Not All Denoising Steps Are Equal: Model Scheduling for Faster Masked Diffusion Language Models

Anonymous ACL submission

Abstract

Recent advances in masked diffusion language models (MDLMs) narrow the quality gap to autoregressive LMs, but their sampling remains expensive because generation requires many full-sequence denoising passes with a large Transformer and, unlike autoregressive decoding, cannot benefit from KV caching. In this work, we exploit the flexibility of the diffusion framework and study model scheduling, where a smaller MDLM replaces the full model at a subset of denoising steps. Across models trained on OpenWebText and LM1B, we show that early and late denoising steps are substantially more robust to such replacement than middle steps, enabling up to a 17% reduction in FLOPs with only modest degradation in generative perplexity under both unconditional and prefix-conditional generation, while preserving sample diversity. We support these findings with a step-importance analysis based on loss and KL divergence between small and large models across timesteps, as well as an exhaustive search over coarse step segments, both of which identify the middle of the diffusion trajectory as most sensitive consistently across datasets. Our results suggest that simple, architecture-agnostic scheduling rules can significantly accelerate MDLM sampling while largely preserving generation quality.

1 Introduction

Masked diffusion language models (MDLMs) (Sahoo et al., 2024) have recently emerged as a competitive alternative to autoregressive language models, narrowing the quality gap (Gong et al., 2025; Nie et al., 2025b; Ye et al., 2025) while offering a different generation paradigm based on iterative denoising. However, MDLM sampling remains expensive: generation requires many full-sequence denoising passes with a large Transformer, and unlike autoregressive decoding, this process cannot benefit from KV caching (Wu et al., 2025a,b). As a result, even when MDLM quality is strong, inference cost can be a practical bottleneck.

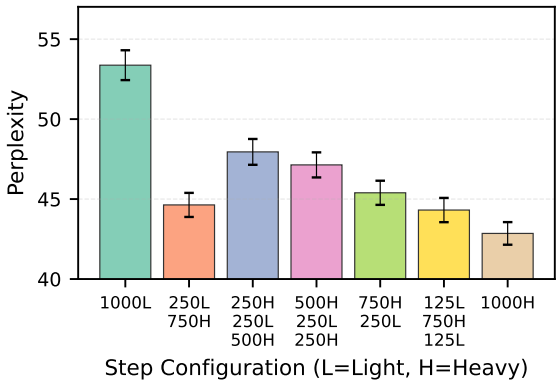

Figure 1: Generative perplexity for model schedules using a heavy 12-block model and a light 4-block model with exactly 250/1000 light steps (16.7% saved FLOPs). Each bar label encodes a schedule as contiguous segments, e.g., $(\mathrm{L125}, \mathrm{H750}, \mathrm{L125})$ denotes the *sandwich* schedule (125 light, 750 heavy, 125 light steps), while placing all light steps in the 2nd or 3rd quarter yields the worst perplexity.

A distinctive feature of diffusion models is that generation proceeds through a sequence of steps that gradually transform a heavily corrupted state into a clean sample. This structure suggests a natural question: are all denoising steps equally "difficult," and therefore equally deserving of full model capacity? In continuous image diffusion, a growing body of work (Shen et al., 2024; Huang et al., 2025) explores timestep-dependent compute allocation, including approaches that skip, cache, or dynamically adjust capacity across the trajectory. These methods are motivated by evidence that model behavior varies systematically across timesteps, often exhibiting relatively smooth or monotonic trends in step difficulty. For example, DyDiT++ (Zhao et al., 2025) analyzes loss differences between small and large diffusion Transformers and reports that these differences can shrink toward one end of the tra-

jectory, suggesting that some timesteps may be handled effectively by smaller models.

Whether similar conclusions hold for discrete masked diffusion in language remains unclear. Text denoising differs from image denoising in both the state space (discrete tokens with masking) and the structure of the prediction problem (categorical distributions over vocabularies, with uncertainty concentrated on masked positions). Consequently, step-importance patterns and effective acceleration strategies may not transfer directly from continuous image diffusion to masked diffusion for text.

In this work, we study model scheduling for faster MDLM sampling: at inference time, we replace a subset of denoising steps of a large "heavy" MDLM with a separately trained smaller "light" MDLM. This approach is intentionally simple and architecture-agnostic: it does not require retraining the heavy model, distillation, or modifying the sampling algorithm beyond choosing which model to run at each step. The central question is then straightforward: **which timesteps are most robust to model replacement, and how should light and heavy steps be arranged to best trade off speed and quality?**

Our empirical results on OpenWebText and LM1B show that denoising steps are not equally important for masked diffusion generation. When we fix a compute budget (e.g., replacing 25% of steps with a light model), the placement of these light steps matters substantially: replacing steps in the middle of the trajectory yields the largest degradation in generative perplexity, while allocating light steps near the beginning and end performs best. In particular, a simple sandwich schedule that places light steps at both ends of the trajectory consistently outperforms schedules that concentrate light steps in the middle. These observations enable meaningful inference savings, achieving up to a 17% reduction in FLOPs with only modest degradation in generative perplexity.

To validate that this pattern is not an artifact of a small set of hand-designed schedules, we perform an exhaustive search over coarse step segments and find the same qualitative conclusion: middle segments are the most sensitive to replacement, while the earliest and latest segments are relatively safe. This yields a practical rule of thumb: under a fixed budget of cheap steps, it is preferable to distribute them across both ends of the trajectory rather than concentrating them in the middle.

We further support these findings with a step-importance analysis based on **model similarity vs. timestep**. We compare light and heavy models on the same corrupted inputs at each timestep and measure their disagreement via differences in masked-token cross-entropy and token-level KL divergence. Both measures exhibit a clear peak in the middle of the trajectory, indicating maximal divergence between small and large models at intermediate noise levels. This provides a mechanistic explanation for why middle-step replacement is most harmful and clarifies how step sensitivity in masked diffusion for text differs from the smoother, often monotonic step-importance trends reported in prior continuous image diffusion analyses.

In summary, our contributions are:

1. Model scheduling for MDLMs: We study an inference-time acceleration strategy that mixes a heavy MDLM with a separately trained light MDLM across denoising steps, without distillation or architecture modification.

2. Empirical step-importance finding: Across two datasets (OpenWebText and LM1B) and both unconditional and prefix-conditional generation, we show that early and late denoising steps are substantially more robust to model replacement than middle steps. This yields a continuous quality–efficiency tradeoff controlled by light-model size and step fraction (e.g., from 3.4% perplexity degradation at 16.7% FLOPs savings to larger reductions at higher substitution rates), while preserving sample diversity.

3. Explanatory analysis: We provide complementary evidence from (i) loss/KL-based similarity across timesteps and (ii) exhaustive search over coarse segments, both identifying the middle of the trajectory as most compute-sensitive. This peaked pattern contrasts with the monotonic trends reported in continuous image diffusion, revealing a qualitatively different step-importance structure in discrete masked diffusion for text.

Section 2 reviews masked diffusion LMs and related efficiency work; Section 3 introduces model scheduling; Section 4 presents empirical results and step-importance analyses; Section 5 discusses limitations and future directions.

2 Related Work

2.1 Combining Diffusion Models

A classical way to combine models at sampling time is guidance, including classifier guidance (Dhariwal and Nichol, 2021) and classifier-free guidance (CFG) (Ho and Salimans, 2022). Closer to our setting, several vision works explicitly mix models of different sizes across the denoising trajectory to trade off speed and quality without retraining: OMS-DPM (Liu et al., 2023) searches for an optimal per-timestep model assignment under a time budget, while T-Stitch (Pan et al., 2023) "stitches" a small model into the early part of the trajectory as a drop-in replacement.

