIT + 3SR	FID ↓	102.3251	54.4212	31.2308	23.5458
	IS ↑	14.432	15.4369	14.912	14.0228
FP-MaskGIT + $\mathcal{L}_{\text{CONS}}$+ no reuse	FID ↓	107.5012	56.7336	31.3828	24.6839
	IS ↑	13.3413	16.3229	15.0487	14.0886
CoFRe FP-MaskGIT + $\mathcal{L}_{\text{CONS}}$ + reuse	FID ↓	96.7331	51.0077	27.6242	22.8381
	IS ↑	14.4074	15.9572	15.0213	14.2128

F.5 Component Ablation for FP-MGM

We use small-scale proxy runs to select the main FP-MGM design choices before running the full experiments. For language, we ablate the number of explicit preprocessing/postprocessing layers, the number of fixed-point iterations used with and without gradients, the learning rate, and gradient clipping. Unless stated otherwise, these ablations are run for 100k training steps and are intended to compare configurations under the same compute setting, not to match the final full-scale results.

Language-modeling ablations on LM1B. Table 21 reports FP-MDLM ablations on LM1B without sentence packing. The main trend is that very shallow stochastic fixed-point training is cheapest, while larger stochastic solver budgets improve perplexity at moderate cost. Using a deterministic large solver budget, $\mathcal{U}(12, 12)$ without gradients and $\mathcal{U}(12, 12)$ with gradients, gives the best validation perplexity among FP-MDLM variants, but it is substantially slower and more memory-intensive. The stochastic setting $\mathcal{U}(0, 4)$ without gradients and $\mathcal{U}(3, 6)$ with gradients gives the best FP-MDLM test perplexity, and provides a better cost–quality trade-off. We therefore use this stochastic solver setting as the default for the larger FP-MDLM runs.

Table 21: **FP-MDLM solver and architecture ablation on LM1B.** Models are trained for 100k steps without sentence packing. $\mathcal{U}(a, b)$ denotes a discrete uniform distribution over solver iterations. We report validation perplexity and test perplexity on 500 samples.

Model	Pre/Post	No-grad	Grad	LR	Clip	Time	VRAM	Val. PPL ↓	Test PPL ↓
FP-MDLM	2/2	$\mathcal{U}(0, 2)$	$\mathcal{U}(1, 3)$	1×10^{-4}	0.5	8h05	36.0GB	50.66	48.057
FP-MDLM	2/2	$\mathcal{U}(0, 4)$	$\mathcal{U}(3, 6)$	1×10^{-4}	0.5	9h53	41.0GB	46.54	**44.885**
FP-MDLM	2/2	$\mathcal{U}(12, 12)$	$\mathcal{U}(12, 12)$	1×10^{-4}	0.5	16h49	56.2GB	42.50	45.136
FP-MDLM	1/1	$\mathcal{U}(0, 2)$	$\mathcal{U}(1, 3)$	1×10^{-4}	0.5	**7h05**	**30.7GB**	47.37	50.740
FP-MDLM	1/1	$\mathcal{U}(0, 4)$	$\mathcal{U}(3, 6)$	1×10^{-4}	0.5	8h53	36.18GB	44.60	46.620
FP-MDLM	1/1	$\mathcal{U}(0, 3)$	$\mathcal{U}(2, 4)$	1×10^{-4}	0.5	7h52	34.4GB	49.19	47.690
FP-MDLM	1/1	$\mathcal{U}(0, 4)$	$\mathcal{U}(3, 6)$	2×10^{-4}	0.5	8h51	36.18GB	44.06	45.926
FP-MDLM	1/1	$\mathcal{U}(0, 4)$	$\mathcal{U}(3, 6)$	1×10^{-4}	1.0	8h50	36.18GB	44.60	46.773
FP-MDLM	1/1	$\mathcal{U}(0, 4)$	$\mathcal{U}(3, 6)$	2×10^{-4}	1.0	8h50	36.18GB	42.795	45.175
MDLM DiT	–	–	–	3×10^{-4}	–	11h02	44.0GB	**42.24**	48.522

Language-modeling ablations on OWT. Table 22 shows the corresponding proxy experiment on OWT with sequence length 128. The FP-MDLM variants use substantially fewer parameters than the MDLM DiT baseline. Among the FP-MDLM variants, configuration B, with learning rate 2×10^{-4} and gradient clipping 1.0, gives the best validation and test perplexity. It also improves generative perplexity over configuration A across all reported sampling budgets. MDLM remains stronger in likelihood at this small scale, but FP-MDLM is competitive in low-budget generation and uses fewer parameters. For FP-MDLM, we use $K_{\text{pre}} = 1$, $K_{\text{fp}} = 1$, and $K_{\text{post}} = 1$.

Table 22: **FP-MDLM proxy ablation on OWT with sequence length 128.** Models are trained for 100k steps. Generation cells report generative perplexity / unigram entropy.

