LENGTH FROM THE LATENTS: DYNAMIC-LENGTH COMPRESSED LATENT DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models (DLMs) have emerged as a promising alternative to autoregressive models, offering parallel generation and the ability to revise their output through bidirectional attention. However, the adoption of DLMs is still limited due to running diffusion on high-dimensional token-level representations. Some methods overcome this issue by running diffusion in a compressed fixed-dimensional latent space, where the latent space is obtained with a compressor network, and later a decompressor network expanding the denoised latents back to token-level representation followed by decoding them to the output tokens. Yet, current methods run parallel decompression assuming a fixed output sequence length, and thus attend bidirectionally over all output positions, including the ones that are beyond the end-of-sequence token. This results in every sample paying for the maximum length, and in compute wasted on padding. As a solution, we propose PACED (Parallel Append-only Cached Extensible Decompression). PACED masks the decompressor so each output position attends to all latents but only to itself and to preceding positions, ensuring prefix-consistent decompression that is efficiently extendable. We achieve this with the help of training a length head on the denoised compressed latents, the features that are natively length-aware. Post token decoding, if the end-of-sequence token does not appear, we reuse cached internal states of the decompressor, similar to key-value caching, and decompress only the missing positions, instead of entirely rerunning the decompression. We prove this returns exactly the text a full-length decode would have returned, whatever initial length was chosen. Distilling the pretrained decompressor into the prefix-consistent one on sampled latents recovers much of the generation quality while allowing PACED to run batched 1.2–1.8$\times$ faster than full-length decompression. PACED thus makes decompression cost follow the output rather than the maximum length, paving the way for compressed-latent DLMs that support long outputs efficiently.

1 INTRODUCTION

Diffusion language models (DLMs) generate text by iteratively denoising a sequence representation, offering a parallel alternative to autoregressive decoding (Sahoo et al., 2024; Nie et al., 2025). Compressed-latent DLMs (Lovelace et al., 2023; Meshchaninov et al., 2025) run this process in a compact latent space: a compressor maps a passage of up to L tokens to a fixed number $K \ll L$ of continuous latents, a diffusion model learns to generate such latents, and a decompressor maps a generated latent back to L output positions, which are then decoded into tokens. Since the number of latents is fixed, the cost of denoising does not depend on the length of the output.

In contrast, the cost of decompression remains tied to the maximum sequence length: the decompressor evaluates all L output positions in parallel for every sample, and the output is then truncated at the first end-of-sequence (EOS) token, regardless of its actual length. Decompressing fewer positions is not a solution in itself: the decompressor attends bidirectionally over its output positions, so changing their number can change the decoded text, and a length chosen too short truncates the output, requiring one or more additional decompressions at a larger length. An autoregressive decompressor, as in LD4LG (Lovelace et al., 2023), allows the output to be stopped and extended, but gives up the parallelism of decompression, which becomes increasingly costly for long outputs.

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

We propose PACED (**P**arallel **A**ppend-only **C**ached **E**xtensible **D**ecompression), which decompresses only as many positions as each output requires, while keeping decompression parallel (Figure 1). PACED changes only the attention mask of the decompressor: each output position still attends to all latents, but only to itself and to preceding output positions. As a result, a shorter decompression is exactly a prefix of a longer one. Building on this property, a length head predicts the initial number of positions from the denoised latent. If no EOS token appears, the output is extended geometrically, reusing the cached hidden states of the positions already computed and evaluating only the new ones. We prove that, for any initial length, this procedure returns exactly the same output as full-length decompression. The computational savings therefore come with no change to the output, even when the predicted length is inaccurate. This guarantee is specific to compressed-latent DLMs: since the output is fully determined by the denoised latent before decompression begins, the full-length decompression provides a fixed reference that a shorter decompression can be extended to match. In token-space DLMs, by contrast, the length is part of what is being denoised, and changing it alters the generation itself.

Since the mask removes context that the pretrained decompressor relies on, we fine-tune it by distilling the original decompressor on latents sampled from the frozen diffusion model, which recovers much of the generation quality lost by masking. Across three text domains, PACED evaluates 26–53% fewer output positions than full-length decompression and decompresses batched samples 1.2–1.8$\times$ faster, while returning outputs identical to those of full-length decompression.

More broadly, PACED complements the key property of compressed-latent DLMs. Their denoising cost is fixed by the number of latents, independently of the output length, and with PACED the cost of decompression depends on the length of the output rather than on the maximum sequence length. Neither stage then requires the output length to be chosen in advance, which makes compressed-latent DLMs a promising basis for dynamic-length generation that supports long outputs efficiently. The benefits of PACED are expected to grow with few-step samplers, which make decompression a larger share of the generation cost, and with models trained to support longer outputs, for which full-length decompression is most wasteful.

Contributions.

1. **Prefix-consistent decompression.** We introduce an attention mask for compressed-latent decompressors that preserves parallel decompression while making every shorter decompression an exact prefix of a longer one, and we prove this property.
2. **Dynamic-length decompression.** We combine a length head on the denoised latent with cached geometric extension, and prove that the procedure returns the full-length output with a bounded number of evaluated positions.
3. **Distillation and evaluation.** We distill the pretrained decompressor on sampled latents to recover generation quality, and evaluate quality, exactness, positions saved, and batched decompression time on three text domains.

2 Related Work

Existing DLMs differ mainly in the space in which denoising takes place. Most DLMs operate in token space, denoising the full token sequence at every sampling step (Sahoo et al., 2024; Nie et al., 2025). The sequence length must therefore be fixed before denoising starts, and the cost of every step grows with it. Latent DLMs instead run diffusion on continuous representations of text, which can be compressed along two axes. The first is the feature dimension: models such as Diffusion-LM and CoLA keep one latent per token (Li et al., 2022; Guo et al., 2026), so, as in token space, the number of latents to be denoised is tied to the output length and must be set in advance. The second is the sequence dimension: compressed-latent DLMs such as LD4LG and Cosmos map a passage to a fixed number of latents, i.e., at a fixed compression ratio (Lovelace et al., 2023; Meshchaninov et al., 2025). The output length then no longer needs to be specified before denoising, since it is encoded in the denoised latent, and the cost of denoising is independent of the output length and lower than that of denoising at the token level.

To relax the requirement of fixing the output length before denoising, several methods have addressed variable-length generation, primarily in token-space diffusion. For masked diffusion models, these

methods grow the sequence based on the model's confidence (Li et al., 2026), retain inserted mask positions after a counterfactual test (Bouhedja et al., 2026), or read per-position end-of-sequence probabilities as hazard rates (Kobyzev et al., 2026). Other work predicts the response length from the input before denoising (Rottoli et al., 2026) or studies the intrinsic length awareness of diffusion models (Rossi et al., 2026). These methods decide when to stop from an internal signal, without a reference output to verify against, and rely either on the token sequence taking part in iterative denoising or on a full-length forward pass. Neither applies to compressed-latent DLMs, which denoise a fixed-dimensional latent and map it to tokens only afterwards, for instance in a single parallel decompression pass as in Cosmos. The output is thus determined before decompression begins, and the problem is not to choose its length but to decode it faithfully. A Perceiver-style decompressor can be evaluated with a variable number of output queries (Jaegle et al., 2022), but the tokens it returns then depend on the number of requested positions, so an output cannot be extended without being recomputed. This is one of the limitations we address.

A second line of work obtains flexible length through block-causal generation, denoising blocks of tokens (Arriola et al., 2025) or token sets (Arriola & Kuleshov, 2026) sequentially with caching, and has been extended to latent DLMs (Liang et al., 2026; Guo et al., 2026). These latent variants rely on each block of latents corresponding to a span of the output, which holds when only the feature dimension is compressed. Under sequence compression, this correspondence is lost: each latent summarizes the entire passage, as in Cosmos, where the compressor forms every latent by attending over all tokens. A block-causal factorization therefore cannot be applied naively to compressed-latent DLMs without giving up sequence compression, and with it the fixed, length-independent cost of denoising. Notably, recent latent DLMs that report the strongest results among the diffusion models they evaluate keep one latent per token (Liang et al., 2026; Meshchaninov et al., 2026). These methods thus obtain flexible length by modifying the generative process itself, and do not address how a completed compressed latent is decompressed into an output of variable length.