2.2 Diffusion Models Acceleration

Diffusion acceleration methods fall into two broad categories: reducing the number of function evaluations (e.g., DDIM (Song et al., 2021), DPM-Solver (Lu et al., 2022)) and reducing the cost of each evaluation (distillation/consistency and architecture-level adaptivity). Distillation-based methods such as progressive distillation (Salimans and Ho, 2022) and consistency models (Song et al., 2023) aim to preserve quality with fewer steps. A complementary direction makes the *denoiser itself* compute-adaptive. Step-aware or schedule-aware diffusion backbones include DDSM (Yang et al., 2024), which studies step importance and step-dependent capacity, as well as diffusion-transformer specific techniques that skip/caches computation or dynamically route capacity, such as Learning-to-Cache (Ma et al., 2024), Dynamic Diffusion Transformers (DyDiT) (Zhao et al., 2025), MD-DiT (Shen et al., 2024), AdaDiff (Tang et al., 2024). Related NAS-style methods (e.g., Flexiffusion (Huang et al., 2025)) also optimize which segments are full/cached/skipped to meet a compute target. These works are primarily developed and evaluated in continuous image diffusion, and their conclusions about which timesteps deserve more capacity do not necessarily transfer to discrete masked diffusion for text.

2.3 Masked Diffusion Language Models

Diffusion for text has been explored in both continuous and discrete spaces. Continuous-text diffusion includes Diffusion-LM (Li et al., 2022), latent variants (Lovelace et al., 2023), and conditional variants such as DiffuSeq (Gong et al., 2023) and simplex-based (Meshchaninov et al., 2025; Shabalin et al., 2025). Discrete diffusion models for language build on discrete-state diffusion formulations such as D3PM (Austin et al., 2023), and include DiffusionBERT (He et al., 2022) and Score Entropy Discrete Diffusion (SEDD) (Lou et al., 2024). Recent masked diffusion language models (MDLMs) (Sahoo et al., 2024; Shi et al., 2025) show that a simple masked diffusion objective with strong training recipes can close much of the quality gap to autoregressive LMs, and ReMDM (Wang et al., 2025) improves sampling via inference-time remasking and compute scaling. Compared to AR LMs, diffusion LMs can also be stronger learners under data-constrained settings (Ni et al., 2025; Rütte et al., 2025).

Several efforts scale diffusion LMs and explore hybridization with autoregressive decoding, including large-scale MDLM reports such as LLaDA (Nie et al., 2025b) and Dream (Ye et al., 2025), as well as domain/architecture variants such as DiffuCoder (Gong et al., 2025) and DiffuLLaMA (Nie et al., 2025a). A separate line of work targets inference efficiency. Some methods recover or approximate KV-cache benefits for bidirectional diffusion via block/hybrid formulations and cache reuse (Arriola et al., 2025a; Wu et al., 2025a; Sahoo et al., 2025; Arriola et al., 2025b; Wu et al., 2025b). Others reduce the effective number of denoising iterations through adaptive or distilled decoding policies, including FlashDLM (FreeCache and AR-guided step reduction) (Hu et al., 2025), Local-Leap (Kong et al., 2025), and CD4LM (Liang et al., 2026). Finally, dInfer provides a system-oriented framework with modular decoding strategies and KV-cache management for efficient diffusion-LM serving (Ma et al., 2025). These approaches are orthogonal to model scheduling: they keep the denoiser fixed and either reduce the number of denoising iterations or amortize per-step compute via state reuse, whereas we vary *which* denoiser runs at each step. The two axes compose: pairing our schedules with dKV-Cache yields a $1.24\times$ wall-clock speedup over the uncached heavy baseline at unchanged perplexity.

Finally, token difficulty is known to be non-uniform in autoregressive generation, with evidence that per-position perplexity can vary systematically across a sequence (Helm et al., 2025; Zur et al., 2025; Yang and Holtzman, 2025; Bell et al., 2025); this motivates exploring whether "where to spend compute" is similarly non-uniform across diffusion timesteps in masked diffusion generation.

3 Accelerating MDLM via Model Scheduling

3.1 Experimental setup

Masked diffusion language models. Let $x = (x^{(1)}, \ldots, x^{(L)})$ denote a clean token sequence of length L. Autoregressive language models generate x sequentially by modeling $p_\theta(x^{(i)} \mid x^{(<i)})$ and are typically trained with a token-level cross-entropy objective. In contrast, masked diffusion language models (MDLMs) generate text by repeatedly denoising a partially masked sequence. Concretely, we define a discrete forward noising process q that corrupts x into z_t by replacing tokens with a special mask token m according to a time-dependent corruption level.

Forward process $q(z_t \mid x)$. We represent each token as a one-hot vector in $\{0,1\}^{|V|}$. For a normalized time $t \in [0,1]$, the forward process produces a noisy sequence $z_t = (z_t^{(1)}, \ldots, z_t^{(L)})$ with independent per-position marginals

$$q(z_t^{(\ell)} \mid x^{(\ell)}) = \mathrm{Cat}\Big(\alpha_t\, x^{(\ell)} + (1-\alpha_t)\,\pi\Big)\,, \quad (1)$$

where π is a fixed prior distribution over vocabulary tokens. In our setting we use *pure masking*, i.e., $\pi = \delta_m$, so that with probability $(1-\alpha_t)$ a token is replaced by m, and otherwise it is kept unchanged. We use a *linear* schedule $\alpha_t = 1 - t$, so that the expected masked fraction equals t. During training we sample $t \sim \mathcal{U}(0,1)$.

Denoiser and training objective. The denoiser $p_\theta(x \mid z_t, t)$ is parameterized by a bidirectional Transformer that predicts the original token at each position given the noisy sequence and timestep. Training minimizes a weighted masked language modeling loss over masked positions $M(z_t) = \{\ell : z_t^{(\ell)} = m\}$ only. Following prior derivations of the (negative) ELBO for this discrete diffusion process, the objective can be written as

$$\mathcal{L}_{\mathrm{MDLM}}(x) = \mathbb{E}_{t\sim\mathcal{U}(0,1),\, z_t\sim q(\cdot|x)}\left[\ell_\theta(x; z_t, t)\right],$$
$$\ell_\theta(x; z_t, t) = \frac{\alpha_t'}{1-\alpha_t} \sum_{\ell\in M(z_t)} -\log p_\theta\Big(x^{(\ell)} \mid z_t, t\Big)\,. \quad (2)$$

Here α_t' denotes the derivative of α_t with respect to t. The factor $\frac{\alpha_t'}{1-\alpha_t}$ reweights timesteps so that different corruption levels contribute appropriately to the variational objective; for $\alpha_t = 1-t$, we have $\frac{-\alpha_t'}{1-\alpha_t} = \frac{1}{t}$.

Sampling (reverse process). To generate an unconditional sequence of length L, sampling starts from a fully masked sequence $z_{t=1}$ where $z_{t=1}^{(\ell)} = m$ for all ℓ. The reverse process proceeds for T discrete steps with times $t_i = i/T$ for $i = T, T-1, \ldots, 0$. We use the standard MDLM sampler (no remasking): once a token is generated (unmasked), it remains fixed thereafter. Multiple tokens can be updated in parallel at each step (unlike autoregressive decoding); however, each denoising step requires a full bidirectional Transformer forward pass over the entire sequence, and thus inference cost scales with the number of denoising evaluations.

Why sampling is expensive. Although MDLM sampling updates many tokens at once, it typically requires a large number of sequential denoising steps and does not admit the KV-caching efficiency of autoregressive decoding. This motivates our focus on *model scheduling*: replacing the full denoiser with a smaller denoiser on a subset of timesteps to reduce total compute while retaining generation quality.

Model scheduling. Let $\{\theta_k\}_{k\in\mathcal{K}}$ denote a set of denoisers of different sizes (e.g., $k \in \{4, 6, 8, 10, 12\}$ Transformer blocks) trained with the same objective. A *model schedule* is a function $s : \{1, \ldots, T\} \to \mathcal{K}$ that selects which denoiser to use at each reverse step i. Sampling then applies $p_{\theta_{s(i)}}(\cdot \mid z_{t_i}, t_i)$ at step i. If the heavy model has B_H blocks and the light model has B_L blocks, then replacing a fraction p of steps by the light model yields a relative compute reduction of

$$\text{saved FLOPs} \approx p \cdot \frac{B_H - B_L}{B_H}. \quad (3)$$

Models and training details. To isolate timestep scheduling, we train depth-scaled MDLM denoisers following the setup of Sahoo et al. (2024). Our main experiments use OpenWebText (Gokaslan et al., 2019), a broad general-domain corpus also used in prior MDLM work, which lets us study unconditional generation without task-specific structure. All models share the same tokenizer, width, objective, and noise schedule, and differ only in Transformer depth (4/6/8/10/12 blocks). The 12-block model is the *heavy* baseline, while smaller models are candidate *light* denoisers. Full architecture, optimization, and dataset details are given in Appendix A.

