DIAL: ONE DIFFUSION LANGUAGE MODEL, MULTIPLE INFERENCE-TIME COMPUTE BUDGETS

Anonymous authors
Paper under double-blind review

ABSTRACT

Block diffusion language models trade generation quality for inference cost by varying the number of tokens committed per denoising step, yet this block size is typically fixed during training, requiring separate specialized models for different deployment settings. We present DIAL, a training framework for *block-size-elastic* diffusion language models in which the inference block size is a runtime choice shared by a single parameter set. The core mechanism, mixed block-size training, samples block sizes throughout training, exposing one network to a range of denoising granularities from autoregressive-like factorization to highly parallel generation. To preserve performance across this range, DIAL combines per-block noise conditioning with a zero-initialized recurrent draft-refinement loop, a two-stream cached parameterization, and argmax-agreement early stopping. Experiments show that mixed block-size training substantially reduces the degradation that otherwise occurs when a model is evaluated away from its training block size, while a single DIAL model achieves competitive perplexity across all tested inference granularities. By treating block size as an inference-time control, DIAL exposes multiple quality–compute operating points from one trained model, allowing deployment to adapt generation cost to the available compute budget without retraining or maintaining separate block-size specialists.

1 INTRODUCTION

The central tension: rigid training vs. flexible serving. Modern artificial intelligence applications operate under vastly different computational demands. An interactive conversational assistant requires real-time, low-latency generation and can tolerate slight trade-offs in output quality, whereas an offline analytics pipeline prioritizes maximum accuracy. While flexible compute scaling has been explored in continuous domains like computer vision and multimodal research (Cai et al., 2020; Yu et al., 2019; Devvrit et al., 2023; Cai et al., 2025; Graves, 2016; Schuster et al., 2022; Elbayad et al., 2020), conventional autoregressive and discrete diffusion language models remain bound to a static paradigm: compute–quality trade-offs are hardcoded during pretraining. Serving diverse latency–accuracy requirements currently forces practitioners to train, host, and maintain separate specialized model instances for every operating budget.

Block diffusion and the barrier of fixed decoding granularity. Autoregressive transformers (Vaswani et al., 2017; Radford et al., 2019; Brown et al., 2020; Touvron et al., 2023) deliver strong sample quality by evaluating sequence probabilities strictly token-by-token, but their serial nature creates a major latency bottleneck. Discrete diffusion models (Austin et al., 2021; Campbell et al., 2022; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2024; Nie et al., 2025a;b; Gat et al., 2024) generate multiple sequence positions in parallel, trading strict conditional dependencies for decoding speed. Block diffusion (Arriola et al., 2025) bridges these two settings by grouping sequence positions into local blocks of size L'. However, existing block diffusion models fix L' during pretraining. Consequently, learned positional representations, attention statistics, and noise schedule interactions deeply overfit to that single block scale. When evaluated at a block size other than its training configuration, a phenomenon we term *off-design evaluation*, the model suffers catastrophic quality degradation, creating an off-design perplexity gap.

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.

The core challenge of block-size elasticity. Developing a single language model that dynamically adjusts its block size at runtime is non-trivial. Discrete text tokens lack the continuous smoothness of pixel spaces, meaning that varying L' fundamentally alters the conditional dependency structure: small blocks require strict sequential conditioning, whereas large blocks require joint parallel denoising over unobserved tokens. Training a single model across varying block sizes induces severe *gradient interference*, where updates optimized for serial generation degrade parallel prediction, and *statistical mismatch* in normalization and noise conditioning across different block scales.

We address this fundamental challenge by asking: *Can a single set of model parameters learn a unified generative space such that decoding granularity becomes a dynamic runtime choice, exposing a smooth quality–compute frontier without retraining or increasing parameter count?*

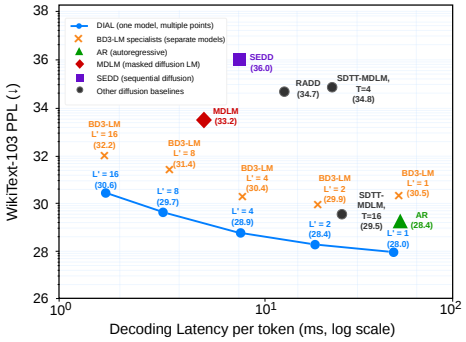

Figure 1: **DIAL quality–compute frontier.** Zero-shot WikiText-103 perplexity vs. decoding throughput. Points show the same DIAL model evaluated across block sizes $L' \in \{1, 2, 4, 8, 16\}$. Orange crosses denote fixed-block BD3-LM specialists (Arriola et al., 2025). Additional markers show conventional AR (Radford et al., 2019) and diffusion baselines including MDLM (Sahoo et al., 2024), SEDD (Lou et al., 2024), and RADD (Ou et al., 2024).

Our framework: elastic generation via joint optimization. In this work, we introduce DIAL, a training and inference framework that achieves block-size elasticity within a single parameter set (Figure 1). The foundation of our approach is **Mixed Block-Size Training (MBT)**, which stochastically resamples the decoding block size L' across optimization steps, exposing one network to the full spectrum of generative granularities.

To overcome gradient interference, statistical mismatch, and decoding efficiency bottlenecks across block choices, DIAL incorporates four targeted technical components:

1. **Per-Block Noise Conditioning (M1, Novel):** Addresses statistical mismatch by computing per-block noise statistics $\bar{\sigma}_b$, providing normalized variance signals that allow the shared attention layers to distinguish local noise levels across varying block scales.
2. **Recurrent Draft Refinement (M2, Novel):** Resolves gradient interference during parallel denoising by feeding zero-initialized self-conditioning feedback across reverse diffusion steps, stabilizing multi-token joint predictions without disrupting autoregressive capabilities.
3. **Two-Stream Parameterization (M3, Adapted):** Enhances decoding efficiency by adapting cached two-stream attention mechanics to strictly separate clean historical context from noisy parallel target blocks, preventing redundant re-computations during generation.
4. **Argmax-Agreement Early Stopping (M4, Adapted):** Optimizes serving latency by applying early convergence checks on target token probabilities, dynamically halting inner sampling steps once output token predictions stabilize.

Key insights and findings. Through systematic empirical evaluations on zero-shot language modeling benchmarks, we demonstrate three primary findings:

- **Elasticity is trainable in a single run:** Mixed block-size training effectively eliminates off-design quality degradation, enabling a single checkpoint to smoothly traverse block sizes $L' \in \{1, 2, 4, 8, 16\}$ while matching or exceeding specialized baselines.
- **One elastic model matches suites of specialists:** Across the entire operating spectrum, a single DIAL parameter set achieves a superior overall quality–compute frontier compared to an ensemble of individually pretrained fixed-block specialist models.
- **Architectural enhancements preserve generation quality:** We disentangle raw sequence parallel arithmetic from architectural gains, demonstrating that while throughput scaling follows block partitioning geometry, our proposed conditioning and refinement modules (M1–M4) are essential to preventing output quality collapse at extreme block settings.

2 BACKGROUND AND MOTIVATION

Absorbing-state discrete diffusion. Let $\boldsymbol{x}_0 \in \mathcal{V}^N$ be a token sequence over a vocabulary $\mathcal{V}$ with a distinguished `[MASK]`. Absorbing-state diffusion (Austin et al., 2021; Sahoo et al., 2024; Shi et al., 2024) independently replaces each token by `[MASK]` with probability $1-\alpha_t$, where α_t decreases from 1 to 0 over $t \in [0,1]$:

$$q(\boldsymbol{x}_t \mid \boldsymbol{x}_0) \;=\; \prod_{i=1}^{N} \big[\alpha_t\, \delta_{x_t^i, x_0^i} + (1-\alpha_t)\, \delta_{x_t^i, [\mathrm{MASK}]}\big]. \tag{1}$$

A network f_θ predicts the clean tokens at masked positions; training minimises the weighted denoising (negative ELBO) objective

$$\mathcal{L}(\theta) = \mathbb{E}_{t,\,\boldsymbol{x}_0,\,\boldsymbol{x}_t}\left[w(t) \sum_{i:\, x_t^i = [\mathrm{MASK}]} -\log f_\theta(x_0^i \mid \boldsymbol{x}_t, t) \right], \qquad w(t) \;=\; \frac{-\dot{\alpha}_t}{1-\alpha_t}, \tag{2}$$

whose minus sign makes $w(t) > 0$ since α_t decreases, so the objective is non-negative as a negative ELBO must be. At inference one starts fully masked and descends a grid $t_0 > \ldots > t_T$, resampling at each step only the positions still masked and freezing the rest,

$$x_{t_{j+1}}^i \;\sim\; \eta_j\, \delta_{[\mathrm{MASK}]} \;+\; (1-\eta_j)\, f_\theta(\cdot \mid \boldsymbol{x}_{t_j}), \qquad \eta_j \;=\; \frac{1-\alpha_{t_{j+1}}}{1-\alpha_{t_j}}, \tag{3}$$

η_j being the probability a masked position stays masked. Forwards dominate cost; we separate *scheduled* $\mathrm{NFE}_{\mathrm{sched}} = (N/L')T$ from *realised* $\mathrm{NFE}_{\mathrm{real}} = \sum_b T_b$, neither being seconds.

What a forward pass costs, and what the dial therefore moves. A forward pass is evaluated over all N positions, but while block b is denoised only its L' positions can affect the output: the prefix is read from cache and the tail is masked out. The useful fraction of a pass is L'/N, so NFE counts passes and not work, and per-pass cost is the same at every block size; our timings are flat to 3% across L' (App. E.9). We execute $N \cdot \mathrm{NFE}_{\mathrm{real}}$ position-forwards, so executed arithmetic and wall clock track chain depth; a position-sparse implementation would pay $L' \cdot \mathrm{NFE}_{\mathrm{real}}$ instead, and the two differ by the factor N/L'.

This is where our cost claim fails, and we state the decomposition because it is not in our favour. Without stopping, matched-T depth is exactly T/L', dividing by 16 over $\mathcal{S}$, while the work term $L'(T/L')$ is *identically* T at every block size: the dial alone moves depth by arithmetic every block-diffusion model has, and moves work not at all. Our numbers are that arithmetic composed with M4, which reduces depth $2.96\times$ at $L'=1$ and $1.00\times$ at $L'=16$. So $16/2.96 = 5.4$ is the depth ratio we report, and 2.96 is the whole rise on the work axis. M4 needs no training, we did not enable it on the specialists, and since our perplexities are sampling-free it would change their depth and not their quality: at our own $L'=4$ ratio the BD3-LM specialist sits at depth 6.22 against DIAL's 6.22. We withdraw any efficiency claim for the dial itself.

Block diffusion and the block-size spectrum. Block diffusion (Arriola et al., 2025) applies Eq. 2 independently within each block $\boldsymbol{x}_0^{(b)}$ while conditioning on the clean past $\boldsymbol{x}_0^{(<b)}$. Within a block all L' positions are denoised under a shared latent and committed together. With $T \gg L'$ the reverse chain unmasks roughly one position per step, so what degrades as L' grows is not a hard conditional-independence assumption but the order-averaging slack of the any-order bound plus the joint commitment at the end; both vanish at $L'=1$. Generating N tokens takes $B = N/L'$ sequential block steps of at most T inner passes. BD3-LM freezes L', coupling a deployment decision to a training one; DIAL removes the coupling.

3 THE DIAL FRAMEWORK

3.1 WHAT THE FRAMEWORK IS

Each module repairs a specific failure the core introduces or leaves standing.