Model	Pre/Post	LR	Clip	Params	Val. PPL ↓	Test PPL ↓	B128	B64	B32	B16	B8
FP-MDLM A	1/1	1×10^{-4}	0.5	104M	58.98	55.390	185.42 / 4.32	216.24 / 4.33	263.59 / 4.34	400.29 / 4.38	744.83 / 4.41
FP-MDLM B	1/1	2×10^{-4}	1.0	104M	53.278	50.287	170.79 / 4.31	201.23 / 4.32	230.00 / 4.34	380.00 / 4.38	637.00 / 4.41
MDLM DiT	–	3×10^{-4}	1.0	169M	**47.10**	**45.211**	**133.15** / 4.29	**150.62** / 4.31	**193.946** / 4.33	**285.50** / 4.36	**552.71** / 4.38

Image-generation ablation. For image generation, we use the same fixed-point replacement idea inside MaskGIT and compare the resulting FP-MaskGIT to fixed-depth MaskGIT baselines. Table 23 reports the completed ImageNette runs. FP-MaskGIT with a 4/4 explicit pre/post architecture substantially reduces training cost relative to MaskGIT-Large, from 17h46m to 9h08m and from 72.45GiB to 35.74GiB. It also improves FID at all reported budgets. Compared with the smaller 12-layer MaskGIT baseline, FP-MaskGIT is more expensive to train but gives better FID across the reported budgets, supporting the choice of the 4/4 FP-MaskGIT configuration for the main image experiments. For FP-MaskGIT, we use $K_{\text{pre}} = 4$, $K_{\text{fp}} = 1$, and $K_{\text{post}} = 4$.

Table 23: **Architecture ablation for FP-MaskGIT on ImageNette.** We compare MaskGIT baselines against FP-MaskGIT with different numbers of explicit preprocessing and postprocessing layers. Smaller FP-MaskGIT variants perform progressively worse than the 4/4 configuration. For each model, we report FID and IS across sampling budgets.

Model	Metric	48	96	192	240	384	480
MaskGIT-Large	FID ↓	174.0856	117.6439	54.6172	44.4930	30.0202	27.3975
	IS ↑	9.2860	13.3696	**16.0220**	**15.6458**	14.6473	**14.2319**
MaskGIT-12	FID ↓	128.9871	77.0653	50.9830	45.6384	38.0946	32.8731
	IS ↑	12.6302	**15.9587**	15.3803	15.2925	**15.2823**	14.1865
FP-MaskGIT 1/1	FID ↓	122.1347	72.4871	46.6109	41.0631	34.2790	30.2190
	IS ↑	12.6696	13.5025	12.8678	12.5092	12.1858	11.8640
FP-MaskGIT 2/2	FID ↓	114.9546	67.6721	41.7999	36.8044	30.7165	27.8659
	IS ↑	13.4492	14.2838	13.6072	13.1163	13.1199	12.3832
FP-MaskGIT 3/3	FID ↓	106.9307	62.4781	36.6818	32.4145	27.4128	24.4974
	IS ↑	14.3704	15.0247	14.3022	13.8891	13.8275	13.2326
FP-MaskGIT 4/4	FID ↓	**100.0964**	**57.5823**	**32.0072**	**27.6470**	**24.1946**	**22.1171**
	IS ↑	**15.0196**	15.8495	15.0822	14.6582	14.4567	13.8718

Overall, these ablations motivate the default FP-MGM recipe used in the main experiments: a stochastic fixed-point solver rather than a fully deterministic deep solve, a compact explicit pre/post architecture for language, and a 4/4 explicit pre/post architecture for image generation. The proxy results also show the main trade-off that carries through the full experiments: fixed-point denoisers reduce parameter and memory cost, but require careful solver-budget and reuse choices to obtain strong generation quality.

F.6 Cross-step consistency recovers low-budget generation quality

In masked generative models, the loss is typically applied only to final predictions at masked positions, so intermediate states are supervised only indirectly. A common extension is *consistency* regularization, which aligns predictions or hidden states across nearby denoising steps. This can be done in output space, for example with KL on logits, or in representation space, for example with MSE or cosine similarity on hidden states. Targets may come from the same model with stop-gradient or from a teacher such as an EMA copy. This adds auxiliary supervision between denoising states while leaving the original masked modeling objective unchanged.

A related idea is self-distillation through time (SDTT), which trains a student with fewer denoising steps to match a teacher with more steps. If $p_\theta^{(m)}$ is generation with m steps and $p_\nu^{(k)}$ the student model with $k < m$, SDTT minimizes $\mathbb{E}_{\mathbf{z}_0\sim\mathcal{D},\,\mathbf{z}_t\sim q_t(\mathbf{z}_t|\mathbf{z}_0)}\,\delta\big(\mathbf{x}_\nu(\mathbf{z}_t,t),\tilde{\mathbf{x}}_\theta^{\text{teacher}}(\mathbf{z}_t,t,m/k)\big)$, where δ is typically a divergence such as KL, sg denotes stop-gradient, and $\tilde{\mathbf{x}}_\theta^{\text{teacher}}$ aggregates teacher predictions over multiple steps. After training, one student step can approximate several teacher steps. Consistency regularization thus provides local temporal supervision, whereas SDTT directly compresses the denoising process.