Evaluation metric. Standard zero-shot LLM benchmarks (e.g., MMLU, HellaSwag, ARC) saturate near random for models of our scale($<$ 120M parameters) and provide little signal, so we follow MDLM (Sahoo et al., 2024; Shi et al., 2025) in using *generative perplexity* computed by a pretrained GPT-2 (Brown et al., 2020) on generated samples as the quality proxy. Unless stated otherwise, we generate 1600 independent samples of length 1024 using $T = 1000$ denoising steps and report mean perplexity. Generative perplexity can be unreliable in some settings (e.g., ReMDM (Wang et al., 2025) discusses failure modes), but under identical training and sampling protocols it gives a consistent *relative* comparison across schedules. We additionally report token-level entropy as a sample diversity measure and evaluate under prefix-conditional generation (Section 3.5).

Second dataset. To test whether our findings generalize beyond a single corpus, we train an identical model family on the One Billion Word Benchmark (LM1B) (Chelba et al., 2014) with 128-token sequence length. All other architecture and training choices remain the same.

3.2 Fixed light-step ratio (25%)

We first consider a simple setting with two models: a 12-block heavy MDLM and a 4-block light MDLM. We replace exactly 25% of the heavy model's denoising steps with the light model and ask: *which steps should be replaced to minimize quality loss?* Under our compute accounting in Eq. 3, the saved FLOPs are 16.7%. Although our framework naturally extends to using more than two models, we focus on this clear two-model setup to make the effect of schedule placement easy to interpret.

We test several hand-crafted schedules that place the 250 light steps in different parts of the trajectory (by quarters), and also a *sandwich* schedule that splits the 250 light steps into two equal segments of 125 and places them at the beginning and end of the trajectory. The results are shown in Figure 1. Replacing steps in the middle of the trajectory (2nd/3rd quarters) yields the worst perplexity, while the sandwich schedule performs best, closely followed by placing all light steps in the first quarter. These results indicate that denoising steps are not equally important for masked diffusion generation. Token-level entropy for these schedules (Table 6 in Appendix E) confirms stable sample diversity across all configurations. Additional hand-crafted schedules for other light model sizes are reported in Appendix B and exhibit the same qualitative pattern. The same pattern holds on our LM1B models (Appendix D, Figure 11), confirming cross-dataset generality.

3.3 Exhaustive search over coarse step segments

To further validate the above trend under a stronger compute reduction, we replace 400 out of 1000 denoising steps (40%) with the 4-block light model. This corresponds to 26.7% saved FLOPs. While some prior works search over timesteps via learned predictors (Liu et al., 2023) or heuristic optimization schemes (Yang et al., 2024), we perform an exhaustive search for transparency.

A naive search over all subsets of 400 steps is infeasible: $\binom{1000}{400} \approx 5 \times 10^{290}$. We therefore partition the 1000 steps into 10 contiguous segments of 100 and select 4 to run with the light model, yielding $\binom{10}{4} = 210$ schedules. Contiguity is also a practical convenience: when a step commits no new positions, the MDLM sampler can reuse the previous logits instead of re-running the network, and staying on one model across a segment allows this bookkeeping. Each schedule is evaluated on 160 unconditional samples with fixed seeds.

Across the 210 schedules, the best configurations consistently place light segments near the beginning and end of the trajectory, while the worst place them predominantly in the middle. We quantify this by counting segment frequency among the top-20 schedules (Figure 2). Middle segments appear disproportionately often in the worst schedules, confirming that mid-trajectory steps are the most sensitive to replacement. Examples of the top-5 best and bottom-5 worst schedules, together with the segment frequency among the 20 worst schedules, are provided in Appendix C.

As a practical rule of thumb suggested by both the hand-crafted and exhaustive results, spreading cheaper steps across *both* ends of the trajectory tends to be preferable to concentrating them in the middle. For example, for 600 light steps one can use a symmetric schedule of $(\mathrm{L}300, \mathrm{H}400, \mathrm{L}300)$.

3.4 Scaling over light model size / light-step fraction

We next study two scaling dimensions: (i) the size of the light model and (ii) the fraction of light steps. Table 1 fixes the *sandwich* placement (125

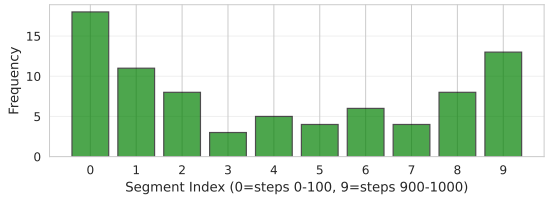

Figure 2: Segment frequency in the top 20 best-performing configurations. Bars show how often each segment is assigned to the light model across the top-20 schedules. Higher frequency suggests that replacing this segment is relatively safe.

light + 750 heavy + 125 light) and varies the light model depth from 4 to 10 blocks, paired with the 12-block heavy baseline. As expected, increasing light-model depth reduces the quality drop while also reducing the achievable FLOPs savings.

Table 2 fixes the model pair (12-block heavy, 4-block light) and varies the percentage of steps executed by the light model from 0% to 100%. In addition to FLOPs-based estimates, we report end-to-end wall-clock time for the same sampling setup. We observe a smooth transition in perplexity as the schedule shifts from fully heavy to fully light, indicating that model scheduling provides a continuous speed–quality tradeoff.

Light Model	Gen. PPL	PPL drop	Saved FLOPs
4b	44.31 $\pm$ 0.76	3.41%	16.67%
6b	43.67 $\pm$ 0.67	1.94%	12.50%
8b	43.45 $\pm$ 0.73	1.40%	8.33%
10b	42.90 $\pm$ 0.70	0.12%	4.17%
12b	42.85 $\pm$ 0.71	0.00%	0.00%

Table 1: Scaling the *light model* number of **b**locks while keeping schedule placement fixed to the sandwich pattern (125 light + 750 heavy + 125 light, i.e., 25% light steps). "PPL drop" is relative to the all-heavy 12-block baseline. Numbers in scriptsize represent 95% confidence intervals.

Wall-clock vs. FLOPs. Measured wall-clock gains (Table 2) trail the FLOPs estimate because the depth-independent output projection dominates runtime at our scale ($\approx$82% at 4 blocks, $\approx$60% at 12). This is a small-model artifact: at production scale block compute dominates and FLOPs predictions tighten. If the end-to-end runtime decomposes as $T \approx T_{\text{out}} + T_{\text{blocks}}$, and only T_{blocks} scales with depth, then the attainable speedup from reducing depth is limited by the fraction of time spent outside the blocks. Writing $\alpha := T_{\text{out}}/T$, the maximum possible speedup satisfies $\text{speedup} \leq \frac{1}{1-\alpha}$.

% light steps	Gen. PPL	Saved FLOPs	Time (s)	Speedup
0	42.9	0.0%	109.7	0.0%
10	43.1	6.7%	106.4	3.0%
20	43.8	13.3%	103.3	5.8%
30	44.7	20.0%	100.4	8.5%
40	45.9	26.7%	97.3	11.3%
50	47.2	33.3%	94.3	14.0%
60	48.6	40.0%	91.1	17.0%
70	50.1	46.7%	90.7	17.3%
80	51.4	53.3%	85.4	22.2%
90	52.5	60.0%	84.8	22.7%
100	53.4	66.7%	78.7	28.3%

Table 2: Scaling the *fraction of light steps* for the (12-block heavy, 4-block light) model pair under a 'sandwich' pattern.