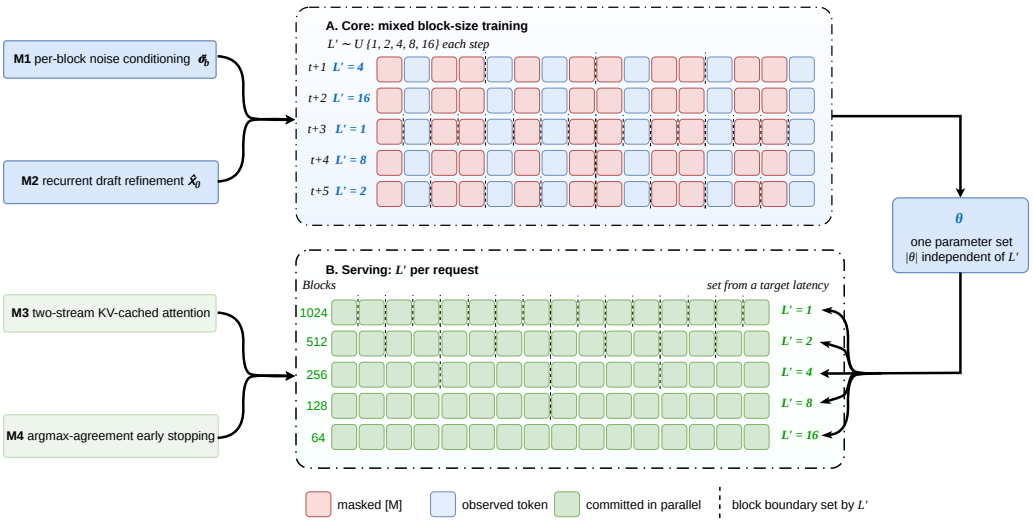

Figure 2: **The DIAL framework: one core mechanism, four modules, one parameter set. (A) Core.** Each optimisation step redraws L' uniformly from $\{1, 2, 4, 8, 16\}$ and swaps in the cached mask of that size. The five rows are five consecutive steps over the same corrupted sequence: masked positions ([M]) and observed tokens ( ) stay put and only the dashed block boundaries move; modules **M1** and **M2** act here. **(B) Serving.** The weights expose L' as a runtime argument and the boundaries set how many positions are committed jointly ( ); **M3** and **M4** act here. Every point in panel B runs the same θ.

One training step. Sample a mini-batch and a global $L' \sim \mathcal{U}(\mathcal{S})$ (core), truncate to a length divisible by L', draw t and build $\boldsymbol{x}_t$ by Eq. 1; compute per-block noise levels $\bar{\sigma}_b$ (M1); with probability p_{sc} take a no-grad draft forward and keep its argmax $\boldsymbol{x}_0^{\text{prev}}$ (M2); forward $f_\theta(\boldsymbol{x}_t, \{\bar{\sigma}_b\}, \boldsymbol{x}_0^{\text{prev}})$ under the cached L'-mask through two-stream attention (M3) and backpropagate Eq. 2 (App. C). L' enters only through the masks and the noise levels derived from them, never through a weight.

Serving a request, and what the last step does. The server passes the requested L' as an argument and generation proceeds block by block, left to right. Block b begins fully masked and runs at most T inner steps of Eq. 3, each conditioned on $\bar{\sigma}_b$ and on the previous step's draft (M2), stopping once the argmax repeats for K consecutive steps (M4). *Leaving the inner loop is what commits*: any position still masked takes the *argmax* of the final draft, so stopping early substitutes a mode for an ancestral draw and changes the generative distribution rather than only saving compute. All L' tokens then join the clean stream together, their cross-attention keys and values are written to the cache exactly once (M3), and block $b+1$ starts.

3.2 Core: Mixed Block-Size Training

Instead of fixing L' during pretraining, we treat the block size as a random variable resampled at each optimization step from a candidate set of granularities $\mathcal{S} \subset \{1, \ldots, N\}$. We optimize the expected loss across this support set:

$$\mathcal{L}_{\text{MBT}}(\theta) \;=\; \mathbb{E}_{L' \sim \mathcal{U}(\mathcal{S})}\big[\,\mathcal{L}_{L'}(\theta)\,\big], \qquad \mathcal{L}_{L'} \text{ the loss of Eq. 2 under the } L'\text{-block mask.} \tag{4}$$

This objective is optimized stochastically by sampling a single $L' \in \mathcal{S}$ per optimization step rather than evaluating all candidates jointly. Block sizes are synchronized across data-parallel workers, and attention masks are cached per (L', N) pair (App. B.1).

Crucially, MBT modifies only the sampling distribution over block masks (Fig. 2A), requiring zero additional parameters or structural changes to the core backbone; the minimal parameter additions in DIAL belong entirely to Sec. 3.3. Because every operating point evaluates the *same* network under a *different* masking scheme, exposing the model to diverse granularities throughout $\mathcal{S}$ forces the parameter set to learn a partition-invariant representation.

While $\mathcal{S}$ can be chosen arbitrarily to cover any target operational spectrum, a theoretical regret bound for the MBT minimizer remains valid for any discrete support set $\mathcal{S}$ (Prop. 3, App. A.4): the excess loss at any $L' \in \mathcal{S}$ is bounded by the sum of pairwise cross-block-size excesses, vanishing exactly when a fully partition-invariant optimum exists. We state this formulation in the Appendix because the empirical cross-block excesses we measure (Table 5) are already tighter than the theoretical upper bound.

3.3 Modules that keep the core trainable

These two modules exist because the core changes the learning problem, not because they improve a fixed-block model. Each is stated with the failure it repairs.

M1: per-block noise conditioning. BD3-LM passes $\sigma = \mathbf{0}$ to its adaptive layer norm, removing the diffusion time signal. This is harmless at a fixed L' but damaging under MBT: with the block size varying, the network cannot distinguish a noisy large block early in t from an almost-clean small block late in t, two situations with identical token statistics but different targets. We restore the signal with the per-block total noise $\bar{\sigma}_b = -\log \bar{\alpha}_b$, for $\bar{\alpha}_b$ the mean keep probability over block b, embedded sinusoidally through a two-layer MLP and used to drive adaLN modulation (Peebles & Xie, 2023; Perez et al., 2018). It costs under 0.05% of non-embedding parameters and buys stable MBT convergence and a 5.2% LAMBADA gain (Table 3).

M2: Recurrent Draft Refinement. Within a block the L' tokens are denoised jointly and must agree without an explicit ordering. M2 adds an additive, zero-initialised self-conditioning path: the argmax $\boldsymbol{x}_0^{\text{prev}}$ of the previous inner step is embedded and added to the noisy embedding before the next,

$$\tilde{\boldsymbol{h}} \;=\; \mathrm{Embed}(\boldsymbol{x}_t) \;+\; W_{\text{sc}}\,\mathrm{Embed}(\boldsymbol{x}_0^{\text{prev}}), \qquad W_{\text{sc}} \leftarrow \mathbf{0} \text{ at init}, \tag{5}$$

so each inner step refines rather than recomputes the block draft (Fig. 5 C). Zero initialisation leaves the loop inert early in training, and the expected training forward cost asymptotes at $1.5\times$ a non-self-conditioned step. The case for M2 is empirical: it is the most valuable single component in our ablation, its removal costing 12.5% on LAMBADA (Table 3).

3.4 Modules that keep every setting efficient

M3 flattens the per-step cost across L'; M4 reduces the number of steps taken.

M3: two-stream parameterisation. DIAL splits the computation in two. The length-N noisy stream carries all the depth: each layer applies block-diagonal self-attention over $\boldsymbol{x}_t$, then strictly block-causal cross-attention into the clean stream, then the feed-forward network. The clean stream carries none: it is a per-token embedding of the committed tokens, detached and *never self-attended*, where BD3-LM fuses both into one $2N$-long pass in which clean positions do attend to one another. Being position-wise, its per-layer keys and values are fixed while a block is denoised, so we write them once per decoded block (App. B.3). Two masked softmaxes over N noisy and N clean keys are *not* one softmax over their union, since each normalises over its own key set: DIAL is an architectural variant of BD3-LM and not a cheaper route to the same function, which is why the M3 row of Table 3 moves perplexity. Counting projections and the vocabulary head, the forward pass falls by $1.50\times$, an analytic figure we do not measure against BD3-LM directly (App. B.2).

M4: argmax-agreement early stopping. Most blocks converge well before the T-th inner step: the argmax stabilises early even when T is large, especially once RDR has committed earlier blocks. We stop the inner loop as soon as the argmax repeats for K consecutive steps,

$$\mathrm{stop}(t) \iff \bigwedge_{k=0}^{K-1} \left[\, \mathrm{argmax}\, p_\theta^{(t-k)} = \mathrm{argmax}\, p_\theta^{(t-k-1)} \right], \tag{6}$$

with $K = 2$ and a three-step warm-up, so a block costs at least five forwards. The overhead is two argmax comparisons on already-computed logits. M4 decouples realised from scheduled forwards, but it is not free: leaving the loop early commits the remaining masked positions at their argmax, changing the generative distribution, and no metric we report sees that change.

Table 1: **The dial: what one elastic model replaces, and what moving it costs.** *Quality:* zero-shot WikiText-103 perplexity (↓, MC-ELBO bound) for six models at 170M parameters and 524B OpenWebText tokens, at every combination of training and inference L'. Rows 1–5 are BD3-LM specialists lacking M1–M3; row 6 is one DIAL model read by changing only inference-time L'. **Bold** is best per column, last column the geometric mean. *Cost:* the same weights on one RTX A6000; forward counts are exact arithmetic and the $T = 100$ row is the matched-T comparison behind the 16×. Stopping rows use Eq. 6 at $T = 20$, batch 4 for $L' \in \{1, 2\}$ and $T = 50$, batch 16 otherwise, so they are not a clean sweep in L'; throughput is comparable only within the batch-16 group, marked * otherwise.

Trained L' \ Eval L'	1	2	4	8	16	**geo. mean**
Specialist $L' = 1$	30.5	33.8	39.4	47.1	56.3	40.4
Specialist $L' = 2$	31.2	29.9	32.6	37.4	44.8	34.8
Specialist $L' = 4$	33.1	31.0	30.4	32.5	37.6	32.8
Specialist $L' = 8$	35.8	33.4	31.9	31.4	33.6	33.2
Specialist $L' = 16$	38.2	35.7	33.8	32.9	32.2	34.5
DIAL (one model)	**28.0**	**28.4**	**28.9**	**29.7**	**30.6**	**29.1**
Forwards at $T = 100$, no stopping	102 400	51 200	25 600	12 800	6 400	–
Scheduled forwards	20 480	10 240	12 800	6 400	3 200	–
Realised forwards	6 930	4 094	6 354	5 120	3 200	–
NFE reduction	2.96×	2.50×	2.01×	1.25×	1.00×	–
Throughput (tok/s) ↑	9.1*	15.9*	10.4	13.1	**21.2**	–

What we can and cannot establish. Properties are proved in App. A and no empirical result rests on any of them. Each $\mathcal{L}_{L'}$ upper-bounds $-\log p_\theta^{(L')}(\boldsymbol{x}_0)$, so one parameter set induces the family the dial selects from (Prop. 1); at $L' = 1$ that bound is on an AR cross-entropy, tight only if the predictive ignores the noise level M1 supplies (Prop. 2). Every perplexity here is therefore loose by an uncharacterised amount and is not comparable to an exact AR likelihood. The rest are weaker than the modules they accompany: Prop. 3 is vacuous at our measured values, and Prop. 5's contraction hypothesis is contradicted by our own stopping times.

4 Experiments

We test the three claims of Sec. 1: whether one elastic model replaces a grid of specialists (Sec. 4.1), what elasticity costs downstream (Sec. 4.2), and which component earns which gain (Sec. 4.3).

Setup. We train DiT-style (Peebles & Xie, 2023) models with RoPE positions (Su et al., 2024), $N = 1024$ and the GPT-2 tokeniser on OpenWebText (Gokaslan et al., 2019), at 170M non-embedding parameters and 524B tokens, matched to the BD3-LM specialists (Arriola et al., 2025) we compare against, using AdamW (Kingma & Ba, 2015; Loshchilov & Hutter, 2019) with EMA (Polyak & Juditsky, 1992) and bf16 (Micikevicius et al., 2018). Position indices are global and independent of the partition, so only the mask varies with L'. Held-out likelihood is the MC-ELBO upper bound on zero-shot WikiText-103 (Merity et al., 2017), LAMBADA (Paperno et al., 2016) and Penn Treebank (Marcus et al., 1993); downstream accuracy uses `lm-evaluation-harness` protocols; efficiency is measured on one RTX A6000, and weights are selected by minimum OpenWebText validation loss. BD3-LM at $L' \in \{4, 8, 16\}$ is released weights re-scored by us; $L' \in \{1, 2\}$ are our own reproductions, none being released; other published rows are quoted, not re-scored (App. D.1).