Experimental Settings For the cross-step consistency regularization experiments, we compare two FP-MDLM models with the same 1M-step base checkpoint, followed by a 30k consistency post-training stage. The baseline is trained from scratch using only the MDLM loss. The consistency model starts from the same training setup, but adds a consistency objective late in training. We explore several consistency objectives and found the best results with $\mathcal{L}_{\text{MSE}}$ introduced in Sec. 3.2; the comparison is reported in Appendix F.8. Specifically, the consistency weight is introduced at step 1M, increased linearly from 0 to 0.1 over 5k steps, and then kept at 0.1 until step 1030000. We evaluate both models using GPT-2 Large generative perplexity and sample entropy, and report results both with and without solution reuse.

Results Table 11 shows that cross-step consistency regularization is the main driver of generation quality. Without reuse, adding $\mathcal{L}_{\text{CONS}}$ improves generative perplexity at every budget, from 375.6 to 104.2 at budget 96 and from 179.7 to 41.7 at budget 768. This is a large improvement over the baseline FP-MDLM, while maintaining non-degenerate entropy across all budgets.

Reuse becomes useful once the model has been regularized with $\mathcal{L}_{\text{CONS}}$. In the baseline checkpoint, reuse and 3SR are inconsistent and can hurt generation quality, especially at the largest budget. After consistency regularization, however, reuse further improves the low-perplexity model at budgets 192, 384, and 768. The best results are obtained with 3SR at budgets 96, 192, and 384, reaching generative perplexities of 101.8, 65.2, and 48.8, while standard reuse is slightly better at budget 768 with 37.6. **Overall, $\mathcal{L}_{\text{CONS}}$ turns FP-MDLM into a much stronger generator, and reuse provides additional gains once the fixed-point denoiser has been regularized.**

Lagged logit analysis. To understand the improvement from $\mathcal{L}_{\text{CONS}}$, we compare masked-token logits at a student denoising step s with logits at cleaner future steps $s + \ell$ on shared contexts. The consistency-trained model has lower lagged logit KL than the baseline across all sampling-step, solver-budget, and lag settings, reducing the KL by 15.2% on average. The effect is strongest for nearby denoising states, with a 19.0% average reduction at lag 1, and remains positive through lag 4. This suggests that $\mathcal{L}_{\text{CONS}}$ acts as a cross-time self-distillation signal: it makes student-step logits more aligned with cleaner future predictions, which helps explain the large generative perplexity gains in Tables 11 and 20.

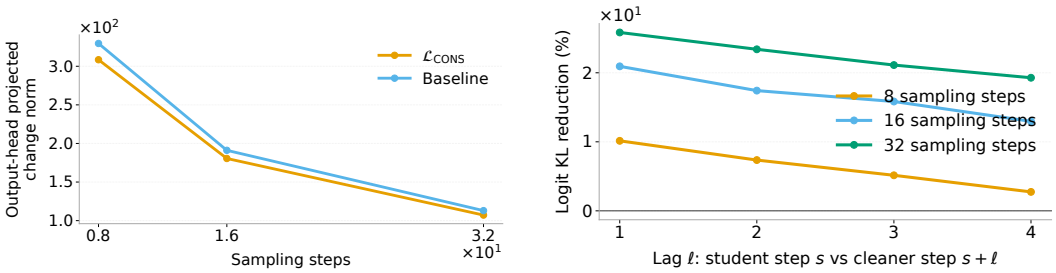


Figure 16: **Lagged logit analysis.** (Left) output-head-projected hidden-state changes decrease as the number of sampling steps increases, for both the baseline and the $\mathcal{L}_{\text{CONS}}$ model. (Right) relative reduction in lagged logit KL from $\mathcal{L}_{\text{CONS}}$ compared to the baseline, measured between a student denoising step s and a cleaner future step $s + \ell$. The consistency-trained model reduces lagged logit KL across lags and sampling-step settings, with the strongest gains at smaller lags.

F.7 Consistency Loss Training Dynamics

In practice, we find that extending the consistency stage for too long can degrade generation quality by over-sharpening the model. We therefore use validation perplexity as an early stopping signal: starting from the pre-$\mathcal{L}_{\text{CONS}}$ checkpoint, we select the first checkpoint whose validation perplexity exceeds the pre-$\mathcal{L}_{\text{CONS}}$ value by 15%. This rule gave the best empirical trade-off between generative perplexity and entropy. We linearly warm up the consistency term at the beginning of post-training to avoid an abrupt change in the optimization objective.

Figure 17 illustrates this behavior. Generative perplexity improves rapidly during early consistency training, but later checkpoints become increasingly over-sharpened, as shown by the sharp drop in sample entropy. The validation perplexity curve is not monotonic: after first increasing beyond the 15% threshold, it can later decrease again. However, these later decreases do not correspond to recovered sample diversity, so selecting a later checkpoint based only on validation perplexity would be misleading. We therefore use the first threshold crossing as a conservative stopping rule. This is qualitatively similar to the caution in SDTT [15], where repeated distillation rounds can accumulate approximation error because each student becomes the next teacher; in both cases, validation metrics and sample diversity should be monitored rather than relying only on the distillation loss.