Schedule	Cond. Gen. PPL (CI)	Entropy
L1000	47.5 $\pm$ 0.44	5.42
L250 $\rightarrow$ H750	40.9 $\pm$ 0.38	5.42
H250 $\rightarrow$ L250 $\rightarrow$ H500	43.2 $\pm$ 0.41	5.41
H500 $\rightarrow$ L250 $\rightarrow$ H250	42.6 $\pm$ 0.41	5.41
H750 $\rightarrow$ L250	41.4 $\pm$ 0.40	5.40
L125 $\rightarrow$ H750 $\rightarrow$ L125	40.9 $\pm$ 0.36	5.42
H1000	39.3 $\pm$ 0.37	5.40

Table 3: Prefix-conditional generation (256-token prefix from OpenWebText). Schedule ranking matches the unconditional setting (Figure 1). **H**eavy: 12-block, **L**ight: 4-block. Entropy remains stable across schedules, indicating no diversity loss from scheduling.

3.5 Prefix-conditional generation and diversity

We also test whether the same scheduling pattern holds under prefix-conditional generation, which is more relevant in practice. We repeat the schedule comparison using prefix-conditional sampling: generation starts from 256 prefix tokens drawn from held-out OpenWebText data, and the remaining positions are denoised. Table 3 reports conditional generative perplexity and token-level entropy. The schedule ranking is unchanged: middle-step replacement gives the worst perplexity, while the sandwich schedule remains close to the all-heavy baseline (40.9 vs. 39.3). Entropy stays stable across all schedules (5.40–5.42), suggesting that scheduling does not reduce sample diversity. The same stability appears in unconditional generation (Appendix E). Results with 128-token prefixes show the same pattern (Appendix F).

Method	Gen. PPL ↓	s/sample ↓	Speedup ↑
H_{1000}	42.9	6.76	1.00×
+ DKV-CACHE	43.2	5.93	1.14×
$L_{250} \to H_{750}$	44.6	6.29	1.08×
+ DKV-CACHE	**44.6**	**5.47**	**1.24×**

Table 4: KV caching composes with model scheduling. Caching accelerates both the all-heavy baseline and the mixed schedule; the combined setting gives the best wall-clock speedup while preserving generative perplexity.

3.6 Composability with KV caching

Model scheduling changes which denoiser is used at each timestep, while KV caching reduces the cost of a denoiser call. To test whether these axes compose in practice, we combine our schedules with DKV-CACHE (Ma et al., 2026): cache-mode steps compute queries only at active positions and reuse cached keys and values for committed positions.

Table 4 shows that the gains stack: caching alone gives 1.14× speedup, scheduling alone gives 1.08×, and their combination reaches 1.24× with no additional PPL degradation. Across the full schedule sweep, adding DKV-CACHE changes PPL by only 0.30 on average and is faster than the corresponding uncached schedule in every case (Appendix G).

4 Why does this work? Step importance analysis

4.1 Model similarity vs timestep

Inspired by OMS-DPM (Liu et al., 2023), we compare model behavior across noise levels. Following the similarity analyses used in dynamic diffusion transformer works (e.g., (Zhao et al., 2025)), we compute loss differences and KL divergences between models at fixed timesteps. For each timestep we evaluate on 500 sequences of length 1024, and crucially compare models on the *same* corrupted inputs.

Loss difference. For a fixed timestep t, we sample $z_t \sim q(\cdot \mid x)$ and compute the (unweighted) masked-token cross-entropy

$$\mathcal{L}_\theta(z_t, t) = \frac{1}{|M(z_t)|} \sum_{\ell \in M(z_t)} -\log p_\theta(x^{(\ell)} \mid z_t, t). \tag{4}$$

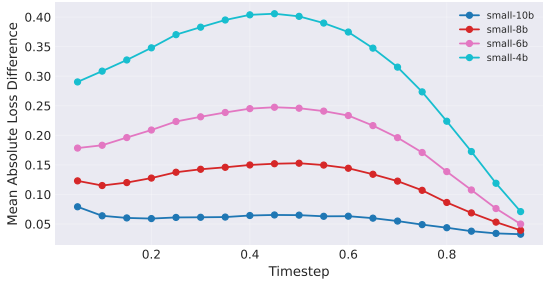

Figure 3: Mean absolute difference in masked-token cross-entropy between each light model and the heavy 12-block baseline across timesteps. Each curve compares one light model to the baseline, evaluated on the same corrupted inputs z_t. Lower values indicate higher similarity.

We then measure the mean absolute loss difference between a light model θ_L and the heavy model θ_H:

$$\Delta_{\text{loss}}(t) = \mathbb{E}_x \, \mathbb{E}_{z_t \sim q(\cdot|x)} \left| \mathcal{L}_{\theta_L}(z_t, t) - \mathcal{L}_{\theta_H}(z_t, t) \right|. \tag{5}$$

The result is presented in Figure 3. Prior work on continuous image diffusion often reports a monotonic trend across timesteps (Pan et al., 2024; Zhao et al., 2025). In contrast, we observe a clear peak in the middle of the trajectory, indicating maximal disagreement between light and heavy models at intermediate noise levels.

KL divergence. We additionally compare the *token distributions* predicted at masked positions. Let $p_\theta(\cdot \mid z_t, t, \ell)$ denote the categorical distribution over the vocabulary for position $\ell \in M(z_t)$. We compute the average token-level KL divergence between two models:

$$\Delta_{\text{KL}}(t) = \mathbb{E}_x \, \mathbb{E}_{z_t \sim q(\cdot|x)} \left[\frac{1}{|M(z_t)|} \times \sum_{\ell \in M(z_t)} \text{KL}\big(p_{\theta_H}(\cdot \mid z_t, t, \ell) \,\|\, p_{\theta_L}(\cdot \mid z_t, t, \ell)\big) \right]. \tag{6}$$

To account for intrinsic ambiguity in text prediction, we compute a baseline $\Delta_{\text{KL}}^{\text{base}}(t)$ as the KL divergence between two independently trained heavy (12-block) checkpoints with different random seeds (different initialization and data order), and report the *relative* divergence $\Delta_{\text{KL}}(t) - \Delta_{\text{KL}}^{\text{base}}(t)$. Figure 4 shows the same qualitative pattern as Figure 3: disagreement peaks near the middle ($t \approx 0.4$–0.6) and is substantially smaller at both ends. Here $t = 1$ corresponds to the fully masked state (start of sampling), and $t \to 0$ corresponds to the nearly unmasked state (end of sampling).

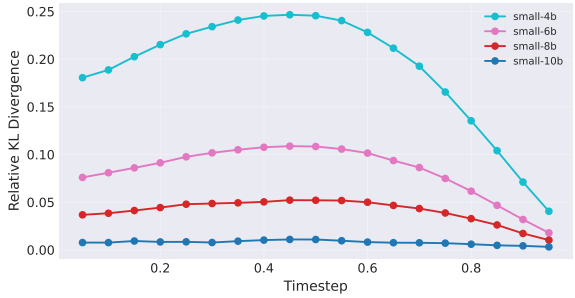

Figure 4: Relative token-level KL divergence (Eq. 6) between model pairs across timesteps, after subtracting a baseline KL curve computed between two independently trained heavy (12-block) checkpoints. Lower values indicate closer agreement. Divergence peaks in the middle of the trajectory, suggesting that intermediate timesteps are most sensitive to model replacement.

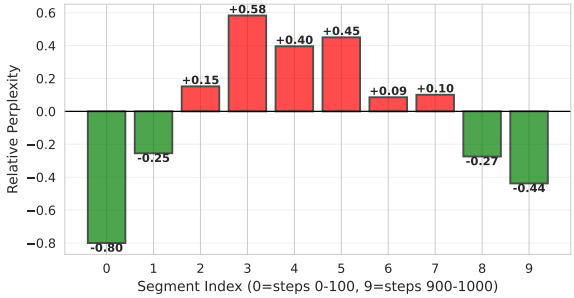

Figure 5: Mean-subtracted segment influence from the exhaustive 10-segment search (Section 3.3). For each segment j, we compute the mean perplexity over schedules that assign segment j to the light model, and subtract the mean perplexity over all schedules. Positive values indicate that replacing this segment is harmful on average; negative values indicate that replacing this segment is relatively safe.

We replicate this KL-divergence analysis on our LM1B models and observe the same characteristic middle-trajectory peak (Appendix D, Figure 12), confirming that the non-monotonic step-importance pattern is a general property of masked diffusion rather than an artifact of the OpenWebText setup.