3 PRELIMINARIES

A diffusion model (Ho et al., 2020) generates data by progressively reversing a noising process. For clean samples drawn from the data distribution $x_0 \sim p_{\text{data}}$, the *forward process* corrupts the data as $x_t = \sqrt{\alpha}_t x_0 + \sqrt{1-\alpha_t}\varepsilon$ with $\varepsilon \sim \mathcal{N}(0, I)$ and α_t being the noise schedule dependent on t. In the *reverse process*, a denoising network x_θ is trained with the regression objective function:

$$\mathcal{L}(\theta) = \mathbb{E}_{t,x_0,\varepsilon}\big[\, \|x_0 - x_\theta(x_t, t)\|_2^2 \,\big], \tag{1}$$

where $t \sim \mathcal{U}(0,1)$ is the timestep. During inference, starting from $t = 1$ to $t = 0$ the model x_θ is tasked with iteratively denoising pure noise into a clean output.

3.1 LATENT DIFFUSION

Since inference in diffusion modelling is iterative, it is costly when the inputs are high-dimensional, such as images. Latent Diffusion Models (LDMs) (Rombach et al., 2022; Lovelace et al., 2023) avoid this issue by learning the denoising process in a lower-dimensional latent space rather than directly in the input space (e.g., pixels or tokens). A pretrained encoder $E(x)$ maps the input to a compressed latent space as $z = E(x)$. The denoising process described above occurs in this continuous latent space, which is more compact than the input space. After denoising is complete, a pretrained decoder $D(z)$ maps the denoised latents back to the input space as pixels or tokens. In the domain of language modelling, two notable methods exist: LD4LG (Lovelace et al., 2023) and Cosmos (Meshchaninov et al., 2025), both built on the idea of diffusion modelling in a compressed latent space. Without loss of generality, we work with Cosmos, whose architecture we describe next.

Cosmos has three main components: a *feature extractor*, a *compressor-decompressor* network, and a *decoder*. In the feature extraction stage, it uses a pretrained text encoder E (initialized from BERT-base (Devlin et al., 2019)) to project an input text sequence of L tokens into hidden representation $h = E(x) \in \mathbb{R}^{L \times d}$, where $d = 768$. This hidden representation serves as a strong starting point for diffusion modelling. Considering that input sequence L can be variable, the compressor network, which employs a Perceiver Resampler architecture (Alayrac et al., 2022), transforms it into a fixed-length sequence of hidden representation of K embeddings, with $K \ll L$. Concretely, the compressor

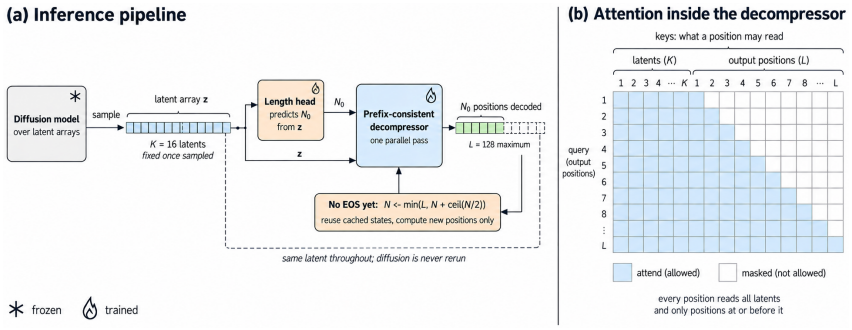

Figure 1: Overview of PACED. **(a)** Inference: a length head predicts an initial length N_0 from the sampled latent, and the decompressor evaluates these positions in one parallel pass. If no end-of-sequence token appears the length grows by half and only the new positions are computed; those already decompressed are reused as context and their tokens cannot change. The latent is fixed throughout the decompression. **(b)** Attention mask: each output position attends to all latents, but only to itself and to preceding output positions.

uses a set of learnable vectors, $u \in \mathbb{R}^{K \times d}$, which act as *queries*. The *keys* and *values* are derived from the concatenation of the hidden states of the encoder and the queries $[u; h]$. The compressor uses cross-attention to compress information from the hidden states into fixed-length latents $z \in \mathbb{R}^{K \times d}$ that correspond to queries u. A diffusion model z_θ operates on the space formed by z using Equation 1 to learn and generate samples. A decompressor, which mirrors the compressor, takes the denoised latents and reprojects them to a sequence of L vectors $\hat{h} \in \mathbb{R}^{L \times d}$, where L is the *maximum sequence length*. Finally, the decoder – a linear projection layer and softmax – converts each embedding in $\hat{h}$ into a probability distribution over a vocabulary V, which yields the output text. We refer the reader to Cosmos (Meshchaninov et al., 2025) for further details.

4 METHOD

By design, the Cosmos decompressor always evaluates all L output positions it supports. Many outputs, however, end well before position L (Rossi et al., 2026): every position after the first end-of-sequence (EOS) token is decompressed, decoded, and then discarded. Our method, PACED (**P**arallel **A**ppend-only **C**ached **E**xtensible **D**ecompression), instead evaluates only as many positions as the output needs. This requires three components, which structure this section. First, since the latent is fully denoised before decompression begins, a length head predicts the required number of positions from it (Section 4.1). Second, when this prediction is too short, decompression should continue from where it stopped rather than restart at a larger length; this is correct only if appending positions leaves the already computed ones unchanged, which we guarantee with an attention mask (Section 4.2). Third, to make such an extension cheap, we cache the states of computed positions and evaluate only the new ones (Section 4.3). Finally, because the mask changes what the decompressor sees, we fine-tune it to recover quality (Section 4.4). Throughout, PACED leaves the compressor and the diffusion model unchanged, and the sampled latent stays fixed, so choosing or extending the output length does not require another diffusion run.

4.1 PREDICTING THE INITIAL OUTPUT LENGTH

In compressed-latent diffusion, the output is fully denoised into a fixed-dimensional latent before decompression begins. The length required by the content is therefore already encoded in the latent, which is natively length-aware, similarly to what Rossi et al. (2026) observe in token-space DLMs. This allows us to predict the output length from the latent prior to decompression.

Let $D_L(z) \in \mathbb{R}^{L \times V}$ denote the vocabulary logits from decompressing latent z at the maximum sequence length L. Let $y_i = \arg\max_{v \in \{1,\ldots,V\}} D_L(z)_{i,v}$, with ties broken deterministically. The required number of positions is

$$R(z) = \begin{cases} \min\{i \in \{1, \ldots, L\} : y_i = \text{EOS}\}, & \text{if EOS occurs,} \\ L, & \text{otherwise.} \end{cases} \tag{2}$$

The decompressed length includes the EOS position because PACED must generate one to identify that the output is complete; it then returns the tokens before the first EOS token.

As shown in Figure 1, a length head g_ψ estimates $R(z)$ from the sampled latent z. We train it with the pinball loss at the 0.95 quantile on encoded texts and independently sampled latents, using as targets the EOS positions produced by full-length decompression. The high quantile favors over-allocation, which avoids an extension when the estimate is uncertain. A held-out calibration set supplies an additional conservative residual margin m, so PACED initially decodes

$$N_0 = \mathrm{clip}_{[2,L]}(\mathrm{round}\,[g_\psi(z) + m]) \tag{3}$$

positions. The head affects compute only: an overestimate evaluates unused positions, while an underestimate triggers extension.