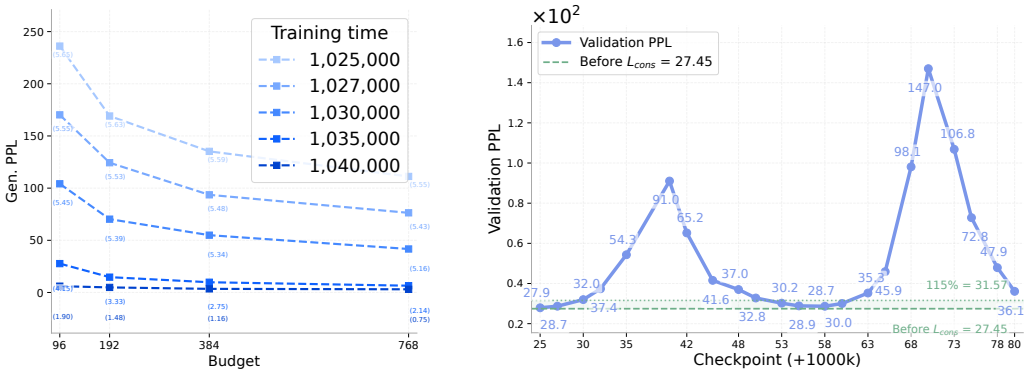


Figure 17: **Checkpoint selection for the $\mathcal{L}_{\text{CONS}}$ post-training stage.** (Left) Generative perplexity across budgets improves rapidly during early consistency training, but later checkpoints over-sharpen the model, as reflected by collapsing entropy values shown in parentheses. Sampling is done without warm-start (e.g. no reuse) (Right) Validation perplexity is not monotonic: it first rises above the 15% threshold, then can decrease again at later checkpoints. Since these later decreases do not recover sample diversity, we use the first checkpoint whose validation perplexity exceeds the pre-$\mathcal{L}_{\text{CONS}}$ value by 15% as our stopping rule. This empirically balances generation quality and sample diversity.

F.8 Ablations on the Consistency Loss Type

We ablate the form of the consistency objective for FP-MaskGIT on ImageNette. Starting from the same FP-MaskGIT checkpoint, we compare output-space consistency with KL on logits, representation-space consistency with MSE or cosine distance, and variants that use an EMA teacher. We also test whether adding this consistency loss at the beginning of the training improves the final model. For each loss, we evaluate both no-reuse and fixed-point reuse at the same sampling budgets.

Table 24 shows that representation-space losses are generally more effective than output-space KL for improving generation quality. In particular, hidden-state MSE with reuse gives the best FID at budgets 48 and 192, while hidden-state cosine with EMA gives the best FID at budget 96. These results support our choice of hidden-state consistency as the main regularizer: it provides a stronger training signal for the fixed-point representation than directly matching logits, and it interacts well with reuse during sampling.

After identifying the three best loss configurations on FP-MaskGIT, we transfer them to FP-MDLM, 120k training steps at 128 length, to test whether they also work in a different modality. We first run this ablation on FP-MaskGIT because it is cheaper to train and evaluate than FP-MDLM, and because its image-generation metrics are more direct than proxy metrics such as generative perplexity and unigram entropy. These configuration are: Hidden state MSE (no EMA), Hidden state cosine (with EMA), Hidden state MSE (with EMA) (Table 25). Once we found the best configuration on FP-MDLM, 120k training steps at 128 length, we test if it extends to longer sequences, i.e. 1024.

Table 24: Ablation on the consistency loss type for FP-MaskGIT.