4.1 Q1 and Q2: one model outperforms the grid, over a 16× scheduled cost range at matched T

Scoring all six models at all five inference block sizes separates how good a model is at its own block size from how gracefully it moves away. Every specialist is at or near its best on its diagonal and degrades monotonically with distance, severely at the ends, the $L' = 1$ specialist rising from 30.5 to 56.3 under a sixteen-token factorisation. Against the best specialist per column, off-diagonal empirical regret averages $+5.7$ PPL for the specialists and -1.8 for DIAL (App. E.1).

Table 2: **Downstream comparison under one shared protocol.** Zero-shot *lm-evaluation-harness* scores: LAMBADA perplexity (↓), BLIMP macro-accuracy over 67 minimal-pair phenomena and HellaSwag accuracy (↑). Every diffusion row is 170M non-embedding parameters at 524B OpenWebText tokens; BD3-LM and DIAL rows are re-scored by us under one pipeline, the SEDD row is quoted. **Bold** marks the best diffusion value per column. One DIAL parameter set supplies both DIAL rows, differing only in inference-time L'; each BD3-LM row is a separate run. The AR row is *context, not a matched baseline*, and its LAMBADA figure is exact continuation scoring rather than an MC-ELBO bound (App. E.6).

Model / L'	LAMBADA PPL ↓	BLIMP ↑	HellaSwag ↑	runs for the dial
BD3-LM bs4 (Arriola et al., 2025)	49.09	0.582	0.271	1 per block size
BD3-LM bs8	49.37	0.560	0.270	1 per block size
BD3-LM bs16	50.02	0.552	0.273	1 per block size
SEDD (Lou et al., 2024)	49.75	–	–	1, no dial
DIAL ($L'=4$)	**46.3**	**0.63**	**0.298**	**1** for all L'
DIAL ($L'=16$)	47.8	0.62	0.295	
Pythia-160M (Biderman et al., 2023)	103.5	0.717	0.283	AR, no dial

The elastic model is best in every column, including against the specialist trained at that exact block size (28.9 against 30.4 at $L'=4$), with a geometric mean 11.3% below the best specialist's. We do not read this as a benefit of elasticity: those specialists lack M1–M3, and against the one architecture-matched fixed-block model we have the sign reverses (Sec. 4.3), so the negative regret is an architecture effect measured on an elasticity axis. The curve is monotone in the deployment-relevant direction, costing 2.6 PPL from $L'=1$ to 16 (9.3%). At $L'=1$ DIAL reaches 28.0 against 28.41 for a matched 170M AR reference, but these are not comparable: ours is a Monte Carlo bound of uncharacterised slack and the AR figure exact (App. E.2).

The lower blocks of Table 1 hold cost ratios we keep apart. At matched T without stopping, $\mathrm{NFE}_{\mathrm{sched}} = (N/L')T$ divides by exactly 16 for 2.6 points of perplexity, which is arithmetic and not a measurement; our serving runs, which vary T, schedule 6.4× and realise 2.2×. Sec. 2 decomposes what of this is ours: none of it. Memory is flat because activation tensors are sized by N at every L' (Apps. B.2, E.15).

M4 attacks the expensive end. At $L'=4$, $T=50$, argmax agreement commits blocks after 49.6% of the scheduled forwards, reaching 2.96× at $L'=1$ and falling to nothing at $L'=16$, where sixteen tokens rarely reach a stable joint argmax within 50 steps: the saving runs opposite to the dial, though the small-L' figures were taken at batch 4 and the rule is batch-coupled, so part of that trend is the batch size. Throughput rises 10.4 to 21.2 tok/s as L' goes $4 \to 16$, a factor of 2.0 rather than the 4 the block ratio predicts.

4.2 Q3: WHAT THE DIAL COSTS ON DOWNSTREAM TASKS

At $L'=4$ DIAL leads the strongest BD3-LM specialist by 5.7% on LAMBADA, 4.8 BLIMP points and 2.5 of HellaSwag while serving five operating points, but those specialists lack M1–M3, and across the full battery only the BLIMP and HellaSwag margins survive their binomial standard errors: two wins, five ties and one loss (App. E.6). DIAL also trails Pythia-160M on three of the four comparably scored tasks, despite more non-embedding parameters and more training tokens.

Between the two DIAL rows, HellaSwag changes by 0.003 against a standard error of 0.005, not resolvable, and BLIMP by 0.01, our reporting granularity there; LAMBADA moves 3.2%. Candidate *ranking* degrades more slowly than absolute likelihood but does degrade, margins shrinking about 5% even where accuracy holds (App. E.17): the cheap end is usable, not free.

4.3 Q4: WHICH MECHANISM EARNS WHICH GAIN

Removing M2 is the most damaging single change on all three corpora (12.5% on LAMBADA, 8.7% on PTB, 4.8% on WikiText-103), largest out of domain, which is what a module for coordinating within-block predictions should do. The ordering of the rest is not stable across corpora and we have no out-of-domain error bars, so we do not rank them.

Table 3: **Leave-one-out ablation.** Each row removes one component from full DIAL and retrains at the same configuration (170M non-embedding parameters, 524B OpenWebText tokens), evaluated at $L' = 4$; columns are perplexity (↓) under one MC-ELBO pipeline. The last two rows are the BD3-LM specialist at fixed $L' = 4$, matched in scale, corpus and budget but not architecture. M4 is absent, being a sampling-time rule that cannot change a likelihood.

Model	LAMBADA ↓	PTB ↓	WT-103 ↓
DIAL (full)	**46.3**	**84.1**	28.9
− Recurrent Draft Refinement	52.1	91.4	30.3
− two-stream attention	49.8	88.2	29.6
− per-block time conditioning	48.7	86.9	29.4
− MBT (fixed $L' = 4$)	50.4	88.6	**28.4**
BD3-LM bs4 (Arriola et al., 2025)	49.09	89.43	30.40
Δ DIAL vs. BD3-LM	−5.7%	−6.0%	−4.9%

The −MBT row is the informative one, and the only architecture-matched fixed-block model here. Specialising to $L' = 4$ while keeping the other three modules is the only change that *improves* a number, WikiText-103 falling $28.9 \to 28.4$ for five usable operating points. It costs elsewhere, $46.3 \to 50.4$ on LAMBADA and $84.1 \to 88.6$ on PTB, so elasticity is a price only in domain and MBT regularises out of it (App. E.13). All rows are one seed.

5 Related work

Absorbing-state and score-based DLMs (Austin et al., 2021; Lou et al., 2024; Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2024; Gat et al., 2024; Deschenaux & Gulcehre, 2025; Nie et al., 2025b) established parallel text denoising; block diffusion (Arriola et al., 2025) and semi-autoregressive decoders (Han et al., 2023; Savinov et al., 2022; Ghazvininejad et al., 2019; Chang et al., 2022) interpolate it with AR generation. Such a schedule is already selectable at inference, so the granularity knob is not new; what BD3-LM fixes at training time is the block size the weights are *good* at. Concurrent work (Jain, 2026) also randomises block configuration, targeting train/inference mismatch at one operating point rather than a deployment control (Table 7).

Once-for-All (Cai et al., 2020), slimmable networks (Yu et al., 2019), LayerDrop (Fan et al., 2020), Matryoshka representations (Kusupati et al., 2022), MatFormer (Devvrit et al., 2023), Flextron (Cai et al., 2024a) and LLaMaFlex (Cai et al., 2025) train one network supporting many sub-networks. MBT is that literature's recipe applied to L', which unlike a sub-network leaves $|\theta|$ and the computation graph intact. M4 moves an adaptive-compute exit criterion (Graves, 2016; Schuster et al., 2022; Elbayad et al., 2020) into the diffusion inner loop; speculative decoding (Leviathan et al., 2023; Cai et al., 2024b; Stern et al., 2018) amortises AR latency with auxiliary models. Closest on the serving side, Fast-dLLM (Wu et al., 2026) obtains block-wise KV caching and confidence-thresholded parallel decoding with *no* training, which is most of what M3 and M4 do; our contribution is the training recipe, not the serving mechanics. Two-stream attention adapts XLNet's (Yang et al., 2019) construction and self-conditioning (Chen et al., 2023; Gu et al., 2018; Lee et al., 2018) inspires M2; App. E.2 expands.

6 Discussion, limitations and conclusion

A fixed-block model keeps a 0.5 PPL edge at its trained block size, barely above the 0.2 we resolve; DIAL pays for itself from the second operating point on (App. D.3). Six limitations bound the claims (App. F): single-seed numbers at 170M; resting on one architecture-matched cell; nothing measuring generated text; minutes per batch-one request. On cost we are dominated from both sides, by a matched AR model at 28.41 for one position-forward per token and by SDTT–MDLM at 29.5 for 16 whole-sequence evaluations (Table 6); and T is a compute control every masked diffusion model already exposes on one parameter set, which we do not test against.

REFERENCES

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Cody Blakeney, Leandro von Werra, et al. SmolLM2: When Smol goes big – data-centric training of a small language model. *arXiv preprint arXiv:2502.02737*, 2025. URL https://arxiv.org/abs/2502.02737.

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block discrete denoising diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2503.09573.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021. URL https://arxiv.org/abs/2107.03006.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning (ICML)*, 2023. URL https://arxiv.org/abs/2304.01373.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. PIQA: Reasoning about physical commonsense in natural language. In *AAAI Conference on Artificial Intelligence (AAAI)*, 2020. URL https://arxiv.org/abs/1911.11641.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in Neural Information Processing Systems (NeurIPS)*, 2020. URL https://arxiv.org/abs/2005.14165.

Han Cai, Chuang Gan, Tianzhe Wang, Zhekai Zhang, and Song Han. Once-for-all: Train one network and specialize it for efficient deployment. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://arxiv.org/abs/1908.09791.

Ruisi Cai, Saurav Muralidharan, Greg Heinrich, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. Flextron: Many-in-one flexible large language model. In *International Conference on Machine Learning (ICML)*, 2024a. URL https://arxiv.org/abs/2406.10260.

Ruisi Cai, Saurav Muralidharan, Hongxu Yin, Zhangyang Wang, Jan Kautz, and Pavlo Molchanov. LLaMaFlex: Many-in-one LLMs via generalized pruning and weight sharing. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=AyC4uxx2HW.

Tianle Cai, Yuhong Li, Zhengyang Geng, Hongwu Peng, Jason D. Lee, Deming Chen, and Tri Dao. Medusa: Simple LLM inference acceleration framework with multiple decoding heads. In *International Conference on Machine Learning (ICML)*, 2024b. URL https://arxiv.org/abs/2401.10774.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2205.14987.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T. Freeman. MaskGIT: Masked generative image transformer. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022. URL https://arxiv.org/abs/2202.04200.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *International Conference on Learning Representations (ICLR)*, 2023. URL https://arxiv.org/abs/2208.04202.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2019. URL https://arxiv.org/abs/1905.10044.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Tri Dao. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*, 2024. URL https://arxiv.org/abs/2307.08691.

Justin Deschenaux and Caglar Gulcehre. Beyond autoregression: Fast LLMs via self-distillation through time. In *International Conference on Learning Representations (ICLR)*, 2025. URL https://arxiv.org/abs/2410.21035.

Devvrit, Sneha Kudugunta, Aditya Kusupati, Tim Dettmers, Kaifeng Chen, Inderjit Dhillon, Yulia Tsvetkov, Hannaneh Hajishirzi, Sham Kakade, Ali Farhadi, and Prateek Jain. MatFormer: Nested transformer for elastic inference. *arXiv preprint arXiv:2310.07707*, 2023. URL https://arxiv.org/abs/2310.07707.

Maha Elbayad, Jiatao Gu, Edouard Grave, and Michael Auli. Depth-adaptive transformer. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://arxiv.org/abs/1910.10073.