4.2 Segment influence from exhaustive search

The exhaustive 10-segment search in Section 3.3 provides another way to estimate step importance. For each segment $j \in \{0, \ldots, 9\}$, let $\mathcal{S}_j$ be the set of schedules in which segment j uses the light model, and let $\mathrm{PPL}(s)$ be the generative perplexity of schedule s. We compute the segment score as

$$I(j) = \frac{1}{|\mathcal{S}_j|} \sum_{s \in \mathcal{S}_j} \mathrm{PPL}(s) \; - \; \frac{1}{|\mathcal{S}|} \sum_{s \in \mathcal{S}} \mathrm{PPL}(s), \tag{7}$$

i.e., the mean perplexity of schedules that use the light model in segment j, mean-subtracted by the average perplexity over all 210 schedules. Figure 5 shows that middle segments have positive scores (worse than average when replaced), while the earliest and latest segments tend to have negative scores (more robust to replacement). This empirically matches the similarity analysis above and supports the conclusion that intermediate denoising steps are the most compute-sensitive for masked diffusion generation.

5 Conclusion

Masked diffusion language models offer a compelling alternative to autoregressive generation, but their practicality is often constrained by expensive sampling that requires many full-sequence denoising passes. In this work we studied *model scheduling* for masked diffusion LMs: replacing a subset of denoising steps of a heavy model with a separately trained lighter Transformer at inference time. Our results show that timestep importance in masked diffusion for text is strongly non-uniform: intermediate timesteps are the most sensitive to model replacement, while early and late steps are comparatively robust. This enables simple schedules, such as sandwich-style allocation of light steps to the ends of the trajectory, to reduce sampling compute with only modest degradation in generative perplexity, while preserving sample diversity. The finding is consistent across two datasets (OpenWebText and LM1B), unconditional and prefix-conditional generation, and multiple analysis methods (schedule search, loss difference, KL divergence). Notably, this peaked pattern contrasts with the monotonic trends reported in continuous image diffusion, indicating a qualitatively different step-importance structure in discrete masked diffusion for text.

Model scheduling serves both as a simple inference-time acceleration recipe and as an experimental tool for exposing this step-importance structure. Because it changes per-step model capacity without modifying the sampler, it is complementary to methods that reduce the number of denoising iterations or recover KV-cache-like efficiency, and composes with them in practice: pairing our schedules with DKV-CACHE yields a $1.24\times$ wall-clock speedup over the uncached heavy baseline at unchanged perplexity.

Limitations

Our study focuses on a two-model setup (heavy and light) with a single depth-scaled MDLM family, which makes the placement effect easy to interpret and isolates it from confounding architectural variation. Scheduling naturally generalizes to more than two capacity levels and to other architecture families. Pretrained MDLM model families spanning multiple scales, comparable to what is available for autoregressive LMs such as Qwen (Yang et al., 2025) or LLaMA (Touvron et al., 2023), are not yet standard; once they emerge, the same scheduling principles can be verified at larger scale and under standard zero-shot benchmarks. Dynamic mechanisms such as early exit or routing policies conditioned on the denoising state are also a natural next step toward further improving the speed–quality tradeoff.

Ethical Considerations

This paper proposes an inference-time efficiency method for masked diffusion language models by scheduling denoising steps across models of different sizes. The primary intended impact is to reduce sampling computation, which lowers energy use, monetary cost, and associated carbon emissions from running and evaluating generative models, and can broaden accessibility for researchers and practitioners with limited compute budgets. We note that the net environmental impact is not unambiguously positive: efficiency gains can induce increased overall usage (a rebound effect) that partially offsets per-sample savings.

Cheaper text generation can also amplify existing misuse risks of generative language models, such as spam, phishing, misinformation, and other automated manipulation, by increasing the feasible volume of generated content. Our method does not introduce new model capabilities beyond those of the underlying MDLM; it changes how computation is allocated during sampling. We recommend that deployments benefiting from these efficiency improvements follow standard responsible-use practices (access controls, abuse monitoring, rate limiting, content safety filtering) and that evaluations report compute and energy metrics alongside quality.

References

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025a. Block Diffusion: Interpolating Between Autoregressive and Diffusion Language Models. *arXiv preprint*. ArXiv:2503.09573 [cs].

Marianne Arriola, Yair Schiff, Hao Phung, Aaron Gokaslan, and Volodymyr Kuleshov. 2025b. Encoder-Decoder Diffusion Language Models for Efficient Training and Inference.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. 2023. Structured Denoising Diffusion Models in Discrete State-Spaces. *arXiv preprint*. ArXiv:2107.03006 [cs].

Tyler Bell, Avinash Mudireddy, Ivan Johnson-Eversoll, Soura Dasgupta, and Raghu Mudumbai. 2025. Slaves to the Law of Large Numbers: An Asymptotic Equipartition Property for Perplexity in Generative Language Models. *arXiv preprint*. ArXiv:2405.13798 [cs].

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, and 12 others. 2020. Language models are few-shot learners. *arXiv preprint arXiv: 2005.14165*.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. *Preprint*, arXiv:1312.3005.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Prafulla Dhariwal and Alex Nichol. 2021. Diffusion Models Beat GANs on Image Synthesis. *arXiv preprint*. ArXiv:2105.05233 [cs].

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

Shansan Gong, Mukai Li, Jiangtao Feng, Zhiyong Wu, and Lingpeng Kong. 2023. DiffuSeq: Sequence to Sequence Text Generation with Diffusion Models. *arXiv preprint*. ArXiv:2210.08933 [cs].

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. 2025. DiffuCoder: Understanding and Improving Masked Diffusion Models for Code Generation. *arXiv preprint*. ArXiv:2506.20639 [cs].

Zhengfu He, Tianxiang Sun, Kuanning Wang, Xuanjing Huang, and Xipeng Qiu. 2022. DiffusionBERT: Improving Generative Masked Language Models with Diffusion Models. *arXiv preprint*. ArXiv:2211.15029 [cs] version: 2.

Falko Helm, Nico Daheim, and Iryna Gurevych. 2025. Token Weighting for Long-Range Language Modeling. *arXiv preprint*. ArXiv:2503.09202 [cs] version: 1.

Jonathan Ho and Tim Salimans. 2022. Classifier-Free Diffusion Guidance. *arXiv preprint*. ArXiv:2207.12598 [cs].

Zhanqiu Hu, Jian Meng, Yash Akhauri, Mohamed S. Abdelfattah, Jae sun Seo, Zhiru Zhang, and Udit Gupta. 2025. Flashdlm: Accelerating diffusion language model inference via efficient kv caching and guided diffusion. *arXiv preprint arXiv: 2505.21467*.

Hongtao Huang, Xiaojun Chang, and Lina Yao. 2025. Flexiffusion: Segment-wise Neural Architecture Search for Flexible Denoising Schedule. *arXiv preprint*. ArXiv:2409.17566 [cs].

Fanheng Kong, Jingyuan Zhang, Yahui Liu, Zirui Wu, Yu Tian, Victoria W., and Guorui Zhou. 2025. Accelerating diffusion llm inference via local determinism propagation. *arXiv preprint arXiv: 2510.07081*.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. 2022. Diffusion-lm improves controllable text generation. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Yihao Liang, Ze Wang, Hao Chen, Ximeng Sun, Jialian Wu, Xiaodong Yu, Jiang Liu, Emad Barsoum, Zicheng Liu, and Niraj K. Jha. 2026. Cd4lm: Consistency distillation and adaptive decoding for diffusion language models. *arXiv preprint arXiv: 2601.02236*.

Enshu Liu, Xuefei Ning, Zinan Lin, Huazhong Yang, and Yu Wang. 2023. OMS-DPM: Optimizing the Model Schedule for Diffusion Probabilistic Models. In *Proceedings of the 40th International Conference on Machine Learning*, pages 21915–21936. PMLR.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv: 1711.05101*.

Aaron Lou, Chenlin Meng, and Stefano Ermon. 2024. Discrete Diffusion Modeling by Estimating the Ratios of the Data Distribution. *arXiv preprint*. ArXiv:2310.16834 [stat].