Method	Reuse	Metric	Budget 48	96	192
Baseline FP-MaskGIT + 10k no KL loss	No reuse	FID ↓	102.3041	56.9086	32.3490
		IS ↑	14.6703	15.9080	15.0480
	With reuse	FID ↓	101.5056	55.2582	31.7141
		IS ↑	14.4204	15.5139	14.5876
Cross-step logit KL no EMA teacher	No reuse	FID ↓	101.7078	55.8024	31.4500
		IS ↑	14.5171	16.0273	14.8362
	With reuse	FID ↓	98.4032	53.7002	29.4128
		IS ↑	14.5321	16.0719	15.0317
Cross-step logit KL EMA teacher	No reuse	FID ↓	102.2678	56.5554	31.2910
		IS ↑	14.0976	16.0031	14.8055
	With reuse	FID ↓	100.4105	54.2807	29.7377
		IS ↑	14.2942	15.9352	14.5184
Latent proj cosine no EMA	No reuse	FID ↓	112.2268	55.3464	30.5902
		IS ↑	13.0748	15.9799	15.0260
	With reuse	FID ↓	102.4396	54.3828	30.4941
		IS ↑	14.2156	15.7187	14.9456
Latent proj cosine with EMA	No reuse	FID ↓	110.7216	54.8109	32.3299
		IS ↑	13.1563	15.6277	14.9067
	With reuse	FID ↓	102.5607	53.8555	30.6918
		IS ↑	14.0288	15.5334	14.9674
Hidden state MSE no EMA	No reuse	FID ↓	107.5012	56.7336	31.3828
		IS ↑	13.3413	**16.3229**	15.0487
	With reuse	FID ↓	**97.8401**	51.0077	**27.6242**
		IS ↑	**15.0866**	16.2692	14.5894
Hidden state MSE with EMA	No reuse	FID ↓	110.4611	54.2080	30.6044
		IS ↑	13.4316	15.5567	14.9206
	With reuse	FID ↓	97.9066	51.1808	28.3367
		IS ↑	14.6514	16.1668	14.5935
Hidden state cosine no EMA	No reuse	FID ↓	110.4582	56.6083	31.1203
		IS ↑	13.7846	15.8706	14.9106
	With reuse	FID ↓	98.7089	54.5677	29.5129
		IS ↑	14.4814	15.7316	14.4452
Hidden state cosine with EMA	No reuse	FID ↓	112.5791	54.7753	29.7427
		IS ↑	13.0173	15.9610	14.9954
	With reuse	FID ↓	98.3099	**50.2515**	28.7888
		IS ↑	14.4018	15.4198	14.7025
Pretraining + hidden state MSE without EMA teacher	No reuse	FID ↓	114.1953	57.1398	31.8960
		IS ↑	14.0358	16.1214	**15.4513**
	With reuse	FID ↓	108.5110	50.3101	28.9113
		IS ↑	14.4205	16.0090	14.8960
Pretraining + hidden state MSE with EMA teacher	No reuse	FID ↓	116.0607	56.6419	30.7830
		IS ↑	13.3924	16.1979	15.0050
	With reuse	FID ↓	108.8888	51.9627	27.8129
		IS ↑	14.3478	15.9366	14.7259
Pretraining + hidden state cosine without EMA teacher	No reuse	FID ↓	112.3853	55.5995	32.6594
		IS ↑	13.6743	15.9413	15.1818
	With reuse	FID ↓	104.7088	53.0664	31.0667
		IS ↑	14.5433	15.9216	15.0648
Pretraining + hidden state cosine with EMA teacher	No reuse	FID ↓	118.5230	57.9347	32.6617
		IS ↑	13.3390	16.1273	15.2384
	With reuse	FID ↓	110.3087	53.6505	30.9416
		IS ↑	13.9843	16.0756	15.3464

Table 25: Generation quality across budgets on OWT, 120k training steps, 128 sequence length. Baseline is 120k training steps. Consistency post training uses 100k pretraining steps followed by 20k post-training adaptation steps.

Method	Reuse	Metric	Budget 96	192	384	768
Baseline FP-MDLM @120k	No reuse	Gen PPL ↓	421.5819	337.1057	262.6982	245.1979
		Entropy ↑	4.4004	4.3862	4.3688	4.3571
	With reuse	Gen PPL ↓	425.2954	310.0746	253.1428	225.9641
		Entropy ↑	4.4033	4.3931	4.3651	4.3455
Hidden state MSE w/o EMA teacher	No reuse	Gen PPL ↓	433.9153	338.9548	274.9060	227.5296
		Entropy ↑	4.4047	4.3851	4.3782	4.3658
	With reuse	Gen PPL ↓	385.6467	266.6036	226.6500	215.4439
		Entropy ↑	4.3992	4.3763	4.3668	4.3705
Hidden state cosine w/ EMA teacher	No reuse	Gen PPL ↓	435.8042	334.9215	277.8421	243.6187
		Entropy ↑	4.4051	4.3832	4.3791	4.3618
	With reuse	Gen PPL ↓	418.2375	306.8842	249.7164	225.9032
		Entropy ↑	4.4014	4.3841	4.3681	4.3673
Hidden state MSE w/ EMA teacher	No reuse	Gen PPL ↓	436.9225	332.7336	279.9604	251.2712
		Entropy ↑	4.4063	4.3819	4.3804	4.3579
	With reuse	Gen PPL ↓	426.5467	317.1626	254.6232	228.4749
		Entropy ↑	4.4022	4.3878	4.3692	4.3661

Matched output-space control on FP-MDLM. We also compare the hidden-state MSE objective against cross-step KL on output distributions on the same FP-MDLM backbone with 3SR. Table 26 reports the results. Hidden-state MSE has lower Gen PPL at all four budgets, while entropy is similar. This comparison isolates the objective form; it is not a system-level speed or quality comparison against CDLM's block-causal architecture and KV caching.

Table 26: **Hidden-state versus output-distribution consistency on FP-MDLM with 3SR.** Both objectives use the same fixed-point backbone and sampling setup.