4.2 Prefix-consistent decompression

However, extending the output is not straightforward with the original decompressor. Its cross-attention uses the output positions as queries and the concatenation of the latents and all output positions as keys and values, so appending new positions alters the hidden states of the existing ones and may therefore change the tokens already decoded. To ensure that extension leaves the existing output unchanged, PACED keeps the cross-attention to the latents unchanged and masks only the attention among output positions: each output position still attends to all latents, but only to itself and to preceding output positions. At layer ℓ, the hidden state $h_i^{(\ell)}$ at output position i can attend to positions in the sequence $[z_1, \cdots, z_K, h_1^{(\ell-1)}, \cdots, h_L^{(\ell-1)}]$ given by the mask

$$\mathcal{M}_{h_i^{(\ell)}, z_k} = 1,\ \mathcal{M}_{h_i^{(\ell)}, h_j^{(\ell-1)}} = \mathbb{1}[j \le i], \tag{4}$$

where $1 \le i, j \le L$ and $1 \le k \le K$. That is, every $h_i^{(\ell)}$ attends to all denoised latents z_k, but to $h_j^{(\ell-1)}$ only when $j \le i$. Although this resembles *causal masking*, the decompressor is not autoregressive: all output positions are computed in parallel from fixed queries, rather than one token at a time.

More precisely, fix a latent z and request any output length of $N \le L$ positions. The logit vector at each of its first N positions is identical to the logit vector at that position when the same decompressor evaluates all L positions:

$$D_N(z)_i = D_L(z)_i \quad \text{for every } 1 \le i \le N. \tag{5}$$

Here $D_N(z)_i$ is the vocabulary-logit vector at output position i when N positions are requested. We call Equation (5) the *prefix property*. It compares two output lengths for the same masked decompressor and the same latent.

Proposition 1 (Prefix property) *Fix z and let $2 \le N \le L$. Suppose the decompressor is deterministic, uses the mask of Equation* (4)*, has positional embeddings and latent conditioning that do not depend on N, and applies its residual, normalization and feed-forward operations to each position separately. Then for every layer ℓ and every $i \le N$, the state $h_i^{(\ell)}$ is the same whether we decompress N positions or L. Equation* (5) *follows.*

Proof. The initial query at position i depends only on i. If the two runs agree through position N before a layer, then for every $i \le N$ they give that position the same latents and the same earlier output states. Its attention result and position-wise computation therefore agree. Induction over layers, followed by the shared vocabulary projection, proves the claim. □

4.3 Cached extension

A naive extension would rerun the full decompression at the larger length, re-evaluating all positions already computed. We instead extend the output efficiently by leveraging the masking of Section 4.2: since output positions do not attend to later ones, their hidden states are unaffected by an extension and can be cached and reused, similar to key-value caching in autoregressive models. Concretely, PACED stores the hidden states of positions 1 through N_0 at every layer and, if no EOS appears,

evaluates only the appended positions. These attend to all latents, to the cached positions, and to preceding appended positions, and are computed together in a single parallel pass. If N_t is the output length after t extensions, we increase it geometrically,

$$N_{t+1} = \min\{L, N_t + \lceil N_t/2 \rceil\}, \tag{6}$$

until EOS appears or the maximum length L is reached. We refer to this cached extension procedure as *recovery*. We use this simple geometric rule to limit the number of extensions; more sophisticated data-driven extension rules are left to future work.

Proposition 2 (Exact recovery) *Under the assumptions of Proposition 1, this procedure returns the same EOS-trimmed token sequence as $D_L(z)$, for any $N_0 \geq 2$.*

Proof. By Proposition 1, cached prefix states equal the corresponding states of a full-length run. Each appended position therefore receives exactly the latent and earlier-position states it would receive in that run. Induction over appended positions and layers gives the same token sequence, and the increasing, capped schedule reaches the first EOS or L. □

Computational cost. The returned text does not depend on N_0; only the number J of evaluated positions does. It obeys

$$R(z) \leq J \leq \min\{L, \max(N_0, \lceil \tfrac{3}{2} R(z) \rceil)\}. \tag{7}$$

The lower bound is the first EOS. If the initial allocation is short, the final geometric extension overshoots it by at most one half. There are at most $\lceil \log_{3/2}(L/N_0) \rceil$ extensions; cached positions are never evaluated again, so the total is J, not the sum of the intermediate output lengths. Thus L is only a cap, whereas full-length decompression always evaluates all L positions.

4.4 Fine-tuning the decompressor

The mask removes future output-position context, so the pretrained decompressor cannot simply be reused. We adapt the masked decompressor on latents sampled from the frozen diffusion model—the latents it will see at inference. Such a latent has no reference text. Instead, the pretrained unmasked decompressor supplies the target. For a sampled latent z, let $T_L(z)$ be the teacher's full-length logits, let y^T be its argmax tokens, and let $\mathcal{I}$ be positions through its first EOS (or all L positions if there is none). At a student output length $N \geq \max \mathcal{I}$, we minimize

$$\mathcal{L}_{\text{sample}} = \tfrac{1}{2} \operatorname{CE}_{\mathcal{I}}\left(D_N(z), y^T\right) + \tfrac{1}{2}\tau^2 \operatorname{KL}_{\mathcal{I}}\left[\operatorname{softmax}(T_L(z)/\tau) \,\|\, \operatorname{softmax}(D_N(z)/\tau)\right]. \tag{8}$$

The two equal-weight terms match the teacher's selected tokens and its token distribution softened with temperature $\tau = 2$. Three of every four updates use this sampled-latent objective; the fourth is clean reconstruction on an encoded text. This both matches the latent distribution seen at inference and recovers much of the quality lost by masking; remaining training details are in Appendix A.

5 Experiments

5.1 Protocol

We ask whether the prefix property holds in practice, whether distillation improves quality, and whether adaptive length returns the full-length output while evaluating fewer positions. The main paired study uses ROCStories (Mostafazadeh et al., 2016), CNN/DailyMail (Hermann et al., 2015) and ELI5 (Fan et al., 2019), with domain-specific Cosmos diffusion models and a shared autoencoder configuration: $K = 16$, $L = 128$, and a 12-layer decompressor of width 768. These are unconditional generation experiments. Each domain has independent saved training, validation and test sets of 1,024, 128 and 256 latents, and every decompressor in a paired comparison is evaluated on identical test latents. We fine-tune each decompressor for 1,000 steps with three random seeds, keeping the starting checkpoint and latent sets fixed. For the adaptive-length experiments, we perform inference with one of these decompressors, selected in advance, on the same 256 test latents. The length head is fitted and calibrated on a separate set of 1,024 sampled latents that does not overlap with the training, validation, or test sets, so that its miss rate and savings are measured on unseen latents (Appendix A).

Generated text is evaluated with GPT-2 (Radford et al., 2019) after EOS trimming. We report sequence and corpus perplexity, MAUVE, adjacent-word repetition, prefix agreement, the initial miss rate $\Pr[N_0 < R(z)]$ and positions saved, $1 - \mathbb{E}[J]/L$, all defined in Appendix A.

Table 1: Generation quality on the 256 test latents per domain. The first two rows use full attention over output positions; the last two are prefix-consistent. For ours, we report the mean and [min, max] over three fine-tuning seeds. The human split-half row is a reference for attainable MAUVE. Metric details are in Appendix A.

Decompressor variant	Sequence PPL ↓	Corpus PPL ↓	MAUVE ↑
ROCStories			
Pretrained Cosmos	35.87	32.63	0.898
Reconstruction-adapted	42.70	37.64	0.870
Prefix-consistent, earlier recipe	44.13	38.89	0.842
Prefix-consistent + distillation (ours)	37.43 [37.40, 37.48]	34.00 [33.95, 34.07]	0.893
Human text (split-half)	n/a	n/a	0.950
CNN/DailyMail			
Pretrained Cosmos	202.07	166.88	0.096
Reconstruction-adapted	292.50	233.57	0.084
Prefix-consistent, earlier recipe	307.40	249.55	0.070
Prefix-consistent + distillation (ours)	238.07 [234.63, 241.16]	196.85 [196.39, 197.10]	0.084
Human text (split-half)	n/a	n/a	0.953
ELI5			
Pretrained Cosmos	138.18	101.08	0.527
Reconstruction-adapted	186.28	138.91	0.426
Prefix-consistent, earlier recipe	186.03	145.88	0.387
Prefix-consistent + distillation (ours)	157.63 [156.70, 159.00]	115.65 [115.39, 116.13]	0.476
Human text (split-half)	n/a	n/a	0.968

5.2 Prefix consistency

For the nine fine-tuned decompressors we compare decompressions of length 32, 64 and 96 against the full 128-position decompression on all 256 test latents: all 6,912 comparisons agree exactly at the token level in fp32, and cached recovery separately reproduces all 256 full-length texts per domain.