Angela Fan, Edouard Grave, and Armand Joulin. Reducing transformer depth on demand with structured dropout. In *International Conference on Learning Representations (ICLR)*, 2020. URL https://arxiv.org/abs/1909.11556.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2407.15595.

Marjan Ghazvininejad, Omer Levy, Yinhan Liu, and Luke Zettlemoyer. Mask-predict: Parallel decoding of conditional masked language models. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2019. URL https://arxiv.org/abs/1904.09324.

Aaron Gokaslan, Vanya Cohen, Ellie Pavlick, and Stefanie Tellex. OpenWebText corpus. *http://Skylion007.github.io/OpenWebTextCorpus*, 2019. URL https://skylion007.github.io/OpenWebTextCorpus/.

Alex Graves. Adaptive computation time for recurrent neural networks. *arXiv preprint arXiv:1603.08983*, 2016. URL https://arxiv.org/abs/1603.08983.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O. K. Li, and Richard Socher. Non-autoregressive neural machine translation. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1711.02281.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. SSD-LM: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2023. URL https://arxiv.org/abs/2210.17432.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, Tom Hennigan, Eric Noland, Katie Millican, George van den Driessche, Bogdan Damoc, Aurelia Guy, Simon Osindero, Karen Simonyan, Erich Elsen, Jack W. Rae, Oriol Vinyals, and Laurent Sifre. Training compute-optimal large language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2203.15556.

Gagan Jain. Adaptive block diffusion: Resolving training–inference mismatch in diffusion language models. *arXiv preprint arXiv:2606.29275*, 2026. URL https://arxiv.org/abs/2606.29275.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. URL https://arxiv.org/abs/2001.08361.

Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, 2015. URL https://arxiv.org/abs/1412.6980.

Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. Matryoshka representation learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2205.13147.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2018. URL https://arxiv.org/abs/1802.06901.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning (ICML)*, 2023. URL https://arxiv.org/abs/2211.17192.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1711.05101.

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning (ICML)*, 2024. URL https://arxiv.org/abs/2310.16834.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. Building a large annotated corpus of English: The PennTreebank. 1993. URL https://aclanthology.org/J93-2004/.

Stephen Merity, Caiming Xiong, James Bradbury, and Richard Socher. Pointer sentinel mixture models. In *International Conference on Learning Representations (ICLR)*, 2017. URL https://arxiv.org/abs/1609.07843.

Meta AI. The Llama 3.2 herd of models: 1B, 3B, 11B & 90B. https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/; model card at https://huggingface.co/meta-llama/Llama-3.2-1B, 2024. URL https://arxiv.org/abs/2407.21783.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations (ICLR)*, 2018. URL https://arxiv.org/abs/1710.03740.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. In *International Conference on Learning Representations (ICLR)*, 2025a. URL https://arxiv.org/abs/2410.18514.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. LLaDA: Large language diffusion models. *arXiv preprint arXiv:2502.09992*, 2025b. URL https://arxiv.org/abs/2502.09992.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. *arXiv preprint arXiv:2406.03736*, 2024. URL https://arxiv.org/abs/2406.03736.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The LAMBADA dataset: Word prediction requiring a broad discourse context. In *Annual Meeting of the Association for Computational Linguistics (ACL)*, 2016. URL https://arxiv.org/abs/1606.06031.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *International Conference on Computer Vision (ICCV)*, 2023. URL https://arxiv.org/abs/2212.09748.

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. FiLM: Visual reasoning with a general conditioning layer. In *AAAI Conference on Artificial Intelligence*, 2018. URL https://arxiv.org/abs/1709.07871.

Boris T. Polyak and Anatoli B. Juditsky. Acceleration of stochastic approximation by averaging. *SIAM Journal on Control and Optimization*, 1992. URL https://epubs.siam.org/doi/10.1137/0330046.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Technical Report*, 2019. URL https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2406.07524.

Nikolay Savinov, Junyoung Chung, Mikolaj Binkowski, Erich Elsen, and Aaron van den Oord. Step-unrolled denoising autoencoders for text generation. In *International Conference on Learning Representations (ICLR)*, 2022. URL https://arxiv.org/abs/2112.06749.

Tal Schuster, Adam Fisch, Jai Gupta, Mostafa Dehghani, Dara Bahri, Vinh Q. Tran, Yi Tay, and Donald Metzler. Confident adaptive language modeling. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022. URL https://arxiv.org/abs/2207.07061.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. MD4: Simplified and generalised masked diffusion for discrete data. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024. URL https://arxiv.org/abs/2406.04329. alias for shi2024simplified.

Mitchell Stern, Noam Shazeer, and Jakob Uszkoreit. Blockwise parallel decoding for deep autoregressive models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. URL https://arxiv.org/abs/1811.03115.

Jianlin Su, Yu Lu, Shengfeng Pan, Bo Wen, and Yunfeng Liu. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 2024. URL https://arxiv.org/abs/2104.09864.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. LLaMA: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. URL https://arxiv.org/abs/2302.13971.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2017. URL https://arxiv.org/abs/1706.03762.

Alex Warstadt, Alicia Parrish, Haokun Liu, Anhad Mohananey, Wei Peng, Sheng-Fu Wang, and Samuel R. Bowman. BLiMP: The benchmark of linguistic minimal pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392, 2020. URL https://arxiv.org/abs/1912.00582.

Chengyue Wu, Hao Zhang, Shuchen Xue, Zhijian Liu, Shizhe Diao, Ligeng Zhu, Ping Luo, Song Han, and Enze Xie. Fast-dLLM: Training-free acceleration of diffusion LLM by enabling KV cache and parallel decoding. In *International Conference on Learning Representations (ICLR)*, 2026. URL https://arxiv.org/abs/2505.22618.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. XLNet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. URL https://arxiv.org/abs/1906.08237.

Jiahui Yu, Linjie Yang, Ning Xu, Jianchao Yang, and Thomas Huang. Slimmable neural networks. In *International Conference on Learning Representations (ICLR)*, 2019. URL https://arxiv.org/abs/1812.08928.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2019. URL https://arxiv.org/abs/1905.07830.

CONTENTS

1 Introduction **1**

2 Background and motivation **3**

3 The DIAL framework **3**
3.1 What the framework is . . . 3
3.2 Core: Mixed Block-Size Training . . . 4
3.3 Modules that keep the core trainable . . . 5
3.4 Modules that keep every setting efficient . . . 5

4 Experiments **6**
4.1 Q1 and Q2: one model outperforms the grid, over a $16\times$ scheduled cost range at matched T . . . 6
4.2 Q3: what the dial costs on downstream tasks . . . 7
4.3 Q4: which mechanism earns which gain . . . 7

5 Related work **8**

6 Discussion, limitations and conclusion **8**

A Derivations and proofs **15**
A.1 The block-diffusion objective . . . 15
A.2 ELBO consistency . . . 15
A.3 Autoregressive factorisation at $L'=1$ . . . 16
A.4 MBT regret bound . . . 17
A.5 A local optimisation observation about RDR . . . 18
A.6 Empirical convergence analysis of the stopping rule . . . 19

B Architecture, masks and cost accounting **20**
B.1 Mask construction . . . 20
B.2 Attention-score and feed-forward accounting, and the full forward pass . . . 21
B.3 Write-on-block KV caching for the clean stream . . . 22

C Pseudocode **23**

D Experimental details **25**
D.1 Datasets and evaluation protocols . . . 25
D.2 Hyperparameters . . . 25
D.3 Training budget, model selection and contamination . . . 25

E Additional results **27**
E.1 Off-design regret: making "elastic" a number . . . 27
E.2 Head-to-head with published discrete-diffusion baselines . . . 29
E.3 Adaptive stopping: per-position analysis . . . 30
E.4 Elasticity replicates across training corpora . . . 31
E.5 Model-size scaling . . . 32
E.6 Downstream task evaluation . . . 32
E.7 LAMBADA accuracy-@1 for diffusion models . . . 33
E.8 Prompt-conditioned generation and text infilling . . . 34
E.9 Wall-clock efficiency, and the comparisons we do not make . . . 34
E.10 Further ablations . . . 34
E.11 Block sizes outside the training support . . . 35
E.12 Quality against realised compute . . . 35
E.13 Sequential ablation: isolating MBT . . . 36
E.14 Seed variability on the central comparisons . . . 37
E.15 Interactive latency, batch size and memory . . . 38
E.16 Mechanism diagnostics: RDR by block size, and gradient alignment . . . 38
E.17 Ranking margins and rank agreement . . . 39

F Failure cases and remaining gaps **40**

A Derivations and proofs

A.1 The block-diffusion objective

We first record the base block-diffusion objective optimized by both BD3-LM and DIAL, since the MBT objective is defined as a mixture over this base loss and the proofs below proceed term by term. Fix a block size L' and write $B = N/L'$; block-wise factorisation gives, for any θ,

$$-\log p_\theta(\boldsymbol{x}_0) \;=\; \sum_{b=1}^{B} -\log p_\theta\big(\boldsymbol{x}_0^{(b)} \mid \boldsymbol{x}_0^{(<b)}\big). \tag{7}$$

Each conditional is a masked diffusion model on the L' tokens of block b given the clean prefix, so applying the absorbing-state negative ELBO (Sahoo et al., 2024; Shi et al., 2024) to each conditional gives, with the positive weight $w(t) = -\dot{\alpha}_t/(1-\alpha_t)$ of Eq. 2,

$$-\log p_\theta\big(\boldsymbol{x}_0^{(b)} \mid \boldsymbol{x}_0^{(<b)}\big) \;\le\; \mathbb{E}_{t,\,\boldsymbol{x}_t^{(b)}}\left[w(t) \sum_{i \in b:\, x_t^i = m} -\log f_\theta\big(x_0^i \mid \boldsymbol{x}_t^{(b)},\, \boldsymbol{x}_0^{(<b)},\, \sigma_b\big) \right]. \tag{8}$$

Summing over b recovers Eq. 2 under the L'-block mask up to a θ-independent constant, and we write that sum as $\mathcal{L}_{L'}(\theta)$. Two properties are used repeatedly below: it upper-bounds the negative log-likelihood for *every* L', and the same parameters appear in every term, with L' entering only through the attention mask and the per-block noise level σ_b.

A.2 ELBO consistency

Proposition 1 (ELBO consistency)**.** *For every fixed L', $\mathcal{L}_{L'}(\theta)$ is a valid upper bound on $-\log p_\theta(\boldsymbol{x}_0)$ under the L'-block factorisation, and $\mathcal{L}_{\mathrm{MBT}}(\theta)$ is a uniform mixture of such bounds, one per $L' \in \mathcal{S}$. Adding the zero-initialised RDR path of Eq. 5 preserves both statements at initialisation.*

Proof. The first claim is Eq. 8 summed over blocks. For the second, take the expectation over $L' \sim \mathcal{U}(\mathcal{S})$; expectation preserves the inequality termwise, so

$$\mathcal{L}_{\mathrm{MBT}}(\theta) \;=\; \frac{1}{|\mathcal{S}|} \sum_{L' \in \mathcal{S}} \mathcal{L}_{L'}(\theta) \;\ge\; \frac{1}{|\mathcal{S}|} \sum_{L' \in \mathcal{S}} \big[-\log p_\theta^{(L')}(\boldsymbol{x}_0)\big], \tag{9}$$

where $p_\theta^{(L')}$ is the model read out under the L'-block factorisation. MBT therefore minimises a uniform mixture of valid bounds rather than a bound on a single mixture distribution, and the distinction matters: DIAL is not a mixture model over block sizes but one parameter set that is simultaneously a valid model under each of $|\mathcal{S}|$ factorisations, which is what lets L' be chosen after training. For the third, $W_{\mathrm{sc}} = \mathbf{0}$ at initialisation makes the inserted term $W_{\mathrm{sc}}\,\mathrm{Embed}(\boldsymbol{x}_0^{\mathrm{prev}})$ identically zero, so the network and hence the bound are unchanged; once W_{sc} trains away from zero the network conditions on $\boldsymbol{x}_0^{\mathrm{prev}}$, which is a deterministic function of the same $\boldsymbol{x}_t$ the bound already conditions on, so each block term remains a conditional cross-entropy and the bound stays valid. Validity is all that is preserved, and in particular nothing here says the bound gets tighter. □

Two caveats on scope. A previous version of this proposition also asserted an $O(1/T)$ inner-loop discretisation error. That rate is standard for the continuous-time absorbing diffusion ELBO with a fixed denoiser, but the RDR loop re-enters the network with a draft that itself depends on T, so the standard argument does not transfer and we withdraw the claim; we have no rate for the inner loop. Separately, an earlier version of this appendix offered a Jensen reading of Eq. 9 as an account of why MBT helps, arguing that minimising the average of the $|\mathcal{S}|$ bounds pushes θ towards agreement across factorisations. We withdraw it. The inequality is true but the conclusion does not follow from it: $-\log \bar{p}$ is minimised by making any single $p^{(L')}$ large, since a log of a sum is dominated by its largest term, so the Jensen step exerts no pressure towards agreement and its slack is precisely the term that would. Whatever pressure towards competence at every L' exists comes from averaging the negative log-likelihoods in the log domain, which is the unremarkable observation that a sum penalises a blown-up term. The argument also reasoned through $\bar{p}$, the mixture over block sizes that this same proof disowns two paragraphs above. We have no theoretical account of why MBT helps.