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. 2023. Latent Diffusion for Language Generation. *arXiv preprint*. ArXiv:2212.09462 [cs].

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. 2022. DPM-Solver: A Fast ODE Solver for Diffusion Probabilistic Model Sampling in Around 10 Steps. *arXiv preprint*. ArXiv:2206.00927 [cs].

Xinyin Ma, Gongfan Fang, Michael Bi Mi, and Xinchao Wang. 2024. Learning-to-Cache: Accelerating Diffusion Transformer via Layer Caching. *arXiv preprint*. ArXiv:2406.01733 [cs].

Xinyin Ma, Runpeng Yu, Gongfan Fang, and Xinchao Wang. 2026. dKV-cache: The cache for diffusion language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Yuxin Ma, Lun Du, Lanning Wei, Kun Chen, Qian Xu, Kangyu Wang, Guofeng Feng, Guoshan Lu, Lin Liu, Xiaojing Qi, Xinyuan Zhang, Zhen Tao, Haibo Feng, Ziyun Jiang, Ying Xu, Zenan Huang, Yihong Zhuang, Haokai Xu, Jiaqi Hu, and 4 others. 2025. dinfer: An efficient inference framework for diffusion language models. *arXiv preprint arXiv: 2510.08666*.

Viacheslav Meshchaninov, Egor Chimbulatov, Alexander Shabalin, Aleksandr Abramov, and Dmitry Vetrov. 2025. Compressed and Smooth Latent Space for Text Diffusion Modeling. *arXiv preprint*. ArXiv:2506.21170 [cs].

Jinjie Ni, Qian Liu, Longxu Dou, Chao Du, Zili Wang, Hang Yan, Tianyu Pang, and Michael Qizhe Shieh. 2025. Diffusion Language Models are Super Data Learners. *arXiv preprint*. ArXiv:2511.03276 [cs].

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. 2025a. Scaling up Masked Diffusion Models on Text. *arXiv preprint*. ArXiv:2410.18514 [cs].

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. 2025b. Large Language Diffusion Models. *arXiv preprint*. ArXiv:2502.09992 [cs].

Zizheng Pan, Jing Liu, Haoyu He, Jianfei Cai, and Bohan Zhuang. 2023. Stitched ViTs are Flexible Vision Backbones. *arXiv preprint*. ArXiv:2307.00154 [cs].

Zizheng Pan, Bohan Zhuang, De-An Huang, Weili Nie, Zhiding Yu, Chaowei Xiao, Jianfei Cai, and Anima Anandkumar. 2024. T-Stitch: Accelerating Sampling in Pre-Trained Diffusion Models with Trajectory Stitching. *arXiv preprint*. ArXiv:2402.14167 [cs].

Dimitri von Rütte, Janis Fluri, Omead Pooladzandi, Bernhard Schölkopf, Thomas Hofmann, and Antonio Orvieto. 2025. Scaling Behavior of Discrete Diffusion Language Models. *arXiv preprint*. ArXiv:2512.10858 [cs].

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. 2024. Simple and Effective Masked Diffusion Language Models. *arXiv preprint*. ArXiv:2406.07524 [cs].

Subham Sekhar Sahoo, Zhihan Yang, Yash Akhauri, Johnna Liu, Deepansha Singh, Zhoujun Cheng, Zhengzhong Liu, Eric Xing, John Thickstun, and Arash Vahdat. 2025. Esoteric Language Models. *arXiv preprint*. ArXiv:2506.01928 [cs].

Tim Salimans and Jonathan Ho. 2022. Progressive Distillation for Fast Sampling of Diffusion Models. *arXiv preprint*. ArXiv:2202.00512 [cs].

Alexander Shabalin, Viacheslav Meshchaninov, and Dmitry Vetrov. 2025. Smoothie: Smoothing Diffusion on Token Embeddings for Text Generation. *arXiv preprint*. ArXiv:2505.18853 [cs] version: 1.

Mingzhu Shen, Pengtao Chen, Peng Ye, Guoxuan Xia, Tao Chen, Christos-Savvas Bouganis, and Yiren Zhao. 2024. MD-DiT: Step-aware Mixture-of-Depths for Efficient Diffusion Transformers.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. 2025. Simplified and Generalized Masked Diffusion for Discrete Data. *arXiv preprint*. ArXiv:2406.04329 [cs].

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. 2023. Consistency Models. *arXiv preprint*. ArXiv:2303.01469 [cs].

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. 2021. Score-Based Generative Modeling through Stochastic Differential Equations. *arXiv preprint*. ArXiv:2011.13456 [cs].

Shengkun Tang, Yaqing Wang, Caiwen Ding, Yi Liang, Yao Li, and Dongkuan Xu. 2024. AdaDiff: Accelerating Diffusion Models through Step-Wise Adaptive Computation. *arXiv preprint*. ArXiv:2309.17074 [cs].

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, and 49 others. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv: 2307.09288*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv: 1706.03762*.

Guanghan Wang, Yair Schiff, Subham Sekhar Sahoo, and Volodymyr Kuleshov. 2025. Remasking Discrete Diffusion Models with Inference-Time Scaling. *arXiv preprint*. ArXiv:2503.00307 [cs].

Chengyue Wu, Hao Zhang, Shuchen Xue, Shizhe Diao, Yonggan Fu, Zhijian Liu, Pavlo Molchanov, Ping Luo, Song Han, and Enze Xie. 2025a. Fast-dLLM v2: Efficient Block-Diffusion LLM. *arXiv preprint*. ArXiv:2509.26328 [cs].

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. 2025b. Fast-dLLM: Training-free Acceleration of Diffusion LLM by Enabling KV Cache and Parallel Decoding. *arXiv preprint*. ArXiv:2505.22618 [cs].

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv: 2505.09388*.

Chenghao Yang and Ari Holtzman. 2025. LLM Probability Concentration: How Alignment Shrinks the Generative Horizon. *arXiv preprint*. ArXiv:2506.17871 [cs] version: 2.

Shuai Yang, Yukang Chen, Luozhou Wang, Shu Liu, and Yingcong Chen. 2024. DENOISING DIFFUSION STEP-AWARE MODELS.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. 2025. Dream 7B: Diffusion Large Language Models. *arXiv preprint*. ArXiv:2508.15487 [cs].

Wangbo Zhao, Yizeng Han, Jiasheng Tang, Kai Wang, Hao Luo, Yibing Song, Gao Huang, Fan Wang, and Yang You. 2025. DyDiT++: Dynamic Diffusion Transformers for Efficient Visual Generation. *arXiv preprint*. ArXiv:2504.06803 [cs].

Amir Zur, Atticus Geiger, Ekdeep Singh Lubana, and Eric Bigelow. 2025. Are language models aware of the road not taken? Token-level uncertainty and hidden state dynamics. *arXiv preprint*. ArXiv:2511.04527 [cs] version: 1.

A Training Details

For the OpenWebText experiments, we closely follow the MDLM training setup (Sahoo et al., 2024) and use their codebase and default design choices whenever possible. We train a family of Transformer-encoder denoisers (Vaswani et al., 2017; Devlin et al., 2019) with depths 4/6/8/10/12 blocks, keeping hidden size 768, MLP ratio 4, vocabulary, tokenizer, objective, and noise schedule fixed across models. The 12-block model serves as the heavy baseline, and the smaller models serve as light denoisers.

Models are trained on OpenWebText (Gokaslan et al., 2019) tokenized with the GPT-2 tokenizer (Brown et al., 2020) for 1M optimization steps with effective batch size 512 and sequence length 1024, corresponding to approximately 262B masked tokens. We use AdamW (Loshchilov and Hutter, 2017) with 2500 linear warmup steps, learning rate $3 \cdot 10^{-4}$, and $\beta = (0.9, 0.999)$; remaining hyperparameters follow Sahoo et al. (2024).

For the LM1B replication, we train the same depth-scaled model family on the One Billion Word Benchmark (Chelba et al., 2014) with sequence length 128. All other architecture and training choices remain the same.

B Additional Light Model Results

Additional hand-crafted schedules for light models with 6, 8, and 10 blocks are presented in Figures 6, 7, and 8 and exhibit the same qualitative pattern as the 4-block light model (Figure 1 in the main text). Error bars correspond to 95% confidence intervals.