A streaming test over lengths 16, 32, . . ., 128 illustrates the practical difference: the full-attention decompressor rewrites 9.22% of the positions it has already emitted and no sequence is append-only, while ours rewrites nothing (Appendix A).

5.3 Generation quality

Table 1 compares four decompressors: Cosmos as pretrained; the same model after 4,000 further steps of clean reconstruction; our prefix-consistent decompressor distilled from that reconstruction-adapted model; and the same decompressor under our recipe, distilling the pretrained model on sampled latents. The second row motivates our choice of teacher: reconstruction adaptation degrades a decompressor on sampled latents, so we distill the pretrained one. Distillation improves on the earlier recipe across all three seeds in every domain, sequence perplexity falling 15.2% on ROCStories (44.13 to 37.43), 22.6% on CNN/DailyMail and 15.3% on ELI5, leaving gaps to the pretrained decompressor of 4.3%, 17.8% and 14.1%.

Distributional quality. GPT-2 perplexity rewards conservative text and cannot distinguish a fluent decompressor from a degenerate one, so Table 1 also reports MAUVE, which reads the gap differently. On ROCStories the distilled decompressor scores 0.893 against the pretrained decompressor's 0.898, up from 0.842, closing 91% of the distributional gap where perplexity still shows a 4.3% deficit, and 63% on ELI5, 54% on CNN/DailyMail. The two metrics disagree about what the mask costs, and the one designed for generated-text distributions is the more favorable. Two caveats apply: on CNN/DailyMail every system scores below 0.1, and a split-half anchor near 0.95 places 0.893 close to, but not at, attainable quality.

5.4 Adaptive length and recovery

Table 2 reports the pretrained decompressor and three prefix-consistent configurations per domain, the latter sharing weights and paired latents. Initial miss rates are 2.3%, 4.3% and 5.1%. Without

Table 2: Adaptive-length decompression on the 256 test latents per domain. *Saved*: fraction of the 128 output positions not evaluated. *Initial miss*: fraction of latents whose predicted length is too short. *Text match*: outputs identical to our full-length decompression.

System	PPL	Saved (%)	Initial miss (%)	Text match
ROCStories				
Pretrained Cosmos (full length)	35.87	0.0	n/a	n/a
Ours (full length)	37.40	0.0	0.0	256/256
Ours (predicted length)	37.43	54.1	2.3	251/256
Ours (predicted + recovery)	37.40	53.4	2.3	256/256
CNN/DailyMail				
Pretrained Cosmos (full length)	202.07	0.0	n/a	n/a
Ours (full length)	238.43	0.0	0.0	256/256
Ours (predicted length)	238.50	43.5	4.3	250/256
Ours (predicted + recovery)	238.43	42.8	4.3	256/256
ELI5				
Pretrained Cosmos (full length)	138.18	0.0	n/a	n/a
Ours (full length)	156.70	0.0	0.0	256/256
Ours (predicted length)	157.00	27.2	5.1	245/256
Ours (predicted + recovery)	156.70	26.1	5.1	256/256

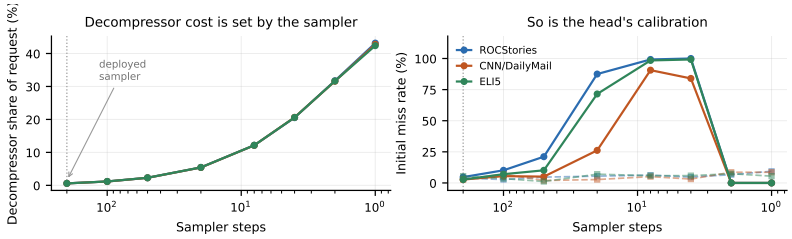

Figure 2: Effect of the number of sampling steps. **Left**: share of the generation cost spent on decompression. **Right**: initial miss rate of a length head calibrated at 200 steps (solid) and of one refitted at each step count (dashed). Below four steps, the outputs degenerate and the head predicts the full length.

recovery, aggregate perplexity barely changes, but this does not ensure that individual outputs are intact, which recovery guarantees.

Recovery reduces savings from 54.1% to 53.4%, 43.5% to 42.8% and 27.2% to 26.1%, since only 6, 11 and 13 of 256 items need an extension. In our $L = 128$ configuration, mean required lengths of 54 to 88 leave limited room for position savings; the observed 26–53% therefore likely understate savings in otherwise comparable models trained with larger maximum sequence lengths.

5.5 Sampling steps, length-head calibration, and batched serving

We locate the regime in which decompression cost matters by reducing the number of sampling steps (Figure 2, Appendix E). A full-length decompression costs about one diffusion step, so its share runs from 0.6% at the 200 steps these systems use today to 43% at one step, the three domains agreeing to within a point: the ratio is a property of the sampler, not of the text.

The sweep also stress-tests the length head. As the step count falls the distribution of required lengths moves, and a head calibrated once at 200 steps degrades monotonically, its miss rate rising from 2.7–4.7% at the default 200-step sampler to 100%, while one refit at each step count stays calibrated, indicating that the degradation lies in the calibrated margin. Across the 48 (domain, step count, head) configurations, recovery reproduced the full-length text on all 12,288 items (fp64; Appendix F examines numerical precision).

The sweep reduces the step count rather than distilling a few-step sampler, and latent quality degrades as it falls (Appendix E), so it tests the length head and recovery rather than few-step quality. The head is also cheap to re-fit: recalibrating only its scalar margin on k=16 sampled latents returns all

eight configurations in which the calibrated head fails to under 8% miss, and k=64 lands within 1.6 points of a margin fit on the full pool.

In batched decompression, positions saved translate into time: requests are sorted by predicted length and grouped; each batch is decompressed to its longest prediction, and the members that still lack an end-of-sequence token are extended together from their cached states. Serving the 256 test latents this way in batches of 16 and 64, PACED decompresses 1.7–1.8× faster than full-length decompression on ROCStories, 1.5× on CNN/DailyMail and 1.2× on ELI5, with every output identical (Table 8). At batch size one a decompression is latency-bound, so the gain appears once requests are batched.

6 Discussion and Limitations

The exactness guarantee of PACED holds with respect to a fixed latent and the maximum sequence length L: extension reproduces the full-length output of the same decompressor, including its stopping behavior, but does not improve it. This guarantee is also what allows a learned length head to be used safely: an underestimated length only triggers an extension, and an overestimated one only evaluates unused positions, but neither changes the output. The main cost of the method lies in the attention mask, which removes context that the pretrained decompressor relies on. Distillation on sampled latents recovers much of the resulting quality loss (Table 1), and relaxing the mask to groups of positions recovers more, at the price of stopping only at multiples of the group size (Appendix C). The end-to-end benefit also depends on the sampler: with many-step samplers, denoising dominates the generation cost, and the length head must be recalibrated when the sampler changes (Section 5.5). Finally, PACED does not extend generation beyond L, which is fixed by the compression ratio of the pretrained model; longer text requires further latents that only the diffusion model can provide.

Looking ahead, compressed-latent diffusion is a promising basis for DLMs that generate long outputs efficiently: sequence compression ties the cost of fully bidirectional latent generation to a fixed number of latents rather than to the output length. This separation of latent generation from decoding into tokens is also what enables PACED: once denoising has completed, the output length can be chosen and extended without rerunning the diffusion model. Three directions follow directly. First, few-step latent samplers make decompression a larger share of the generation cost, reaching 43% at a single step in our sweep, and therefore make dynamic-length decompression increasingly consequential. Second, training compressed-latent models with larger maximum sequence lengths would target the setting in which full-length decompression is most wasteful, since PACED decodes a short output without paying for the longest output the decompressor supports. Third, serving systems can use more than a point estimate of the length, with batching policies that account for the predicted length distribution and regroup requests that require extension, reducing the padding introduced by batching (Section 5.5). Beyond the capacity of a single latent, PACED provides the dynamic decoding component, decoding each completed latent without fixing its output length in advance. Combined with models that generate further latents conditioned on previous ones, it would enable generation beyond L, a promising direction given that existing block-causal methods cannot be applied directly to sequence-compressed latents (Section 2).