A.3 AUTOREGRESSIVE FACTORISATION AT $L'=1$

Proposition 2 (AR factorisation at $L'=1$)**.** *At $L'=1$ the* DIAL *forward pass conditions each position on exactly the strict past. Writing $f_\theta(\cdot \mid \boldsymbol{x}_0^{(<i)}, \sigma)$ for the predictive at noise level σ, $\mathcal{L}_1(\theta)$ is the $(-\dot\alpha_t)$-weighted average over t of the AR cross-entropies of $f_\theta(\cdot \mid \boldsymbol{x}_0^{(<i)}, \sigma(t))$, and it upper-bounds the AR cross-entropy of the mixture predictive induced by an* unstopped, continuous-time *$L'=1$ sampler, with equality if and only if f_θ is invariant to σ at $L'=1$. It says nothing about the sampler we deploy, which is neither (see the second caveat below).* DIAL *at $L'=1$ therefore* induces *an autoregressive factorisation; it is not the same function as any particular causal transformer, and its bound is not in general tight.*

Proof. At $L'=1$ each block is a single token, so $B=N$ and Eq. 7 is literally the autoregressive chain rule; what remains is that the network computes the corresponding conditional. The block-diagonal mask $\mathbf{M}_{\text{self}}$ degenerates to the identity $\mathbb{I}_N$, so self-attention on $\boldsymbol{x}_t$ makes each position a function of its own embedding alone, while $\mathbf{M}_{\text{cross}}$ becomes the strictly lower-triangular mask of causal attention, so the cross path is a causal attention output from $\boldsymbol{x}_t$'s queries onto the clean stream's keys and values. Composing the two with adaLN modulation, position i's representation at every layer is a function of x_t^i and of $\boldsymbol{x}_0^{(<i)}$ and of nothing else, by induction over layers: the self path contributes only position i and the cross path only strictly earlier clean positions, and neither introduces a dependence on x_t^j for $j \neq i$ or on x_0^j for $j \geq i$. The receptive field is therefore exactly the strict past, which is what the chain rule requires.

This is a statement about the receptive field, not about function identity with any particular causal transformer. DIAL at $L'=1$ is a cross-attention-only causal network reading a per-token embedding of the prefix; a standard causal transformer contextualises the prefix through self-attention, and BD3-LM at $L'=1$ does so too through the clean-to-clean quadrant $\mathbf{M}_{\text{BC}}$ that DIAL does not have (App. B.1). All three admit the same conditional dependencies and hence the same factorisation; they are different parameterisations of the family of causal conditionals, not the same function.

For the loss, write out the cancellation explicitly. A single-token block has exactly one maskable position i, whose prefix $\boldsymbol{x}_0^{(<i)}$ is clean, so the summand of Eq. 8 is supported only on the event $\{x_t^i = m\}$. On that event the network sees the clean prefix and, through M1, the per-block noise level $\bar\sigma_b$. At $L'=1$ a block is one token, so $\bar\sigma_b = -\log\alpha(t)$ is a strictly monotone function of t and the predictive therefore *does* depend on t: $f_\theta(x_0^i \mid \boldsymbol{x}_t) = f_\theta(x_0^i \mid \boldsymbol{x}_0^{(<i)}, \sigma(t))$. An earlier version of this proof dropped that argument and concluded an exact identity; the conclusion was wrong, and it is our own M1 that makes it wrong. For a model that passes $\sigma = \mathbf{0}$, such as BD3-LM, the dropped argument is constant and the original identity does hold.

Carrying σ through, Eq. 1 masks position i with probability $1-\alpha_t$, so

$$\mathbb{E}_q\Big[\mathbb{I}[x_t^i = m]\,\big(-\log f_\theta(x_0^i \mid \boldsymbol{x}_t)\big)\Big] \;=\; -(1-\alpha_t)\,\log f_\theta\big(x_0^i \mid \boldsymbol{x}_0^{(<i)}, \sigma(t)\big), \tag{10}$$

and the $(1-\alpha_t)$ still cancels the $1/(1-\alpha_t)$ in $w(t)$ pointwise, leaving

$$\mathcal{L}_1(\theta) \;=\; \sum_i \int_0^1 (-\dot\alpha_t)\,\big(-\log f_\theta\big(x_0^i \mid \boldsymbol{x}_0^{(<i)}, \sigma(t)\big)\big)\,\mathrm{d}t. \tag{11}$$

For any schedule with $\alpha_0 = 1$ and $\alpha_1 = 0$ we have $\int_0^1 (-\dot\alpha_t)\,\mathrm{d}t = 1$, so $(-\dot\alpha_t)$ is a probability density on $[0,1]$ and Eq. 11 is an *average* of AR cross-entropies indexed by the noise level, not a single one.

That density is the right one. At $L'=1$ the reverse chain of Eq. 3 unmasks its single token at step j with probability $[\alpha(t_{j+1}) - \alpha(t_j)]/(1-\alpha(t_0))$, whose continuum limit is exactly $-\dot\alpha_t$. The conditional an *unstopped, continuous-time* $L'=1$ sampler would realise is therefore the mixture $p_\theta^{(1)}(x_0^i \mid \boldsymbol{x}_0^{(<i)}) = \int_0^1 (-\dot\alpha_t)\, f_\theta(x_0^i \mid \boldsymbol{x}_0^{(<i)}, \sigma(t))\,\mathrm{d}t$, and by Jensen applied to the convex function $-\log$,

$$\mathcal{L}_1(\theta) \;\geq\; -\sum_i \log p_\theta^{(1)}\big(x_0^i \mid \boldsymbol{x}_0^{(<i)}\big), \tag{12}$$

with equality if and only if $f_\theta(\cdot \mid \boldsymbol{x}_0^{(<i)}, \sigma)$ is constant in σ for almost every t. So $\mathcal{L}_1$ is a valid bound on a genuine AR cross-entropy, which is what the rest of the paper uses, but the gap is

a σ-dispersion term we do not measure. Two consequences worth stating. The minimiser of $\mathcal{L}_1$ still attains the AR cross-entropy, since a σ-invariant optimal predictor is admissible, so nothing in Prop. 3 is affected. And the $L' = 1$ sampler draws from a noise-level mixture of predictors rather than from a single best predictor, which is a sharper form of the caveat below. One consequence runs in our favour and we state it for completeness: M1 is what makes f_θ depend on σ, so DIAL's bound is strict at $L' = 1$ where BD3-LM's, which passes $\sigma = \mathbf{0}$, is tight. Our reported perplexities therefore overstate our induced model's cross-entropy by more than the baselines' do, and every DIAL-versus-specialist margin in Table 1 is conservative by an amount we do not measure. The per-token weight $w_{\text{tok}} = -\dot{\alpha}(t_b)/(1 - \alpha(t_b))$ of Alg. 3 is a Monte Carlo estimator of Eq. 11 under a finite sample of t and is unbiased for it. □

Two caveats belong with this proposition. It concerns the *dependency structure* and the *objective*: at $L' = 1$ the model factorises and trains exactly as an AR model. It does not say that the DIAL network is architecturally identical to a standard causal transformer in general, only that at $L' = 1$ the two masks induce the same per-position receptive field; nor does it say that *sampling* is AR.

The second caveat is the one a careful reader should press, and we state it against ourselves. The mixture above is the law of an unstopped sampler in the continuous-time limit. The sampler we actually run is neither: it takes T discrete steps and M4 exits the inner loop early, committing any position still masked at the argmax of the final draft. At $L' = 1$ that gap is widest, since Table 1 shows $2.96\times$ fewer realised than scheduled forwards, so most tokens exit before the schedule ends and are committed greedily. The deployed $L' = 1$ system is therefore closer to a greedy decoder taking about seventeen forwards per token than to the ancestral mixture this proposition bounds, and the proposition should be read as applying to the M4-disabled sampler. We have no bound on the distance between the two laws.

A.4 MBT REGRET BOUND

The only standing assumption is that the minimisers below exist; no convexity, non-negativity or smoothness of the losses is used.

Assumption 1 (Bounded sharing conflict)**.** For every $L' \in \mathcal{S}$ the minimiser $\theta^\star_{L'} = \arg\min_\theta \mathcal{L}_{L'}(\theta)$ exists, as does $\theta^\star_{\text{MBT}} = \arg\min_\theta \bar{\mathcal{L}}(\theta)$ with $\bar{\mathcal{L}} = |\mathcal{S}|^{-1} \sum_{L' \in \mathcal{S}} \mathcal{L}_{L'}$, and the worst-case *excess* loss from serving one block size's optimum at another,

$$\epsilon_{\text{share}} \;=\; \max_{L'_1,\, L'_2 \in \mathcal{S}} \big[\mathcal{L}_{L'_1}(\theta^\star_{L'_2}) - \mathcal{L}_{L'_1}(\theta^\star_{L'_1})\big], \tag{13}$$

is finite. By construction $\epsilon_{\text{share}} \geq 0$, and it vanishes exactly when some parameter vector minimises every $\mathcal{L}_{L'}$ simultaneously, i.e. when a block-partition-invariant optimum exists.

Proposition 3 (MBT regret)**.** *Under Assumption 1, for every $L' \in \mathcal{S}$, $\mathcal{L}_{L'}(\theta^\star_{\text{MBT}}) - \mathcal{L}_{L'}(\theta^\star_{L'}) \leq (|\mathcal{S}| - 1)\,\epsilon_{\text{share}}$. Only loss differences enter, so the bound is zero exactly when a block-partition-invariant optimum exists.*

Proof. Fix a target $L'^\star \in \mathcal{S}$. Multiplying the defining inequality $\bar{\mathcal{L}}(\theta^\star_{\text{MBT}}) \leq \bar{\mathcal{L}}(\theta^\star_{L'^\star})$ by $|\mathcal{S}|$ and splitting off the $L'^\star$ term on each side gives

$$\mathcal{L}_{L'^\star}(\theta^\star_{\text{MBT}}) + \sum_{L' \neq L'^\star} \mathcal{L}_{L'}(\theta^\star_{\text{MBT}}) \;\leq\; \mathcal{L}_{L'^\star}(\theta^\star_{L'^\star}) + \sum_{L' \neq L'^\star} \mathcal{L}_{L'}(\theta^\star_{L'^\star}), \tag{14}$$

so that, rearranging,

$$\mathcal{L}_{L'^\star}(\theta^\star_{\text{MBT}}) - \mathcal{L}_{L'^\star}(\theta^\star_{L'^\star}) \;\leq\; \sum_{L' \neq L'^\star} \big[\mathcal{L}_{L'}(\theta^\star_{L'^\star}) - \mathcal{L}_{L'}(\theta^\star_{\text{MBT}})\big]. \tag{15}$$

Note that only *differences* of losses appear, never an absolute loss, which is what makes the bound informative. Each summand is now bounded by ϵ_{share}: since $\theta^\star_{L'}$ minimises $\mathcal{L}_{L'}$ we have $\mathcal{L}_{L'}(\theta^\star_{\text{MBT}}) \geq \mathcal{L}_{L'}(\theta^\star_{L'})$, hence

$$\mathcal{L}_{L'}(\theta^\star_{L'^\star}) - \mathcal{L}_{L'}(\theta^\star_{\text{MBT}}) \;\leq\; \mathcal{L}_{L'}(\theta^\star_{L'^\star}) - \mathcal{L}_{L'}(\theta^\star_{L'}) \;\leq\; \epsilon_{\text{share}} \tag{16}$$

by Eq. 13. Substituting into Eq. 15, and there being $|\mathcal{S}| - 1$ summands, gives $\mathcal{L}_{L'^\star}(\theta^\star_{\text{MBT}}) - \mathcal{L}_{L'^\star}(\theta^\star_{L'^\star}) \leq (|\mathcal{S}| - 1)\,\epsilon_{\text{share}}$, the claim. □

The bound is tight at both ends. When $\epsilon_{\text{share}} = 0$ it gives zero regret, and that is correct rather than an artefact: if some θ minimises every $\mathcal{L}_{L'}$ then $\bar{\mathcal{L}}$ attains the average of the individual minima, and since no term can fall below its own minimum every term must equal it, so $\theta^\star_{\text{MBT}}$ is simultaneously optimal at every block size. At the other end the constant cannot be improved below $|\mathcal{S}| - 1$ in general, since the target block size contributes nothing to its own regret and each of the other $|\mathcal{S}| - 1$ objectives can independently pull the joint minimiser away by ϵ_{share}.