C Additional Exhaustive Search Results

Figure 9 shows the top-5 best and bottom-5 worst schedules from the exhaustive search (Section 3.3). Figure 10 complements Figure 2 in the main text by showing segment frequency among the 20 *worst*-performing schedules. Middle segments dominate, confirming that replacing them is most harmful.

D LM1B Generalization

To test whether our findings generalize beyond OpenWebText, we trained an identical model family on the One Billion Word Benchmark (LM1B) (Chelba et al., 2014) with 128-token sequence length (see Section 3.1 for details).

Figure 11 and Table 5 show the hand-crafted schedule comparison on LM1B. The same qualita-

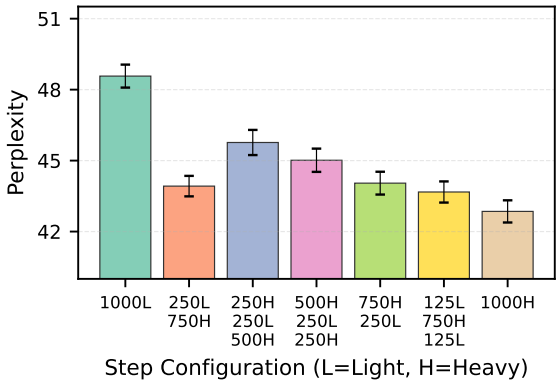

Figure 6: Generative perplexity (GPT-2) for hand-crafted model schedules using a heavy 12-block model and a light 6-block model with exactly 250/1000 light steps. Each bar label encodes a schedule as contiguous segments, e.g., $(\mathrm{L125}, \mathrm{H750}, \mathrm{L125})$ denotes the *sandwich* schedule (125 light steps, 750 heavy steps, 125 light steps).

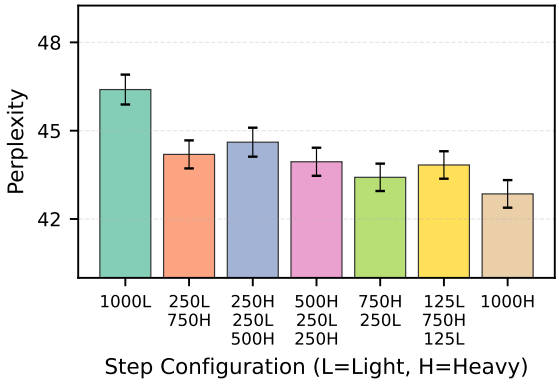

Figure 7: Generative perplexity (GPT-2) for hand-crafted model schedules using a heavy 12-block model and a light 8-block model with exactly 250/1000 light steps. Each bar label encodes a schedule as contiguous segments, e.g., $(\mathrm{L125}, \mathrm{H750}, \mathrm{L125})$ denotes the *sandwich* schedule (125 light steps, 750 heavy steps, 125 light steps).

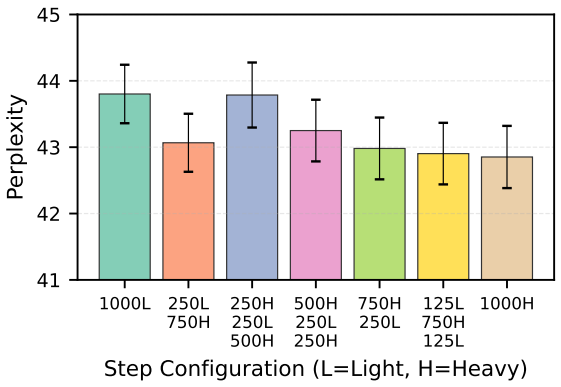

Figure 8: Generative perplexity (GPT-2) for hand-crafted model schedules using a heavy 12-block model and a light 10-block model with exactly 250/1000 light steps. Each bar label encodes a schedule as contiguous segments, e.g., $(\mathrm{L125}, \mathrm{H750}, \mathrm{L125})$ denotes the *sandwich* schedule (125 light steps, 750 heavy steps, 125 light steps).

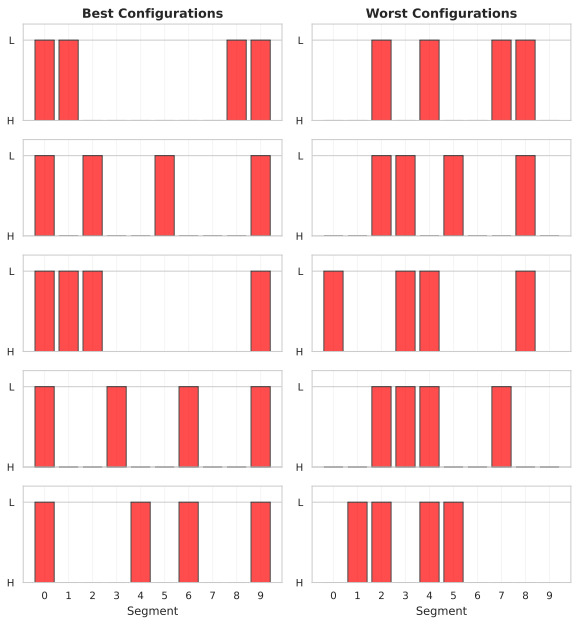

Figure 9: Comparison of the top 5 best (left) and worst (right) model scheduling configurations among the 210 coarse schedules. Each row shows one configuration. Red bars indicate light (4-block) model placement. Segments 0–9 correspond to steps 0–100, 100–200, . . . , 900–1000, where segment 0 is closest to the fully masked state ($t \approx 1$). Best configurations concentrate light segments near both ends, while worst configurations place light segments in the middle.

tive pattern as on OpenWebText (Figure 1) is observed: middle-step replacement yields the worst perplexity, while endpoint and sandwich placements perform best.

Schedule	Gen. PPL (CI)	Entropy
L128	242.0 $\pm$4.69	4.30
L32 $\rightarrow$ H96	207.9 $\pm$3.96	4.25
H32 $\rightarrow$ L32 $\rightarrow$ H64	215.5 $\pm$4.66	4.24
H64 $\rightarrow$ L32 $\rightarrow$ H32	211.9 $\pm$4.83	4.24
H96 $\rightarrow$ L32	209.1 $\pm$4.48	4.24
L16 $\rightarrow$ H96 $\rightarrow$ L16	210.5 $\pm$4.16	4.26
H128	188.8 $\pm$4.35	4.21

Table 5: Generative perplexity (with 95% CI) and token-level entropy for hand-crafted model schedules on LM1B (128-token context, 4-block light / 12-block heavy, 32/128 light steps). **H**eavy: 12-block, **L**ight: 4-block. The same schedule ranking as on OpenWebText (Table 6) is reproduced. Entropy variation across schedules is <0.06 nats.

We also replicate the KL-divergence analysis (Section 4.1) on the LM1B model family. Figure 12 shows the same characteristic peak in the middle of the denoising trajectory, confirming that the non-monotonic step-importance pattern is a general property of masked diffusion rather than an artifact of the OpenWebText setup.

E Unconditional Generation Entropy

Table 6 reports token-level entropy alongside generative perplexity for the unconditional OpenWebText schedules shown in Figure 1. Entropy remains stable across all schedules (range 5.27–5.30), indicating that model scheduling does not reduce sample diversity even under substantial model substitution.

Schedule	Gen. PPL (CI)	Entropy
L1000	53.4 $\pm$0.62	5.27
L250 $\rightarrow$ H750	44.6 $\pm$0.50	5.28
H250 $\rightarrow$ L250 $\rightarrow$ H500	48.0 $\pm$0.54	5.30
H500 $\rightarrow$ L250 $\rightarrow$ H250	47.1 $\pm$0.52	5.29
H750 $\rightarrow$ L250	45.4 $\pm$0.50	5.29
L125 $\rightarrow$ H750 $\rightarrow$ L125	44.3 $\pm$0.50	5.27
H1000	42.9 $\pm$0.47	5.29

Table 6: Unconditional generative perplexity (with 95% CI) and token-level entropy for OpenWebText schedules (same data as Figure 1). **H**eavy: 12-block, **L**ight: 4-block. Entropy variation across schedules is <0.03 nats, confirming stable sample diversity.