Objective	Metric	96	192	384	768
Hidden-state MSE	Gen PPL ↓	**101.791**	**65.182**	**48.755**	**37.846**
	Entropy ↑	5.434	5.380	5.283	5.142
Output-distribution KL	Gen PPL ↓	104.4	75.6	58.9	45.3
	Entropy ↑	5.432	5.370	5.290	5.151

F.9 Tradeoff Between Denoising Steps and Fixed-Point Iterations

This sweep studies how generation quality changes when compute is allocated differently between denoising steps and fixed-point iterations. We measure this tradeoff on FP-MDLM + $\mathcal{L}_{\text{CONS}}$ + 3SR, which corresponds to our CoFRe model. The heatmap shows that these two sources of compute are not interchangeable: using very few denoising steps gives poor generative perplexity even when many FP iterations are used, while increasing the number of denoising steps substantially improves sample quality. Once enough denoising steps are used, additional FP iterations still help, but with smaller gains. The entropy heatmap shows that the best generative perplexity values are not obtained by collapsing sample diversity, since entropy remains in a similar non-degenerate range across the strongest configurations. Overall, the sweep indicates that the quality–diversity tradeoff depends on how inference compute is split between outer denoising and inner fixed-point solving, supporting the use of non-uniform depth allocation at sampling time.

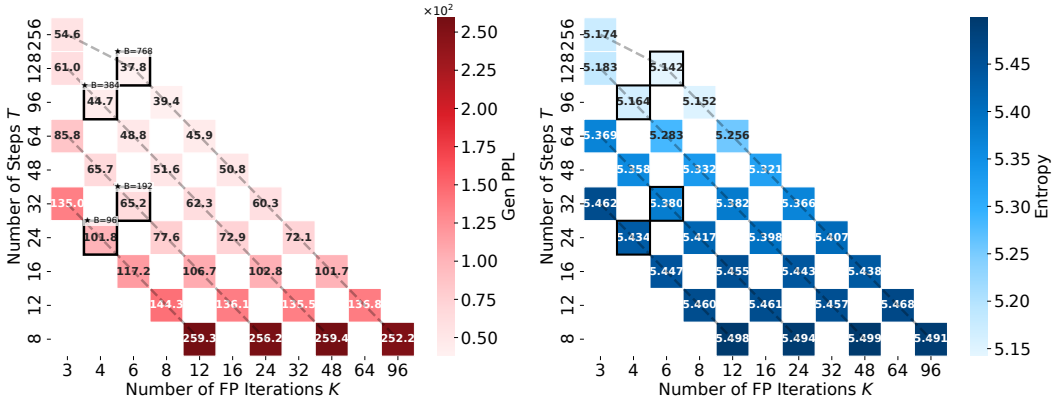


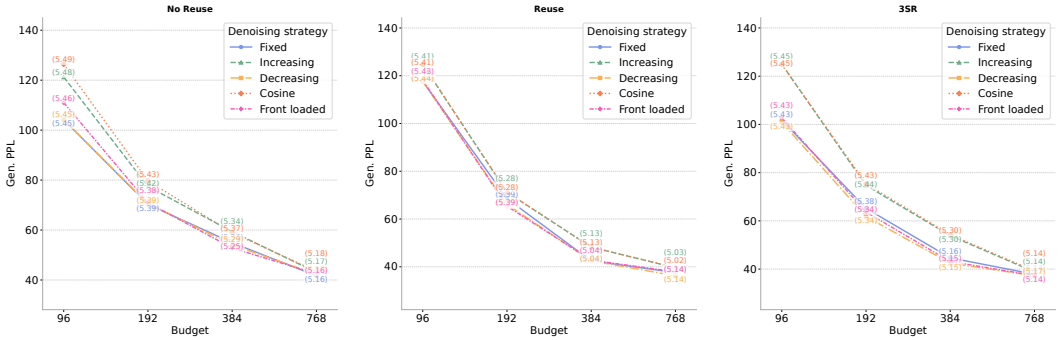
Figure 18: **Tradeoff between denoising steps and fixed-point iterations.** We sweep the number of denoising steps and FP iterations for FP-MDLM with consistency regularization and three-state reuse. The left heatmap reports generative perplexity, where lower is better, and the right heatmap reports sample entropy. Each annotated cell corresponds to one evaluated allocation; blank cells were not evaluated. Allocating compute to very few denoising steps leads to poor generative perplexity even with many FP iterations, whereas configurations with sufficiently many denoising steps and moderate FP depth achieve the best quality while maintaining non-collapsed entropy.

F.10 Different budget allocation strategies

We also ablate how the fixed-point iteration budget should be distributed across denoising steps for FP-MDLM+L_{CONS}. Keeping the checkpoint and total forward-pass budget fixed, we compare five allocation strategies: fixed, decreasing, increasing, cosine, and front-loaded schedules. We evaluate each schedule under the three initialization regimes used throughout the paper: no reuse, full reuse, and 3SR. Table 27 and Figure 19 show that decreasing schedules perform best overall, with the lowest mean generative perplexity, the best average rank, and the most wins across initialization-budget settings. Fixed allocation is a strong second, while increasing and cosine schedules consistently underperform. This suggests that FP-MDLM benefits from spending more fixed-point computation early in the denoising trajectory, when the sequence is most corrupted and the denoising problem is hardest.