Two limitations should be kept in view, and the first is severe enough that we state it with numbers rather than in the abstract.

The bound is far too loose to constrain anything we observe. Substituting the trained specialist for $\theta^\star_{L'}$, exactly as we do when reporting empirical regret in App. E.1, Table 5 *is* an estimate of the cross-block excesses that define ϵ_{share}: its largest entry is 24.1, so $(|\mathcal{S}| - 1)\,\epsilon_{\text{share}}$ is of order 96, against an observed off-diagonal regret near zero. Taking instead the 5.4 of the released-weight sub-block (App. E.1) gives order 22, which is vacuous by the same margin. A bound that permits regret several times the loss itself carries no information about this model. Earlier drafts of this appendix asserted that we do not measure ϵ_{share}; that was wrong, and the measurement is unflattering.

The row-sum form is the less loose one, though still vacuous. Eq. 15, *before* it is weakened by taking the maximum, bounds the regret at $L'^\star$ by the off-diagonal sum of the $L'^\star$ specialist's excess over the MBT model, which is a sum over Table 1 and not over Table 5: at $L'^\star = 4$ that is $(33.1 - 28.0) + (31.0 - 28.4) + (32.5 - 29.7) + (37.6 - 30.6) = 17.5$, about $5.5\times$ tighter than the max form, and still a bound of $+17.5$ on a quantity we measure at -1.5. An earlier version of this passage read the sum off the regret table instead, giving 10.2; that substitution is invalid exactly because DIAL's regret is negative, and it understates the bound by the 7.3 points of negative regret it silently discards. Both figures are indicative rather than exact, since Table 5 is in perplexity points while $\mathcal{L}$ is a mean negative log-likelihood in nats; the ordering of magnitudes is what matters here, and it does not depend on the conversion.

Second, it is a statement about exact minimisers of population objectives, so it says nothing about what stochastic optimisation over 10^6 steps reaches. The obstruction is sharper than generic non-convergence: MBT draws one L' per step and AdamW's preconditioner is nonlinear in the gradient, so the fixed point of Adam under stochastically alternating objectives need not minimise their average even in the limit.

The bound is one-sided and gives no reason to expect the elastic model to *beat* the specialists. Empirically the regret is negative at every operating point, with the largest margins at the corners of the dial (28.0 against 30.5 at $L' = 1$ and 30.6 against 32.2 at $L' = 16$), consistent with implicit-regularisation arguments for multi-task training in which sharing benefits the hard large-block regime more than it costs the easy small-block one. Our theory does not predict this, and we report it as an empirical finding.

A.5 A LOCAL OPTIMISATION OBSERVATION ABOUT RDR

We label this an observation rather than a justification, and say at the outset what it is: an instance of the elementary fact that a differentiable objective admits a descent step along any non-zero gradient. It carries no information specific to self-conditioning beyond the claim that the relevant gradient is non-zero, and a reviewer who reads it as tautological is reading it correctly. We keep it only because it is the precise statement that motivates the *zero* initialisation of W_{sc}, which is a real design choice with a measurable failure mode (App. E.10), and because stating it exactly makes clear how little it claims.

The condition needed is not "the model is better than random": it is an alignment condition on the *gradient* of the log-likelihood with respect to the self-conditioning path. A model can be better than random while its draft embedding is redundant with, or misaligned against, the direction that would reduce the loss. We therefore state alignment as an assumption, and treat the ablation (Table 3) rather than this proposition as the evidence for RDR.

Assumption 2 (Draft alignment)**.** At $W_{\text{sc}} = \mathbf{0}$, the expected gradient $\mathbb{E}[g] = \mathbb{E}\big[\nabla_{W_{\text{sc}}} \log p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_t, W_{\text{sc}}\,\text{Embed}(\boldsymbol{x}_0^{\text{prev}}))\big|_{W_{\text{sc}}=0}\big]$ is non-zero, so conditioning on $\boldsymbol{x}_0^{\text{prev}}$ carries information about $\boldsymbol{x}_0$ that the noisy stream does not already supply.

Proposition 4 (Descent direction at zero initialisation)**.** *Under Assumption 2 and with $W_{\mathrm{sc}} = \mathbf{0}$ at initialisation, the first-order effect of one gradient step in W_{sc} on the expected denoising loss is non-positive, and strictly negative for small enough step size.*

Proof. Let $\mathcal{L}(\theta; W_{\mathrm{sc}}) = \mathbb{E}_{(\boldsymbol{x}_0, \boldsymbol{x}_t, \boldsymbol{x}_0^{\mathrm{prev}})}\big[-\log p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_t, W_{\mathrm{sc}}\,\mathrm{Embed}(\boldsymbol{x}_0^{\mathrm{prev}}))\big]$ and $g(\boldsymbol{x}_0, \boldsymbol{x}_0^{\mathrm{prev}}) = \nabla_{W_{\mathrm{sc}}} \log p_\theta(\boldsymbol{x}_0 \mid \boldsymbol{x}_t, W_{\mathrm{sc}}\,\mathrm{Embed}(\boldsymbol{x}_0^{\mathrm{prev}}))\big|_{W_{\mathrm{sc}}=0}$. The integrand is dominated by an integrable envelope, since the logit map is Lipschitz in W_{sc} on the bounded embedding set, so we may differentiate under the expectation and $\nabla_{W_{\mathrm{sc}}}\mathcal{L}|_{W_{\mathrm{sc}}=0} = -\mathbb{E}[g]$. A gradient step $W_{\mathrm{sc}}^{(1)} = \eta\,\mathbb{E}[g]$ then gives $\mathcal{L}(\theta; W_{\mathrm{sc}}^{(1)}) = \mathcal{L}(\theta; \mathbf{0}) - \eta\,\|\mathbb{E}[g]\|^2 + O(\eta^2)$, so the first-order term is strictly negative whenever $\mathbb{E}[g] \neq \mathbf{0}$, which is Assumption 2. We state no step-size threshold, since Assumption 2 supplies no smoothness constant with which to bound the $O(\eta^2)$ remainder. □

What this does and does not give. It is a statement about the derivative at $W_{\mathrm{sc}} = \mathbf{0}$, so it justifies the *zero* initialisation rather than the value of the converged loop: the path is inert at the start of training and can only be turned on in a descent direction. Nothing in the argument is particular to RDR; the same two lines would apply to any zero-initialised additive path whose gradient does not vanish, which is exactly why we do not present it as evidence that self-conditioning helps. It says nothing about the fully trained recurrent loop, about the exposure mismatch between training with $p_{\mathrm{sc}} = 0.5$ and inference with the loop always on, or about error accumulation across inner steps. With a random non-zero W_{sc} the first drafts are noise, Assumption 2 can fail, and the conditional likelihood can decrease instead; empirically this shows up as divergence on four of five seeds (App. E.10), which is the observation the proposition rationalises.

A.6 Empirical convergence analysis of the stopping rule

We present this as an empirical convergence analysis rather than a guarantee. The geometric decay below follows from a contraction assumption that we *measure* on generation traces and do not establish for the model: Lipschitz continuity of the logit map does not by itself imply a contraction, which additionally requires the modulus to be below one in the relevant metric, and the inner map here composes network nonlinearities, an argmax, changing masks and changing diffusion states. The conditional statement is still worth making because it says what the measured ρ would buy, and because it makes the assumption falsifiable.

Assumption 3 (Inner-loop contraction)**.** The inner denoising map satisfies $\mathrm{TV}(\boldsymbol{p}^{(t+1)}, \boldsymbol{p}^{(t)}) \leq \rho\,\mathrm{TV}(\boldsymbol{p}^{(t)}, \boldsymbol{p}^{(t-1)})$ for some $\rho < 1$, where the draft fed to step $t+1$ is the argmax of step t.

Proposition 5 (Predictive drift at the stopping time; a diagnostic, not a justification for M4)**.** *Let $C_0 \geq \mathrm{TV}(\boldsymbol{p}^{(1)}, \boldsymbol{p}^{(0)})$ and suppose Assumption 3 holds with a limit $\boldsymbol{p}^{(\infty)}$. Any block committed by the rule of Eq. 6 has run at least $W + K$ inner steps, where W is the warm-up, and its predictive distribution at the stopping time τ satisfies $\mathrm{TV}(\boldsymbol{p}^{(\tau)}, \boldsymbol{p}^{(\infty)}) \leq \rho^{W+K} C_0/(1-\rho)$ almost surely.*

Proof. Iterating Assumption 3 gives $\mathrm{TV}(\boldsymbol{p}^{(j+1)}, \boldsymbol{p}^{(j)}) \leq \rho^j C_0$, so the iterates are Cauchy and, summing the tail of the geometric series from any step t, $\mathrm{TV}(\boldsymbol{p}^{(t)}, \boldsymbol{p}^{(\infty)}) \leq \sum_{j \geq t} \rho^j C_0 = \rho^t C_0/(1-\rho)$. Eq. 6 tests K consecutive agreements after a W-step warm-up and so cannot fire before step $W+K$, giving $\tau \geq W + K$ pathwise. Since the tail bound is decreasing in t, substituting $t = \tau$ gives the claim for every trajectory. □

Three things about this statement should be read before it is used, and together they are why we call it a diagnostic.

It is not a bound on the bias M4 introduces, despite the name it used to carry. The quantity bounded is the drift of the *predictive* distribution between step τ and the contraction's limit. The bias M4 actually introduces is different: leaving the loop early sends the residual masked positions to the argmax of the final draft (Alg. 2), so the committed token is a mode where Eq. 3 would have drawn a sample. Nothing here bounds the distance between those two laws, and we do not have a bound for it.

The bound also runs the wrong way for the module. Since ρ^t decreases in t, a sampler that never stops ($\tau = T$) enjoys a strictly better guarantee than one that stops at $W + K = 5$. Read as an

argument, the proposition favours not using M4, which is the honest reading and the reason the case for M4 is the measurement.

Finally, Assumption 3 presumes a limit $\boldsymbol{p}^{(\infty)}$ for a loop that is neither autonomous nor infinite: $\bar{\sigma}$ and η_j both change with j, the partially unmasked state evolves, and the loop terminates at T. The assumption should be read as describing the observed behaviour of the first T iterates rather than as asserting a fixed point.