7 Conclusion

We presented PACED, a dynamic-length decompression method for compressed-latent DLMs. As the denoised latent already determines the output and encodes its length, the output length does not need to be fixed before decompression. By restricting attention among output positions to the current and preceding ones, PACED ensures that a shorter decompression is an exact prefix of a longer one. A length head predicts how many positions to decompress, and if this prediction is too short, the output is extended by reusing the positions already computed, with a result identical to full-length decompression. Across three text domains, PACED evaluates 26–53% fewer positions and decompresses batched samples 1.2–1.8× faster than full-length decompression. Since the cost of denoising is already independent of the output length, PACED makes compressed-latent DLMs a promising foundation for efficient long-output generation.

Reproducibility Statement

Appendix A specifies data roles, frozen components, adaptation and generation seeds, checkpoint selection, metrics, and timing scope; Appendix E does the same for the sampler-step study. Reproducing model execution also requires the underlying Cosmos implementation, datasets, and checkpoints.

LLM Usage Statement

LLM-based assistance was used for literature discovery, implementation, experiment orchestration, and manuscript editing. The authors are responsible for verifying the proofs, code, sources, and reported results.

References

Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katherine Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *Advances in neural information processing systems*, 35:23716–23736, 2022.

Marianne Arriola and Volodymyr Kuleshov. Set diffusion: Interpolating token orderings between autoregression and diffusion for fast and flexible decoding. In *International Conference on Machine Learning*, 2026. arXiv:2607.01775.

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2025. Oral; arXiv:2503.09573.

Wail Bouhedja, Amr Mohamed, and Guokan Shang. CARVE: Verified expansion for variable-length generation in diffusion language models. In *Findings of the Association for Computational Linguistics: EMNLP*, 2026. arXiv:2608.30922.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 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, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Angela Fan, Yacine Jernite, Ethan Perez, David Grangier, Jason Weston, and Michael Auli. ELI5: Long form question answering. In *Association for Computational Linguistics (ACL)*, 2019.

Hongcan Guo, Qinyu Zhao, Yian Zhao, Shen Nie, Rui Zhu, Qiushan Guo, Feng Wang, Tao Yang, Hengshuang Zhao, Guoqiang Wei, and Yan Zeng. Continuous latent diffusion language model. *arXiv preprint arXiv:2605.06548*, 2026.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2015.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Andrew Jaegle, Sebastian Borgeaud, Jean-Baptiste Alayrac, et al. Perceiver IO: A general architecture for structured inputs and outputs. In *International Conference on Learning Representations (ICLR)*, 2022. arXiv:2107.14795.

Ivan Kobyzev, Abbas Ghaddar, and Yufei Cui. Survival-guided length control for efficient diffusion language models. In *Empirical Methods in Natural Language Processing (EMNLP)*, 2026.

Jinsong Li, Xiaoyi Dong, Yuhang Zang, Yuhang Cao, Jiaqi Wang, and Dahua Lin. Beyond fixed: Training-free variable-length denoising for diffusion large language models. In *International Conference on Learning Representations (ICLR)*, 2026. arXiv:2508.00819.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori B. Hashimoto. Diffusion-LM improves controllable text generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. arXiv:2205.14217.

Jiajun Liang, Yucheng Liao, Yukang Cao, Jiazhe Wei, Ken Li, Wende Tan, Jiankun Zhang, ZY Cui, Jingkang Yang, Liucheng Guo, Shiqi Yang, B. Yang, Caifeng Shan, Ziwei Liu, and Chenyang Si. AURORA-LM: Autoencoding unified representation for continuous-latent diffusion language modeling. *arXiv preprint arXiv:2608.02602*, 2026.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Shekhtman, and Kilian Q. Weinberger. Latent diffusion for language generation. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023. arXiv:2212.09462.

Viacheslav Meshchaninov, Egor Chimbulatov, Alexander Shabalin, Aleksandr Abramov, and Dmitry Vetrov. Cosmos: Compressed and smooth latent space for text diffusion modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Viacheslav Meshchaninov, Alexander Shabalin, Egor Chimbulatov, Nikita Gushchin, Ilya Koziev, Alexander Korotin, and Dmitry Vetrov. How to train your latent diffusion language model jointly with the latent space. *arXiv preprint arXiv:2605.07933*, 2026.

Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, et al. A corpus and cloze evaluation for deeper understanding of commonsense stories. In *NAACL-HLT*, 2016.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 2019.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *2022 IEEE/CVF conference on computer vision and pattern recognition (CVPR)*, pp. 10674–10685. ieee, 2022.

Vittorio Rossi, Giacomo Cirò, Davide Beltrame, Luca Gandolfi, Paul Röttger, and Dirk Hovy. Diffusion language models are natively length-aware. *arXiv preprint arXiv:2603.06123*, 2026.

Michael Rottoli, Subhankar Roy, and Stefano Paraboschi. Predict-then-diffuse: Adaptive response length for compute-budgeted inference in diffusion LLMs. In *International Joint Conference on Neural Networks (IJCNN)*, 2026.

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. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

A Protocol and Implementation Details

A.1 Data and model roles

The main experiments reuse existing ROCStories, CNN/DailyMail and ELI5 diffusion checkpoints at step 180,000. The shared autoencoder checkpoint is the 128-token, 16-latent Wikipedia model at step 100,000. Fine-tuning changes 126,266,162 decompressor parameters; it neither retrains the diffusion model nor changes the compressor. The domain-specific masked (prefix-consistent) and reconstruction-adapted full-attention checkpoints are distinct from the pretrained decompressor. Their domain adaptation uses 4,000 steps, learning rate 10^{-4} and 8,000 clean training texts, with the masked stage distilling the reconstruction-adapted teacher. ROCStories uses this same preparation recipe rather than its older mixed-corpus checkpoint. The subsequent sampled-latent stage uses the pretrained full-attention decompressor as teacher, whose generation quality is stronger in the paired comparison.

ROCStories uses its target story field, CNN/DailyMail the highlights field and ELI5 the prepared answer field. Training examples come from the local training split, and no reference evaluation text is used to fit the head. The evaluation samples in the main tables are sampled latents, not encodings of held-out documents, so a domain name denotes the distribution used to train the diffusion model and adapt the decompressor rather than a conditional benchmark task. Tokenized sequences are capped at 128 positions, which limits what these corpora can support.

Table 3: Main paired protocol, identical in structure across the three domains. Seeds in different rows have different roles.

Component	Setting
Diffusion sampler	200 steps; Euler; tanh schedule ($d = 5$, Cosmos); self-conditioning
Decompressor	12 layers; width 768; 12 heads; 16 latent inputs
Training latent set	1,024 sampled latents; generation seed 1701
Validation latent set	128 sampled latents; generation seed 2701
Test latent set	256 sampled latents; generation seed 3701
Clean fine-tuning pool	First 512 prepared training examples
Fine-tuning randomness	Seeds 701, 702, 703; fixed starting checkpoint
Fine-tuning optimizer	AdamW; learning rate $2 \cdot 10^{-5}$; batch size 16
Fine-tuning budget	1,000 steps; gradient norm clipped to 1.0
Fine-tuning mixture	3 sampled steps per 1 clean reconstruction step
Distillation	Equal CE and temperature-scaled KL; temperature 2
Selection	Lowest validation teacher-token CE; steps 0, 250, 500, 750, 1000
Head latent set	Independent 1,024 sampled latents; generation seed 4701
Head fitting/calibration	First 768 / last 256 latents in that set
Additional head data	3,000 clean training texts
Head optimization	Seed 801; AdamW 10^{-3}; 40 epochs; batch size 64
Head architecture	Mean/max pooling; MLP widths 512, 256; dropout 0.1
Length loss	Pinball quantile 0.95; held-out empirical residual margin
Inference	Decompressor seed 701; fp32; geometric growth factor 1.5

Training uses the full 128 positions with probability 0.2, and otherwise the maximum supervised target length in the minibatch plus a uniformly sampled integer from 0 to 8, clipped to L with a minimum width of 4. The supervised mask includes the first EOS and excludes padding. The group ablation additionally rounds widths up to a multiple of the group size. The initial student and every selected final student undergo prefix checks, and nonfinite losses abort the run. Replication runs reuse the saved sampled-latent sets and their hashes rather than relying on nominally equal generation seeds after different RNG-consuming operations.