Table 27: Overall comparison of denoising strategies. Lower Gen. PPL and lower average rank are better. Wins count the number of init-budget settings where the strategy obtains the best Gen. PPL. More than 12 wins are possible in total because ties count as wins for all tied strategies.

Strategy	Mean Gen. PPL ↓	Avg. rank ↓	Wins
Decreasing	**64.65**	**1.33**	**8 / 12**
Fixed	65.56	2.00	5 / 12
Front loaded	65.69	2.42	2 / 12
Increasing	73.10	4.08	0 / 12
Cosine	73.84	4.58	0 / 12



(a) No Reuse (b) Reuse (c) 3SR

Figure 19: **Generative perplexity as a function of budget for different denoising strategies.** Entropy values are shown in parentheses. We observe that decreasing schedules perform best overall.

F.11 Latency and generation quality for language modeling

We measure generation-only sampling latency, defined as the wall-clock time from fully masked token IDs to final generated token IDs. The timed region includes all denoising-loop computation: model forward passes, fixed-point iterations, reuse/CoFRe (3SR) initialization, mask-confidence updates, categorical sampling, and loop logic. We synchronize CUDA before and after sampling, and exclude decoding, external Gen. PPL evaluation, file I/O, model loading, and warmup/compilation effects. All methods are evaluated under matched transformer-block budgets with the same ancestral-cache sampler, batch size, sequence length, precision, tokenizer, device, and number of samples. We use two warmup batches followed by 20 timed batches.

Table 28 and Figure 20 show that CoFRe is modestly slower than MDLM+SDTT at equal transformer-block budget, with slowdowns between $1.12\times$ and $1.45\times$, but achieves substantially better generation quality. For example, at budget 96, CoFRe reduces Gen. PPL from 193.05 to 101.791 with latency increasing from 1.71s to 2.47s. At budget 384, it improves Gen. PPL from 62.29 to 48.755 with only a $1.15\times$ slowdown. Overall, CoFRe is not faster than MDLM+SDTT at equal budget in the current implementation. Its latency advantage is therefore budget-dependent: for example, CoFRe at budget 192 (Gen. PPL 65.2, 4.40s) is close in Gen. PPL to MDLM+SDTT at budget 384 (62.3, 6.39s), although at lower entropy.

Table 28: Generation-only sampling latency and quality on OWT. Latency is measured from fully masked token IDs to final generated token IDs, excluding tokenizer decoding and external Gen. PPL evaluation.

Budget	**Latency (s)** ↓		**Slowdown**	**Gen. PPL** ↓	
	MDLM+SDTT	**CoFRe**	**CoFRe / MDLM+SDTT**	**MDLM+SDTT**	**CoFRe**
96	1.71	2.47	1.45×	193.05	101.791
192	3.29	4.40	1.34×	89.17	65.182
384	6.39	7.34	1.15×	62.29	48.755
768	12.51	14.01	1.12×	47.04	37.846

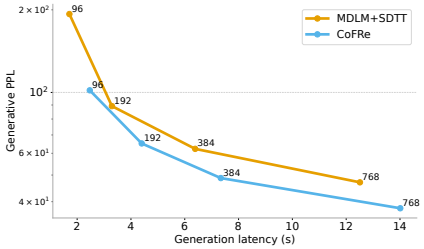


Figure 20: **Generation quality as a function of wall-clock sampling latency on OWT.** We report generation-only latency, measured from fully masked token IDs to final generated token IDs, excluding decoding and external Gen. PPL evaluation. Points are annotated by their transformer-block budget. CoFRe is modestly slower than MDLM+SDTT at matched budget, but reaches substantially lower Gen. PPL at lower wall-clock latency in the low- and medium-budget regimes.

F.12 Fixed-point residual analysis.

We measure the relative hidden residual

$$r_t^{(n)} = \frac{\|F_\theta(h_t^{(n)}; \tilde{h}_t, t) - h_t^{(n)}\|_2}{\|h_t^{(n)}\|_2}$$

across FP iterations and denoising steps. As shown in Figure 21, residuals decrease steadily with iterations, confirming that the repeated block behaves as an iterative fixed-point solver. Without reuse, the mean residual drops from 75.97 to 0.0180 after four iterations; with full reuse, it starts much closer and drops from 0.317 to 0.00545. Excluding the first step, full reuse reduces the median initial residual from 76.68 to 0.0385, i.e. it starts about $1984\times$ closer to the current fixed point. These results validate the solver and the benefit of warm starts. However, lower residual does not necessarily imply better generation: full reuse is numerically closest to equilibrium, but can over-reuse stale states, motivating the token-aware 3SR rule.