Where the guarantee comes from is also not where the name suggests. The bound is a consequence of the *step count* that Eq. 6 enforces; the argmax agreements themselves contribute nothing to it. That gap is real rather than an artefact of a loose proof: $\boldsymbol{p}^{(t)} = (0.51, 0.49)$ and $\boldsymbol{p}^{(t+1)} = (0.90, 0.10)$ agree on the argmax while differing substantially in total variation, so no bound on distributional distance can be derived from agreement alone. Turning the observed agreements into distributional information would need a further stability condition linking argmax equality to $\mathrm{TV}(\boldsymbol{p}^{(t)}, \boldsymbol{p}^{(t-1)})$, which we neither assume nor verify. What the rule does bound directly is the *committed token*, which is the argmax and is by construction unchanged over the agreement window; that is the relevant target here, since only the argmax is committed.

Regressing successive-iterate total variation on a generation trace gives $\rho \approx 0.30$ and $C_0 \approx 0.10$ at $T = 100$, $L' = 4$. We do not instantiate the bound at those values, because our own stopping statistics rule them out. Under $\rho = 0.30$ successive drafts would differ by $\rho^{W+K} C_0 \approx 2.4 \times 10^{-4}$ in total variation by the first step at which Eq. 6 can fire, so the argmax would be stable almost everywhere and M4 would exit at $\tau = W + K = 5$ on essentially every block. What we measure is 24.8 inner steps per block at $L' = 4$, 40.0 at $L' = 8$, and no firing at all within 50 steps at $L' = 16$ (Table 1). The fitted ρ is therefore inconsistent with the observed stopping times by a factor of five at $L' = 4$ and by more than an order of magnitude at $L' = 16$, and batch coupling does not close the gap, since a conjunction over 16 sequences and L' positions at 10^{-4} per position still holds with overwhelming probability.

The natural reading is that a regression of successive-iterate total variation recovers a short-horizon slope, while the inner loop plateaus at a noise floor rather than contracting geometrically, which is precisely why a fitted slope is not a contraction modulus. So Assumption 3 does not hold in the regime where M4 operates, and Prop. 5 should be read as a statement about a process our sampler is not, with no numerical instantiation we are willing to defend. We keep it only to make the shape of the missing guarantee explicit. App. E.3 reports the top-k stability and total-variation thresholds we used as cross-checks.

B Architecture, masks and cost accounting

B.1 Mask construction

We construct attention masks from two predicates,

$$\mathbf{M}_{\text{self}}(q, k) = \mathbb{I}\big[\lfloor q/L' \rfloor = \lfloor k/L' \rfloor\big], \qquad \mathbf{M}_{\text{cross}}(q, k) = \mathbb{I}\big[\lfloor k/L' \rfloor < \lfloor q/L' \rfloor\big]. \tag{17}$$

For the SDPA back-end we materialise dense boolean tensors of shape $N \times N$ and substitute $-\infty$ entries; for the `flex_attention` back-end we pass the predicates to `create_block_mask`, which compiles a sparse block mask at $\mathcal{O}(NL')$ cost. Both sit behind a `MaskCache` keyed by (L', N, device), so redrawing L' under MBT costs a dictionary lookup rather than a mask rebuild. The support is finite and the sequence length fixed, so exactly $|\mathcal{S}|$ masks are ever built and every draw after the first of each is a hit; we quote no hit rate because over a 10^6-step run the figure is arbitrarily close to one by construction and measures nothing about the mechanism. The fused $2N \times 2N$ mask of BD3-LM is the disjoint union of $\mathbf{M}_{\text{BD}}$ (noisy-to-noisy), $\mathbf{M}_{\text{OBC}}$ (noisy-to-clean) and $\mathbf{M}_{\text{BC}}$ (clean-to-clean); under the two-stream split the first two become $\mathbf{M}_{\text{self}}$ and $\mathbf{M}_{\text{cross}}$, and the third has no counterpart at all, because DIAL's clean stream is a per-token embedding that is never self-attended (Fig. 3). Dropping that quadrant is the substantive architectural change, not a bookkeeping one: in BD3-LM a clean position's key is a contextual function of the whole committed prefix, whereas in DIAL it depends only on the token at that position.

Two further respects in which the split is a modification rather than a factorisation of the same computation deserve stating, since the rest of this appendix accounts for cost as though the two were

comparable. First, BD3-LM passes every query through *one* softmax over all $2N$ keys, while DIAL applies two masked softmaxes in sequence, one over N noisy keys and one over N clean keys; even with identical admissible edges the normalisers differ, so the two attention operators compute different functions for any non-degenerate score matrix. Second, DIAL runs one feed-forward block per layer, on the N noisy positions only, where BD3-LM runs it over all $2N$ fused positions. We therefore describe DIAL throughout as an architectural variant of BD3-LM at matched scale and training budget, and read the two-stream row of Table 3 as a mechanism effect rather than as an efficiency-only change.

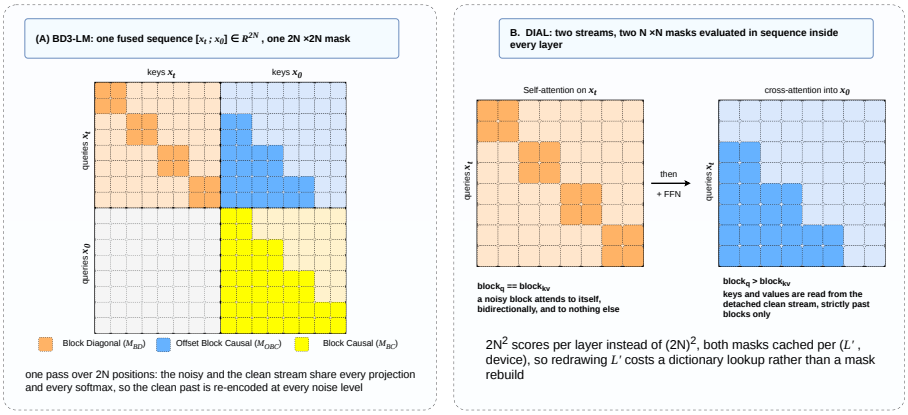

Figure 3: **One fused mask versus two stream masks**, drawn at $N = 8$ and $L' = 2$; a saturated cell is attended, a pale cell of the same hue is blocked. **(A)** BD3-LM concatenates the streams into $[\boldsymbol{x}_t; \boldsymbol{x}_0]$ and builds one $2N \times 2N$ mask with four quadrants: block-diagonal (▭) for noisy-to-noisy, offset block-causal (▭) for noisy-to-clean, block-causal (▭) for clean-to-clean, and a zero quadrant (▭) because clean positions never read the noisy stream; every query passes through one softmax over $2N$ keys, so the clean past is re-encoded at every noise level. **(B)** DIAL applies two $N \times N$ predicates in sequence inside every layer: block-diagonal self-attention on the noisy stream (▭), then strictly block-causal cross-attention into the detached clean stream (▭), then the feed-forward network. The two live quadrants of (A) survive as those masks; the clean-to-clean quadrant has no counterpart because the clean stream is embedded per token and never self-attended. The result is $2N^2$ scores per layer instead of $(2N)^2$.

B.2 Attention-score and feed-forward accounting, and the full forward pass

We give two accountings. The first covers attention scores and feed-forward blocks only and is directly comparable to the published BD3-LM numbers; the second adds the projections and the vocabulary head, and is the one to quote for a forward pass, since the first overstates the saving.

Attention scores and feed-forward blocks. Per-layer counts, ignoring biases and the vocabulary projection. Attention scores are counted in FLOPs and the feed-forward blocks in multiply–accumulates, following the convention of the BD3-LM paper so the two rows are comparable to its published accounting. **BD3-LM** fuses the streams into a length-$2N$ sequence, so attention costs $16N^2d$ and the feed-forward blocks, run over all $2N$ fused positions, cost $16Nd^2$ per layer. **DIAL** splits them: noisy self-attention costs $4N^2d$ and cross-attention another $4N^2d$, for $8N^2d$ of attention against BD3-LM's $16N^2d$. Its feed-forward block runs on the N noisy positions *only*, since the clean stream is an embedding read through key and value projections and never passes through an MLP (App. B.1), so the feed-forward cost is $8Nd^2$, half of BD3-LM's. Both of these terms therefore fall by a factor of two.

The whole forward pass. That factor does not survive the terms the convention above omits, and the difference is large enough to state. Counting every matrix product as one multiply–accumulate, BD3-LM spends $6Nd^2$ on the fused stream's query, key and value projections and $2Nd^2$ on its output projection, while DIAL spends $4Nd^2$ on the noisy stream's self-attention and another $4Nd^2$ on cross-attention, whose keys and values are projected from the N clean positions. The projection

term is thus $8Nd^2$ for *both*, and only the score and feed-forward terms halve:

$$C^{\text{layer}}_{\text{BD3-LM}} = 8N^2d + 24Nd^2, \qquad C^{\text{layer}}_{\text{DIAL}} = 4N^2d + 16Nd^2, \tag{18}$$

a ratio of 1.63 rather than 2 at $N=1024$, $d=768$. Adding the vocabulary head, which both evaluate at all N positions and which costs $NdV = 3.95\times10^{10}$ multiply–accumulates at $V = 50\,258$, the twelve-layer totals are 2.91×10^{11} for BD3-LM against 1.94×10^{11} for DIAL, a ratio of 1.50. So the honest headline for M3 is a $1.5\times$ cheaper forward pass, not a halved one; the head alone is 20% of DIAL's total at this scale and dilutes any saving in the body.

The clean stream depends only on committed tokens, so its per-layer keys and values do not change across the T inner steps of a block and we compute them once per block (App. B.3), leaving $4N^2d$ of attention on the inner-step path. Block-diagonal masks compress self-attention from $4N^2d$ to $4NL'd$, which FlashAttention-2 (Dao, 2024) exploits: a $256\times$ reduction on that path at $N=1024$, $L'=4$. On memory, BD3-LM stores a $2N\times2N$ attention buffer per layer, $4N^2$ entries and so $8N^2$ bytes in BF16, against DIAL's two $N\times N$ buffers at $2N^2$ entries and $4N^2$ bytes. The $2\times$ ratio is what matters and is unaffected.

What the accounting does and does not claim. These constant factors are not a contribution, and the operational claim needs stating precisely, because three different quantities are easy to conflate here. The *peak activation tensor dimensions* are set by N and are independent of L', which is what makes peak memory nearly flat across the dial (measured in App. E.15). The *number of attended score pairs* is not: it is NL' on the self-attention path, sixteen times larger at $L'=16$ than at $L'=1$. And *wall clock* follows neither exactly, since a masked kernel realises only part of the sparsity. We claim only the first, and only as an accounting statement that the measurement corroborates. Two details: cross-attention is the only sub-layer receiving a gate but no shift and no scale, which is why the adaLN projection emits nine chunks of width d rather than six; and σ reaches the network through that same projection, which is how a resampled block size becomes legible to the model at all.

B.3 Write-on-block KV caching for the clean stream

During sampling the cross-attention context at block k is the decoded prefix followed by a fully masked tail. The prefix does not move while block k is denoised, so its per-layer keys and values would otherwise be recomputed identically on each of the T inner steps; we remove that recomputation with a cache written once per block, when the block is committed, and read by every inner step of the blocks that follow.

What is cached is exactly the per-layer cross-attention key and value tensors of the committed prefix. Writing Embed for the clean-stream token embedding, which is the whole of the clean stream and involves no self-attention (App. B.1), the cached quantities at layer l after block b has been committed are

$$K_b^{(l)} \;=\; W_K^{(l)}\,\mathrm{Embed}\big(\boldsymbol{x}_0^{(\le b)}\big), \qquad V_b^{(l)} \;=\; W_V^{(l)}\,\mathrm{Embed}\big(\boldsymbol{x}_0^{(\le b)}\big), \tag{19}$$

so caching is an append along the sequence axis and needs no incremental re-encoding of the prefix. Had the clean stream been a multi-layer causal encoder instead, $K_b^{(l)}$ would be a contextual function $h_i^{(l)} = F(h_{\le i}^{(l-1)})$ of the whole prefix, and the write-once argument would require an incremental causal encode rather than a projection; we chose the per-token parameterisation partly for that reason.