Streaming test. The append-only claim of Section 4.3 is measured on ROCStories by decompressing successive lengths $16, 32, \ldots, 128$ and counting, at each extension, how many already-evaluated positions change. The full-attention decompressor changes 10,576 of 114,688 such positions (9.22%), and not one of its 256 sequences is append-only under this schedule; the prefix-consistent one changes none, and all 256 are. These counts include positions beyond EOS, so they measure the property

Table 4: All sampled-latent adaptation runs. Each seed shares its domain's starting prefix-consistent checkpoint and saved train/validation/test latents. Selection uses validation teacher-token cross-entropy. Prefix checks compare 32, 64, and 96 positions with a full length on 256 test latents.

Domain	Seed	Selected step	Seq. PPL	Corpus PPL	Prefix checks
ROCStories	701	1000	37.40	33.99	768/768
ROCStories	702	1000	37.48	33.95	768/768
ROCStories	703	1000	37.40	34.07	768/768
CNN/DailyMail	701	1000	238.43	197.10	768/768
CNN/DailyMail	702	1000	241.16	197.08	768/768
CNN/DailyMail	703	1000	234.63	196.39	768/768
ELI5	701	1000	156.70	115.42	768/768
ELI5	702	1000	159.00	116.13	768/768
ELI5	703	1000	157.18	115.39	768/768

itself rather than only the visible text: a decompressor can rewrite discarded positions while leaving the trimmed string intact, and such rewrites should not count as preserving the prefix.

A.2 Recovery algorithm and cache semantics

For a fixed denormalized latent z, maintain the current length N, the predicted token sequence y, and a cache $C^{(\ell)}$ of layerwise output-position states. The initial cache is empty. To append positions $a+1$ through b, where $a = N$ before the extension, proceed as follows:

1. Initialize mask-token queries at absolute positions $a+1, \ldots, b$.
2. At layer ℓ, each new position attends to all latents, the cached states for positions $1, \ldots, a$, and states at earlier or equal positions in the new block. It cannot attend to later positions in that block.
3. Compute attention, residual and feed-forward outputs for the new positions only, then append their layerwise states to $C^{(\ell)}$.
4. Apply the vocabulary projection to the new final-layer outputs and append their argmax IDs to y.

Evaluate positions 1 through N_0 first and set $N = N_0$. While y contains no EOS and $N < L$, set $b = \min(L, N + \lceil N/2 \rceil)$, append positions $N+1$ through b, and set $N = b$. Return the prefix of y before EOS, or all L tokens if EOS never appears. Predicted token IDs are never substituted as query embeddings. Cached states are not recomputed, but their key and value projections currently are, so this is a layer-state cache rather than a persistent projected key/value cache. The distinction matters for the memory and latency numbers.

A.3 Metrics

Generated text is evaluated with GPT-2 (Radford et al., 2019) after EOS trimming. For a generated string s, let n_s be the number of next-token predictions and ℓ_s their mean negative log-likelihood; sequence PPL is $|S|^{-1} \sum_s \exp(\ell_s)$ and corpus PPL is $\exp(\sum_s n_s \ell_s / \sum_s n_s)$, over the set S of scorable outputs. The two can differ substantially, and neither is the generative model's own likelihood. Text with fewer than two evaluator tokens is excluded and counted: ROCStories test latent 178 gives an empty output for each fine-tuned seed, so those PPL rows use 255 outputs while all other quality rows use 256. On the common 255 IDs, seed-701 pretrained, prefix-consistent-initialization and fine-tuned PPL are 35.92, 44.19 and 37.40. All agreement and length tests retain that example.

We also report adjacent-word repetition, the fraction of adjacent word pairs containing the same word; pooled distinct bigrams; token-prefix agreement; the initial miss rate $\Pr[N_0 < R(z)]$, defined against the same decompressor's full-length output; and distinct positions saved, $1 - \mathbb{E}[J]/L$. MAUVE (Pillutla et al., 2021) is computed on a separate set of 1,024 freshly sampled latents, shared by all decompressors within a domain, against an independent reference set with a `gpt2` featurizer; it is therefore not comparable with work using the default `gpt2-large` featurizer, although comparisons

Table 5: Additional ROCStories generation evidence: three generation/head seeds, 512 samples each, one fixed prefix-consistent decompressor, whose full-length and predicted-length outputs share latents within each seed. The pretrained decompressor is evaluated on separately sampled latents and is not a paired comparator. PPL is mean external sequence perplexity; saved positions are relative to the full 128 positions. Prefix match compares the predicted-length prefix to the full-length decompression, not complete EOS-trimmed texts. This table does not evaluate cached recovery or the new sampled-latent adaptation recipe.

System	PPL	Saved (%)	Prefix match
Pretrained Cosmos (full length)	41.27	0.0	n/a
Ours (full length)	47.49	0.0	reference
Ours (predicted length), no recovery	47.65	55.9	1536/1536

across our decompressors are internally consistent. Paired bootstrap intervals for CNN/DailyMail and ELI5 resample the same latent indices for both decompressors, with 2,000 resamples and bootstrap seed 801. These metrics characterize fluency and distributional match under GPT-2, not task accuracy or factuality.

Does the soft-target term matter? A controlled ablation supports it, on a single run. A hard-target-only group-8 student shares the initial checkpoint and sampled-latent-set hashes, seed 701, the 1,000-step budget, the clean/sampled mixture, width sampling and validation selection with the main recipe; only the soft-target weight changes from one half to zero. Selection chooses step 250 for hard-only and step 1,000 for hard-plus-soft, so the training budget and selection rule are controlled but the selected checkpoints differ in the number of training steps. The hard-plus-soft objective lowers sequence PPL by 14.67, with a paired example-bootstrap interval of 6.15 to 25.76 lower, and changes corpus PPL less. All 256 outputs can be scored. This supports the soft-target term for one run rather than at seed level.

B Earlier ROCStories Protocol and Baselines

This section records an earlier evaluation protocol, retained for the baselines it contains. It is distinct from the ROCStories runs in Tables 1 and 2. Pretrained Cosmos uses three generation seeds, 201, 202 and 203, with 512 samples each: sequence PPL averages 41.27 over a range of 41.07 to 41.42. A fixed prefix-consistent checkpoint evaluated under three generation seeds gives PPL 47.49 at full length and 47.65 under predicted lengths, the latter saving 55.89% of positions. The historical underprediction diagnostic is 4.82%, computed with an EOS-index rather than EOS-inclusive convention, so it is not combined with the miss rates of the main study. Although seed labels match, head fitting and other RNG-consuming operations differ, so these pretrained and prefix-consistent samples are not asserted to be paired.

An autoregressive BART-base decoder adapted for 4,000 steps on 12,000 clean ROCStories examples gives PPL 28.72, ranging from 28.27 to 29.14 across three adaptation seeds with 512 generated outputs per run, at a mean output length of about 52.3 tokens. This baseline is materially stronger in GPT-2 perplexity and differs from the parallel decompressor in both architecture and training; these experiments do not identify the cause of the gap.