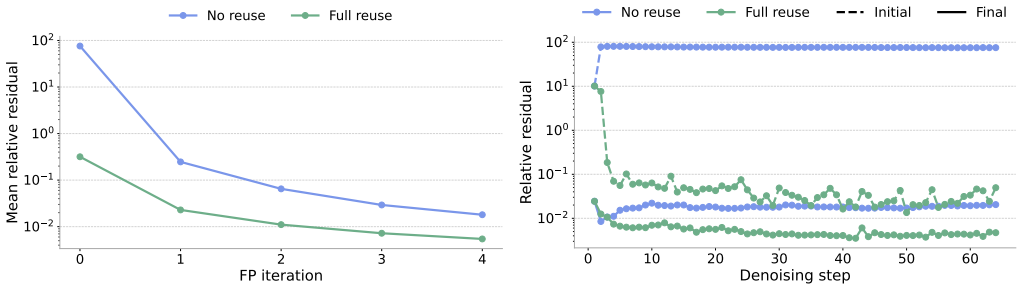


Figure 21: **Fixed-point residual analysis.** (Left) Mean relative residual decreases with FP iterations, showing that the repeated block approaches a fixed point. Full reuse starts much closer to the solution than no reuse. (Right) Across denoising steps, reuse strongly reduces the initial residual and yields lower final residuals under the same iteration budget. Residuals validate the solver and warm-start mechanism, while generation ablations are needed to compare full reuse and 3SR quality.

F.13 Effect of Training Duration on Generation Quality

In this section, we compare the effect of the training duration on generation quality. We compare here MDLM trained at 100k steps, MDLM trained at 1M steps (checkpoint from [58]), FP-MDLM trained at 100k steps, FP-MDLM trained at 1M steps and FP-MDLM trained at 100k + $\mathcal{L}_{\text{CONS}}$ +3SR (so basically CoFRe at 100k training steps). For all the models, we evaluate both the generative perplexity and the unigram entropy. We report the results in Table 29.

Table 29: **Training-stage comparison for MDLM and FP-MDLM generation on OpenWebText.** We report generative perplexity and entropy across sampling budgets. The table compares MDLM and FP-MDLM checkpoints at different training stages, together with the FP-MDLM checkpoint trained with cross-step consistency regularization and evaluated with three-state reuse.

		Budget			
Method	**Metric**	**96**	**192**	**384**	**768**
MDLM @100k	Gen PPL ↓	642.425	277.504	177.962	134.952
	Entropy ↑	5.866	5.779	5.730	5.693
MDLM	Gen PPL ↓	830.820	343.330	196.790	143.880
	Entropy ↑	5.910	5.810	5.750	5.700
FP-MDLM @100k	Gen PPL ↓	379.484	268.113	213.801	180.315
	Entropy ↑	5.790	5.755	5.722	5.694
FP-MDLM	Gen PPL ↓	375.631	273.275	215.197	179.655
	Entropy ↑	5.810	5.763	5.726	5.702
FP-MDLM @100k + $\mathcal{L}_{\text{CONS}}$ + 3SR	Gen PPL ↓	96.802	61.891	49.669	37.717
	Entropy ↑	5.519	5.411	5.409	5.246

We observe that the generative quality after 100k steps closely matches that at 1M steps, suggesting that 100k steps are likely sufficient to test new algorithms or framework when pretraining a model. If a model fails by that time, it is unlikely to succeed at 1M steps. Further analysis is needed to determine whether this reflects limitations of the metrics (generative perplexity and entropy may not capture differences at that scale) or constraints imposed by model capacity. Pynadath et al. [55] observes similar results at 50k steps when analyzing the generative frontiers of these models.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: claims made in the abstract and introduction (Section 1) are supported in Sections 2 and 3 and through experimentation in Section 4. Additionally, the supplementary material (appendices) further support our contributions.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: limitations to our work are discussed and analyzed in Appendix A.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Appendix C.3 states the assumptions (local contraction of the fixed-point map) and gives a complete proof of the solver error bound; it also states explicitly which connection to the training loss is not proven. Theoretical results from previous works are properly credited and referenced.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: the proposed method is introduced in Section 2, with further details and algorithms provided in Appendix D and E. Details on the datasets (including splits, dataset statistics and details on sampling) as well as the evaluation metrics are available in Appendix D. In addition, our code is publicly available at `https://github.com/andreamiele/fp-mgm/`.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: code is available at https://github.com/andreamiele/fp-mgm/; all datasets used (OpenWebText, ImageNette) are publicly available.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: experimental details on the datasets implemented and generated, as well as the splits used are available in Appendix D. Details on the hyperparameters and other model details are provided in Appendix D.1, D.1.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: for the OWT component controls that support the attribution of gains to each component, we report mean $\pm$ standard deviation over three seeds (Appendix F.2). Other results are single runs, because training a full model (for instance FP-MDLM) multiple times is costly; we use the same evaluation pipeline for all methods to ensure comparability.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: experimental details and information on the computational resources used are detailed in Appendix D.5.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: our work respects the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: we discuss potential positive and negative societal impacts of our work in Appendix B.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: the release of our model does not pose any direct safety risk.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: we acknowledge all sources of code and data used in our work, with citations provided both in the paper and the accompanying codebase.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.

- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: the released code is documented in the public repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: experiments in this paper do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

Justification: experiments in this paper do not involve crowdsourcing nor research with human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: our method does not involve the usage of LLMs as part of the core research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.