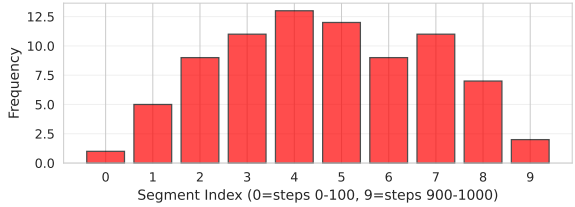

Figure 10: Segment frequency in the bottom 20 worst-performing configurations (highest perplexity) from the exhaustive search (Section 3.3). Bars show how often each segment is assigned to the light (4-block) model. Higher frequency suggests that replacing this segment is harmful.

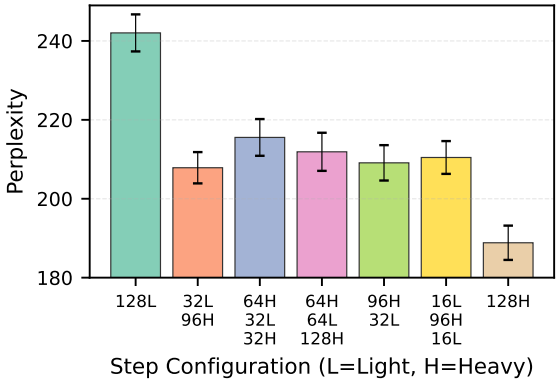

Figure 11: Generative perplexity for hand-crafted model schedules on LM1B (128-token context, 4-block light / 12-block heavy, 250/1000 light steps). The middle-step sensitivity pattern from Figure 1 reproduces on a different dataset and sequence length. Error bars correspond to 95% confidence intervals.

F Prefix-Conditional Generation

To complement the prefix-conditional results with 256-token prefixes reported in the main text (Table 3), we provide the corresponding bar plot (Figure 13) and an additional evaluation with 128-token prefixes (Figure 14, Table 7). In both settings, the same schedule ranking holds: middle-step replacement is most harmful, while the sandwich schedule performs best among mixed schedules.

G KV-Caching Composability

We evaluate DKV-CACHE (Ma et al., 2026) on the same OpenWebText schedules used in the main text, using 800 samples of length 1024. On cache-mode steps, our sequence-shrink implementation computes queries only at active (currently masked) positions and reuses cached keys and values for committed (already-unmasked) positions. We use a refresh interval of c_s=8: cached keys/values are

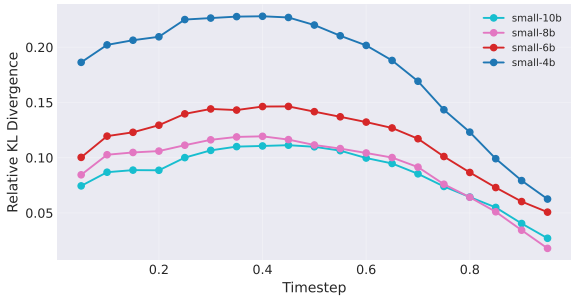

Figure 12: Relative token-level KL divergence (Eq. 6) between model pairs trained on LM1B across timesteps, after subtracting a baseline KL curve computed between two independently trained heavy (12-block) checkpoints. The same middle-trajectory peak observed on OpenWebText (Figure 4) is reproduced, confirming cross-dataset generality.

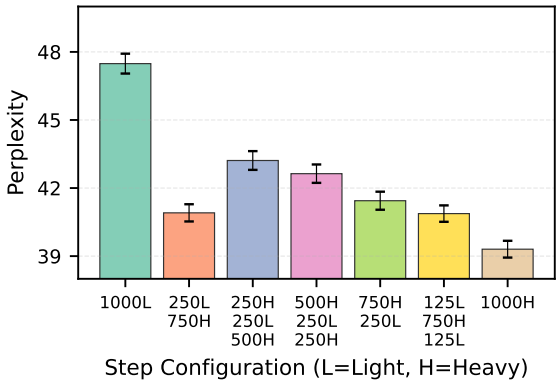

Figure 13: Conditional generative perplexity for hand-crafted schedules on OpenWebText with 256-token prefixes (4-block light / 12-block heavy, 250/1000 light steps). The schedule ranking matches the unconditional setting (Figure 1). Error bars correspond to 95% confidence intervals.

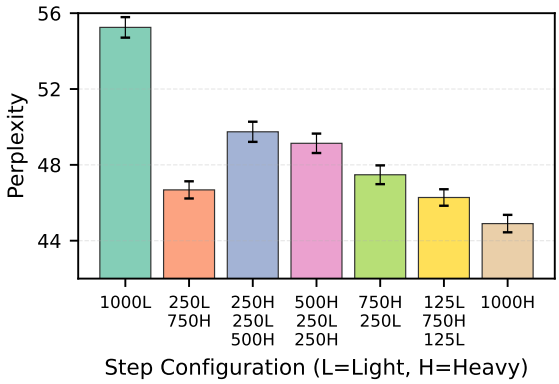

Figure 14: Conditional generative perplexity for hand-crafted schedules on OpenWebText with 128-token prefixes (4-block light / 12-block heavy, 250/1000 light steps). The pattern is consistent with the 256-token prefix setting (Figure 13). Error bars correspond to 95% confidence intervals.

Schedule	Cond. Gen. PPL (CI)	Entropy
L1000	55.3 ± 0.54	5.39
L250 → H750	46.7 ± 0.45	5.39
H250 → L250 → H500	49.7 ± 0.53	5.39
H500 → L250 → H250	49.1 ± 0.51	5.38
H750 → L250	47.5 ± 0.49	5.38
L125 → H750 → L125	46.3 ± 0.43	5.39
H1000	44.9 ± 0.46	5.38

Table 7: Prefix-conditional generation (128-token prefix from OpenWebText). **H**eavy: 12-block, **L**ight: 4-block. The same schedule ranking as the 256-token prefix setting (Table 3) is observed. Entropy remains stable across schedules.

recomputed every 8 denoising steps to bound staleness. When the schedule switches between the light and heavy model, the cache is refreshed before reuse. The full per-schedule results are reported in Table 8.

Across all seven schedules, DKV-CACHE preserves perplexity within 95% confidence intervals (max $\Delta\mathrm{PPL} = 0.7$, on $\mathrm{H}_{250} \rightarrow \mathrm{L}_{250} \rightarrow \mathrm{H}_{500}$) and accelerates every schedule, with the largest speedup ($1.47\times$) on the all-light L_{1000} configuration where per-step cache reuse has the most room to amortize.

Schedule	Gen. PPL ↓		s/sample ↓		Speedup ↑
	no cache	+DKV-CACHE	no cache	+DKV-CACHE	
L_{1000}	$53.4_{\pm 0.62}$	$53.3_{\pm 0.93}$	4.90	4.60	1.47×
$L_{250} \rightarrow H_{750}$	$44.6_{\pm 0.50}$	$44.6_{\pm 0.72}$	6.29	5.47	1.24×
$H_{250} \rightarrow L_{250} \rightarrow H_{500}$	$48.0_{\pm 0.54}$	$48.7_{\pm 0.85}$	6.29	5.54	1.22×
$H_{500} \rightarrow L_{250} \rightarrow H_{250}$	$47.1_{\pm 0.52}$	$47.5_{\pm 0.82}$	6.33	5.65	1.20×
$H_{750} \rightarrow L_{250}$	$45.4_{\pm 0.50}$	$45.8_{\pm 0.78}$	6.29	6.10	1.11×
$L_{125} \rightarrow H_{750} \rightarrow L_{125}$	$44.3_{\pm 0.50}$	$44.3_{\pm 0.75}$	6.29	6.17	1.10×
H_{1000}	$42.9_{\pm 0.47}$	$43.2_{\pm 0.73}$	6.76	5.93	1.14×

Table 8: Full composability sweep on OpenWebText. DKV-CACHE uses refresh interval c_s=8. PPL is GPT-2 generative perplexity with 95% confidence intervals; speedup is relative to uncached H_{1000}. Adding DKV-CACHE preserves quality within uncertainty and improves wall-clock time for every schedule.