C Ordering Granularity

Replacing the total order $j \leq i$ with the group order $\lfloor j/B \rfloor \leq \lfloor i/B \rfloor$ allows positions within a group of B to attend to each other; $B{=}1$ recovers the token-level order and $B \to \infty$ the full-attention decompressor. At group-aligned lengths the proof of Proposition 1 still applies, because every position in a group sees the same set of earlier groups however many groups follow. The grouping is a property of the attention mask alone, so all positions are still computed in one parallel pass and B costs granularity rather than parallelism. The group-ordered students of both domains use the same sampled-latent sets, starting checkpoint, teacher, 1,000-step budget and three fine-tuning seeds as their token-ordered counterparts. Table 6 sweeps B on both domains. Quality rises monotonically, and

Table 6: Ordering granularity trades length granularity against quality, not parallelism. B is the size of the position groups the partial order is defined over: $B{=}1$ is a total order on output positions and $B \to \infty$ is the full-attention decompressor, which has no prefix property. Perplexities are means and [min, max] over three adaptation seeds sharing the starting checkpoint, latent sets and training budget within a domain. "Gap closed" is the fraction of the $B{=}1$-to-full-attention sequence-perplexity gap recovered. "Saved" is the share of the 128-position length left unevaluated when the length may stop only on a multiple of B, computed from the deployment run's required lengths under an oracle length choice so that it isolates granularity from head error. Every row is evaluated in a single parallel pass.

Group size B	Sequence PPL ↓	Gap closed	Saved (%)
ROCStories			
1	37.43 [37.40, 37.48]	n/a	57.7
8	36.17 [36.06, 36.28]	81%	54.9
16	35.95 [35.78, 36.09]	95%	51.9
32	35.79 [35.62, 35.94]	105%	45.7
∞ (full attention)	35.87	100%	0.0
CNN/DailyMail			
1	238.07 [234.63, 241.16]	n/a	49.0
8	217.13 [216.04, 218.16]	58%	46.4
16	211.75 [210.62, 213.69]	73%	43.5
32	210.19 [209.26, 211.02]	77%	36.0
∞ (full attention)	202.07	100%	0.0

the gap closes further on ROCStories: 58%, 73% and 77% at $B{=}8$, 16 and 32 on CNN/DailyMail, against 81%, 95% and 105% on ROCStories, where the $B{=}32$ students reach the full-attention decompressor's perplexity, 35.79 against 35.87 with a seed range of 35.62 to 35.94. The cost is granularity, since a decompression may stop only at a multiple of B: oracle-allocated savings fall from 49.0% at $B{=}1$ to 43.5% and 36.0% on CNN/DailyMail, and from 57.7% to 51.9% and 45.7% on ROCStories. In both domains, $B{=}16$ offers the best trade-off: it recovers nearly three quarters of the gap on CNN/DailyMail and 95% on ROCStories for 5.5 and 5.8 points of saving, where $B{=}32$ spends a further 7.5 and 6.2 points for four and ten more. Adjacent repetition follows the same pattern: 6.04%, 4.98%, 4.93% and 4.89% across the four group sizes on CNN/DailyMail, against 4.47% for the pretrained decompressor, and 1.91%, 1.65%, 1.55% and 1.56% on ROCStories, against 1.48%. We report this as a characterized trade-off, not a recommended operating point.

D Timing of Length Policies

Table 7 times the decompressor used for inference under six length policies per domain, including denormalization, head inference, token selection, EOS checks and cache operations, but not diffusion or latent transfer. Fixed initial widths of 32 and 64 trigger frequent extension and are not fixed-increment policies. Fixed 96 is faster than the learned policy on ROCStories.

A separate comparison on CNN/DailyMail controls the sources of noise that Table 7 leaves open. It reuses the seed-701 decompressors and the same 256 test latents, shares layer-input caching, attention, token selection and vocabulary projection across all policies, and times the first 64 test examples with five repetitions, randomized policy order within each example and repetition, and warmup for every policy, on an A100-SXM-64GB. A paired bootstrap averages repetitions within each example before resampling examples 10,000 times. Learned recovery reduces latency by 6.23% relative to a calibrated constant policy (95% interval 0.86% to 11.23%), but its 3.37% reduction relative to full-length decompression has interval -0.78% to 6.30%. Peak allocated memory is 3.032 and 3.037 GB for the full and learned policies. These intervals are conditional on the checkpoint and hardware and do not reflect fine-tuning-seed uncertainty.

At batch size one these measurements cannot resolve an advantage for learned recovery over full-length decompression: a single decompression is latency-bound, and at $L = 128$ it costs about the same whatever its length. Batched serving is where positions saved become time.

Table 7: Measured decompressor execution on an A100-SXM-64GB, fp32, batch size one: first 32 test latents, three repetitions. Time includes denormalization, the head when used, token selection, EOS checks, and cache operations; it excludes diffusion. Memory is incremental peak allocated GPU memory above the resident-model baseline, in decimal MB. Fixed policies use the same geometric recovery. Matches are to the full-length output on this 32-item subset.

Policy	Mean ms	Peak MB	Calls	Text match
ROCStories				
Full length	11.71	15.7	1.00	32/32
Learned, no recovery	10.72	12.7	1.00	32/32
Learned + recovery	11.11	21.1	1.03	32/32
Fixed 32 + recovery	30.68	15.1	2.81	32/32
Fixed 64 + recovery	12.94	19.3	1.22	32/32
Fixed 96 + recovery	10.39	15.4	1.00	32/32
CNN/DailyMail				
Full length	11.55	15.7	1.00	32/32
Learned, no recovery	10.55	20.4	1.00	31/32
Learned + recovery	10.88	27.3	1.03	32/32
Fixed 32 + recovery	31.96	16.5	3.00	32/32
Fixed 64 + recovery	16.30	18.5	1.56	32/32
Fixed 96 + recovery	11.64	24.4	1.12	32/32
ELI5				
Full length	11.23	15.7	1.00	32/32
Learned, no recovery	10.32	20.4	1.00	31/32
Learned + recovery	10.65	27.3	1.03	32/32
Fixed 32 + recovery	38.41	16.5	3.66	32/32
Fixed 64 + recovery	22.40	18.8	2.16	32/32
Fixed 96 + recovery	15.20	23.9	1.47	32/32

Batched serving. Table 8 serves the 256 saved test latents per domain in batches of 16 and 64, replaying the initial lengths chosen by the length head of Table 2. Requests are sorted by predicted length before batching, as a serving system would do, and each batch is decompressed to its longest prediction and extended from cached states until every member has an end-of-sequence token. Decompression is 1.2–1.8$\times$ faster than full-length decompression across domains, in proportion to the positions saved, and all 1,536 outputs are identical to the full-length outputs in fp32. Times are medians of five passes over all 256 requests and include token selection, end-of-sequence checks and trimming; the baseline decompresses the same batches at the full length with the decompressor's own forward pass. The measurement covers decompression only: at the default 200-step sampler, diffusion still dominates the generation cost (Section 5.5).

Table 8: Batched decompression time on the 256 test latents per domain, with batch size B. Positions saved are lower than in Table 2 because each batch is decompressed to its longest predicted length. All outputs are identical to the full-length outputs.

Domain	B	PACED (ms)	Full (ms)	Speedup	Saved (%)	Identical
ROCStories	16	322	555	1.72$\times$	52.5	256/256
	64	257	461	1.79$\times$	49.3	256/256
CNN/DailyMail	16	371	559	1.51$\times$	40.9	256/256
	64	313	462	1.48$\times$	34.0	256/256
ELI5	16	476	557	1.17$\times$	24.4	256/256
	64	377	461	1.22$\times$	18.4	256/256

E Sampler-Step Sweep and Calibration Transfer

This study varies only the number of sampling steps. The autoencoder, diffusion weights, decompressor and head architecture are those of the main paired study, and the fine-tuned decompressor used for inference is loaded unchanged. Latents are freshly sampled at each step count, so these sets

Table 9: Measured sampler-step sweep on the deployed prefix-consistent decompressor, batch size one. Diffusion and decompression times are per request, and share is decompression's fraction of the two. $\bar{R}$ is the mean required length. Initial miss is $\Pr[N_0 < R(z)]$ for the head calibrated once at the reference step count (transfer) and for a head refit at that step count (refit). Timings are in fp32, the deployed precision. Preserved counts test items whose recovered text equals the full-length text under the transfer head, from repeating the sweep in fp64. Fewer steps come from the same model, not from a distilled few-step sampler; Table 10 reports what that costs in text quality.

Steps	Diffusion (ms)	Decompr. (ms)	Share (%)	$\bar{R}$	Miss transfer / refit (%)	Preserved
ROCStories						
200	1932	11.61	0.6	53.7	4.7 / 3.9	256/256
100	971	11.53	1.2	54.1	10.2 / 2.7	256/256
50	488	11.47	2.3	57.1	21.1 / 4.7	256/256
20	200	11.48	5.4	59.7	87.5 / 5.5	256/256
8	83	11.50	12.2	63.7	99.2 / 6.2	256/256
4	44	11.44	20.5	65.5	100.0 / 4.7	256/256
2	24	11.27	31.7	38.4	0.0 / 6.6	256/256
1	15	11.36	43.2	40.3	0.0 / 9.4	256/256
CNN/DailyMail						
200	1912	11.44	0.6	66.3	2.7 / 2.7	256/256
100	964	11.38	1.2	69.8	5.5 / 7.0	256/256
50	480	11.22	2.3	73.5	5.1 / 2.0	256/256
20	196	11.21	5.4	79.0	26.2 / 2.7	256/256
8	81	11.25	12.1	79.8	90.6 / 5.1	256/256
4	44	11.39	20.6	59.9	84.0 / 3.1	256/256
2	24	11.31	31.6	75.9	0.0 / 8.6	256/256
1	15	11.23	42.8	84.5	0.0 / 8.6	256/256
ELI5						
200	1926	11.47	0.6	93.8	2.7 / 5.5	256/256
100	947	11.35	1.2	93.5	7.0 / 3.5	256/256
50	471	11.11	2.3	98.1	10.2 / 1.2	256/256
20	192	11.03	5.4	106.8	71.5 / 7.0	256/256
8	80	11.05	12.2	115.7	98.4 / 5.9	256/256
4	43	11.07	20.6	109.4	99.2 / 5.9	256/256
2	24	11.04	31.6	69.1	0.0 / 7.4	256/256
1	15	11.03	42.4	70.1	0.0 / 5.5	256/256

Table 10: External GPT-2 mean sequence perplexity of the deployed decompressor's output as the number of sampling steps is reduced. Reducing the step count of a model trained for 200 steps degrades the latents sharply; these numbers are why we treat the few-step regime as the setting a distilled sampler would create, and use the sweep to instrument cost structure and calibration rather than to claim few-step generation quality.

Domain	Sampler steps							
	200	100	50	20	8	4	2	1
ROCStories	38.6	46.6	79.3	252.7	957.0	1159.0	373.8	643.3
CNN/DailyMail	241.2	288.6	470.4	1219.3	2364.4	2223.3	363.5	305.2
ELI5	141.1	184.5	275.5	641.2	1140.7	743.7	586.1	660.7

are not the saved paired test latent sets of Section 4.4, and the head is refit in this study rather than reloaded from the main inference run. Absolute miss rates at 200 steps therefore differ slightly from Table 2 while reproducing its qualitative behavior, including complete output preservation.

Reducing the step count of a model trained at 200 steps is not the same as distilling a few-step sampler, and we do not present it as one. It is a controlled way of moving the latent distribution that the decompressor and head are asked to consume, which is the perturbation a faster sampler would introduce. Table 10 records the text-quality cost so that the cost-structure and calibration results are read in the right light.

Reading Table 10. Perplexity there is not monotone in the step count, and the non-monotonicity is an artifact rather than a recovery. At one and two steps the outputs collapse to short generic text, the mean required length on ROCStories falling from 65.5 positions at four steps to 38.4 at two and on ELI5 from 109.4 to 69.1, and short generic strings obtain low GPT-2 perplexity. The low perplexities in the last two columns therefore indicate degeneration, which is why we use this sweep to study cost structure and calibration rather than quality.

Two features of Table 11 are worth drawing out. Recalibration helps most where it is needed: at step counts where the reference margin has collapsed, a few dozen labeled latents recover essentially the oracle margin's miss rate, while at the reference sampler itself recalibration is a small and sometimes slightly negative adjustment, as expected. And having any fresh calibration matters far more than how much of it: k=16 already returns every configuration in which the calibrated head fails to under 8% miss, and k=64 comes within 1.6 points of the oracle everywhere. Output preservation is not re-measured in this table; Table 9 establishes it for these configurations.

F Numerical Precision of Recovery

Propositions 1 and 2 hold in exact arithmetic. On hardware, two executions that are equal mathematically can differ in the last bits when they run with different tensor shapes, because the attention and matrix-multiply kernels they select reduce in different orders; at a near-tie this can decide an argmax. This appendix reports the preservation measurements.

Sampler-step sweep. With the decompressor in fp64, recovery reproduces the full-length text on all 12,288 items across the 48 (domain, step count, head) configurations of Appendix E, including every configuration in which the transferred head misses on all or nearly all items. This is the result reported in Section 5.5 and Table 9.

Inference configuration. In fp32, the precision used for inference, recovery preserves every output at the default 200-step sampler in all three domains, and all 6,912 prefix comparisons of Section 5.2 agree exactly.

Recovery from every initial length. Two fp32 audits test the recovery mechanism directly. The first re-runs the recovery policy used for inference from every admissible initial length: for 48 freshly sampled ELI5 latents at 200, 100 and 1 steps, decompressing positions 1 through N_0 and growing geometrically to EOS or L for every $N_0 \in [2, 128]$ gives 18,288 recovery sequences, all of which agree with the full-length decompression at every position. The second fixes the growth schedule and compares lengths 32, 64 and 96 against the full-length decompression for 256 latents at five step counts on ELI5 and ROCStories: all 7,680 comparisons agree.

Table 11: Margin-only recalibration. The length head is fit once at the reference sampler and never refit; only its scalar safety margin is recomputed from k freshly sampled latents, each labelled by one full-length decompression. Entries are initial miss rates $\Pr[N_0 < R(z)]$ on an independent evaluation set, averaged over 20 draws of the calibration subset. "None" keeps the reference margin; "oracle" uses the full calibration pool. Because recovery is exact, the realized length is a deterministic function of $(N_0, R(z))$, so these policies are evaluated arithmetically from the same labelled pairs.

Steps	$\bar{R}$	None	k=16	k=32	k=64	k=128	k=256	Oracle
ROCStories								
200	54.0	2.3	3.6	4.2	4.0	3.2	3.3	2.3
50	57.0	17.6	4.0	3.9	3.4	4.3	3.3	3.3
20	60.0	88.3	5.2	5.2	6.1	4.8	4.6	4.5
8	64.6	99.2	5.3	5.2	4.1	4.2	4.1	4.3
4	65.0	99.8	4.2	3.8	2.4	2.8	3.0	2.9
1	41.6	0.0	8.1	9.3	8.4	7.4	6.6	6.6
CNN/DailyMail								
200	67.1	6.8	6.1	9.0	8.6	7.6	7.1	6.8
50	71.0	6.2	2.7	4.3	3.5	3.9	3.4	3.5
20	78.6	32.8	3.0	5.4	4.7	3.7	4.2	4.3
8	82.9	91.4	4.2	5.5	5.4	5.5	5.2	4.9
4	60.6	86.7	4.6	7.3	5.6	5.2	5.2	4.9
1	85.1	0.0	4.7	7.3	7.2	7.1	5.6	5.7
ELI5								
200	90.5	5.1	6.2	5.3	5.7	4.8	5.2	5.1
50	98.9	14.3	5.1	4.1	3.5	3.6	2.9	2.7
20	106.4	77.7	4.6	6.6	5.0	3.5	3.5	3.5
8	116.5	98.8	7.2	4.4	3.2	3.0	2.1	2.1
4	111.0	99.6	3.1	3.5	2.1	2.4	2.0	2.1
1	70.3	0.0	5.1	7.6	7.0	5.5	5.0	4.9