Three properties then make the cached and recomputing paths produce identical logits rather than merely close ones. Both Embed and the projections $W_K^{(l)}, W_V^{(l)}$ of Eq. 19 are position-wise, so the keys of a span depend only on the token ids in that span; the cross-attention key and value path carries no rotary or additive positional term, so writing a block's keys when it is decoded gives the same tensor a later recomputation would; and we cache only committed positions, projecting the masked tail fresh at every inner step and concatenating it after the cached prefix, which preserves key order and therefore the attention pattern. Empirically, at $N=1024$ with the full twelve-layer model, we observed identical token sequences with and without the cache at every L' we test. We claim this as an observation over the configurations tested, not as bitwise equivalence: it holds for the

attention back-end, precision and seeds we used, and reduction-order differences on other back-ends could break exact bitwise agreement without changing the argument.

The cache holds one key and one value tensor of shape $[B, N, d]$ per layer, 36 MiB per sequence in BF16 at our configuration, and is the only inference-time state DIAL keeps beyond the working sequence. On one RTX A6000 at $N = 1024$ with batch 2 it removes essentially all of the 3.3 ms the clean-stream key and value projection costs within a 58 ms inner step, a $1.06\times$ end-to-end speedup at both $(L' = 4, T = 8)$ and $(L' = 16, T = 16)$. The gain is modest because the inner step is dominated by the noisy stream's own attention and feed-forward blocks and by the vocabulary projection, which we still evaluate at all N positions. Every throughput number here is measured with the cache enabled.

The waste we leave on the table, and why it matters for our cost claim. That last clause deserves more than a subordinate mention, because it is the reason the dial reduces measured cost at all. Under M3 no position outside the active block can influence the active block's output: the noisy stream's self-attention mask $\mathbb{1}[\lfloor q/L' \rfloor = \lfloor k/L' \rfloor]$ confines it to the block, cross-attention reads only committed blocks from cache, and the feed-forward, adaLN and vocabulary projections are position-wise. An implementation could therefore gather the L' active positions, run the noisy stream on those alone, and scatter the result; RoPE uses global indices, so the gather need only carry them. Per-pass cost would then scale with L' rather than with N, and by App. B.2 the vocabulary head alone is 20% of the forward.

We did not build this, and the consequence is not in our favour. Our measured $5.4\times$ reduction along the dial exists because a pass costs the same at every L', so shortening the chain shortens the wall clock. Under a gathered implementation the operative unit becomes $L' \cdot \mathrm{NFE}_{\mathrm{real}}$, which *rises* $2.96\times$ across the same sweep (App. E.12). The efficiency result we report is therefore a property of dense evaluation and would not survive the obvious optimisation. We make no claim about how much of the analytic ratio a gathered kernel would realise, since narrow GEMMs at $L' = 4$ are launch- and bandwidth-bound and we have not measured one.

C Pseudocode

Algorithm 1 is the training step summarised in Sec. 3, Algorithm 2 the sampler of Sec. 3.4, and Algorithm 3 the collation and loss sub-routine the training step composes from. All hyperparameter values match App. D.2.

Algorithm 1 DIAL training step (one optimisation step).

1: Sample mini-batch $\boldsymbol{x}_0$ and a global block size $L' \sim \mathcal{U}(\mathcal{S})$ ▷ MBT
2: Truncate $\boldsymbol{x}_0$ to a length divisible by L'; form $B = N/L'$ blocks
3: Sample t; build $\boldsymbol{x}_t$ by Eq. 1; compute per-block $\bar{\sigma}_b$
4: With prob. p_{sc}: no-grad forward $\rightarrow \boldsymbol{x}_0^{\mathrm{prev}}$; else $\boldsymbol{x}_0^{\mathrm{prev}} = \mathbf{0}$
5: Forward $f_\theta(\boldsymbol{x}_t, \{\bar{\sigma}_b\}, \boldsymbol{x}_0^{\mathrm{prev}})$ under the cached L'-mask (Eq. 5)
6: Backpropagate the masked loss of Eq. 2; update θ

Two points of the sampler are worth isolating, because they are where a re-implementation most easily diverges. The agreement test compares *drafts* $\hat{\boldsymbol{x}}_0$, which exist at every position of the block whether or not that position has been unmasked, so the rule can fire while part of the block is still m; the commitment rule is what then makes the block well formed, and it is the only place a token is written without passing through Eq. 3. And the test reduces over the whole batch, so a block runs until its last straggler agrees (App. E.3); per-sample stopping can only stop earlier, which is why we call the reported savings conservative.

Three conventions in Alg. 3 are worth stating explicitly, because they are the points at which an implementation most easily drifts from Eq. 1. First, α_{tok} holds the schedule value $\alpha(t_b)$ and not the diffusion time t_b; under the loglinear schedule these coincide only up to the reparameterisation $\alpha(t) = 1 - t + \epsilon$, and we always pass the former. Second, $\mathrm{Bern}(\alpha_{\mathrm{tok}}) = 1$ is the *keep* event, so the where selects $\boldsymbol{x}_0$ on that branch and m otherwise, matching the masking probability $1 - \alpha_t$ of Eq. 1. Third, the loss weight is $w_{\mathrm{tok}} = -\dot{\alpha}/(1 - \alpha)$, whose denominator is the masking probability and whose leading minus sign makes it positive; for the loglinear schedule this reduces to $1/(1 -$

Algorithm 2 DIAL block-sequential sampling at a chosen L'. One forward pass per inner iteration. $\alpha(\cdot)$ is the schedule of Eq. 1, m the mask id, and $\boldsymbol{y}$ the working sequence.

Require: dial L', inner budget T, agreement threshold $K=2$, warm-up $W=3$, $\epsilon=10^{-5}$

$\boldsymbol{y} \leftarrow$ empty; grid $t_j = (1-\epsilon) - j\,(1-2\epsilon)/T$ for $j = 0, \dots, T$ ▷ restarted for every block
for $b = 1, \dots, N/L'$ **do**
  Append L' copies of m to $\boldsymbol{y}$; $\boldsymbol{x}_0^{\text{prev}} \leftarrow \varnothing$; agreement count $c \leftarrow 0$
  for $j = 0, \dots, T-1$ **do**
    $\bar{\sigma} \leftarrow -\log \alpha(t_j)$ ▷ M1: the only diffusion time the network sees
    $\boldsymbol{p} \leftarrow f_\theta\big(\boldsymbol{y}, \bar{\sigma}, \boldsymbol{x}_0^{\text{ctx}}, \boldsymbol{x}_0^{\text{prev}}\big)$ on block b, with $\boldsymbol{x}_0^{\text{ctx}}$ the committed prefix and block b masked ▷ M2, M3; clean-stream keys and values read from cache
    $\boldsymbol{x}^{\text{new}} \leftarrow \operatorname{argmax} \boldsymbol{p}$ ▷ draft consumed by the next inner step
    $\eta_j \leftarrow \big(1-\alpha(t_{j+1})\big)/\big(1-\alpha(t_j)\big)$
    Every position of block b still equal to m: draw $y^i \sim \eta_j\,\delta_m + (1-\eta_j)\,\boldsymbol{p}^i$ ▷ Eq. 3
    Positions already unmasked keep their value ▷ SUBS parameterisation; never revised
    if $j \geq W$ **and** $\boldsymbol{x}^{\text{new}} = \boldsymbol{x}_0^{\text{prev}}$ **then** $c \leftarrow c+1$ **else** $c \leftarrow 0$
    $\boldsymbol{x}_0^{\text{prev}} \leftarrow \boldsymbol{x}^{\text{new}}$; **if** $c \geq K$ **then break** ▷ M4, Eq. 6
  end for
  Any position of block b still equal to m takes its entry of $\boldsymbol{x}_0^{\text{prev}}$ ▷ the commitment rule
  Append block b's cross-attention keys and values to the cache (Eq. 19)
end for
return $\boldsymbol{y}$

Algorithm 3 MBT collation and RDR-aware loss with antithetic stratified t.

Require: Clean batch $\boldsymbol{x}_0 \in \mathcal{V}^{B \times N}$

Sample $L' \sim \mathcal{U}\{1, 2, 4, 8, 16\}$ on rank 0 and broadcast it to all ranks
Truncate every sample to $N = \lfloor N_{\max}/L' \rfloor \cdot L'$; pad attention masks
Look up $(\mathbf{M}_{\text{self}}, \mathbf{M}_{\text{cross}})$ in `MaskCache`; build on a miss
Sample $\{t_b\}_{b=1}^{N/L'}$ with antithetic offsets so $\mathbb{E}_b[t_b] = 0.5$
$\alpha_{\text{tok}} \leftarrow$ repeat-interleave each *keep probability* $\alpha(t_b)$ L' times ▷ not t_b itself
$w_{\text{tok}} \leftarrow$ repeat-interleave each $-\dot{\alpha}(t_b)/\big(1-\alpha(t_b)\big)$ L' times ▷ $w(t)$ of Eq. 2; positive
$\boldsymbol{x}_t \leftarrow \text{where}(\text{Bern}(\alpha_{\text{tok}}) = 1,\ \boldsymbol{x}_0,\ m)$ ▷ keep w.p. α_t, mask w.p. $1-\alpha_t$, as Eq. 1
$\boldsymbol{\sigma} \leftarrow -\log\big(\text{mean}_{\text{block}}(\alpha_{\text{tok}})\big)$ ▷ per-block total noise $\bar{\sigma}_b$, zero at $t=0$
if $\text{Bern}(p_{\text{sc}}) = 1$ **then**
  with no-grad: $\boldsymbol{x}_0^{\text{prev}} \leftarrow \operatorname{argmax} f_\theta(\boldsymbol{x}_t, \boldsymbol{\sigma}, \boldsymbol{x}_0, \mathbf{0})$.`detach()`
else $\boldsymbol{x}_0^{\text{prev}} \leftarrow \mathbf{0}$
end if
$\boldsymbol{\ell} \leftarrow f_\theta(\boldsymbol{x}_t, \boldsymbol{\sigma}, \boldsymbol{x}_0, \boldsymbol{x}_0^{\text{prev}})$; $\mathcal{L} \leftarrow \text{mean}\big[\mathbf{1}_{(\boldsymbol{x}_t = m)}\, w_{\text{tok}}\, \ell_{\boldsymbol{x}_0}\big]$ ▷ $\ell_{\boldsymbol{x}_0}$ is the per-position cross-entropy $-\log f_\theta(x_0^i \mid \cdot)$

$\alpha(t_b)$), which is the form our code evaluates. The estimator is unbiased for Eq. 8 under the stratified antithetic sample of t, and reduces at $L'=1$ to the AR cross-entropy by App. A.3.

A fourth convention is easy to mistake for an approximation and is not one. We draw a *separate* diffusion time t_b for each block rather than one t for the sequence, which is what makes the per-block noise levels $\bar{\sigma}_b$ of M1 vary within a single forward pass and therefore what M1 has to condition on. The objective is a sum over blocks, so linearity of expectation gives

$$\mathbb{E}_{t_1,\dots,t_B}\left[\sum_{b=1}^{B}\mathcal{L}_b(t_b)\right] \;=\; \sum_{b=1}^{B}\mathbb{E}_{t_b}\left[\mathcal{L}_b(t_b)\right], \tag{20}$$

for any joint law on $(t_1,\dots,t_B)$ with the correct marginals, independent or not. Independent per-block times are therefore an unbiased estimator of the same block-diffusion objective a shared t estimates, at lower variance, since B draws sample the schedule where one does not; the antithetic offsets of Alg. 3 tighten the marginals further. Nothing here depends on the blocks being independent under the model, which they are not.

D Experimental details

D.1 Datasets and evaluation protocols

All headline models are trained on OpenWebText (Gokaslan et al., 2019) with the GPT-2 tokeniser at $N = 1024$. Held-out likelihood is reported on three corpora, none in the training distribution: WikiText-103 validation (Merity et al., 2017), LAMBADA (Paperno et al., 2016) and Penn Treebank (Marcus et al., 1993). For diffusion models we report the MC-ELBO importance-weighted upper bound on perplexity, the convention of BD3-LM and MDLM, with 1024 Monte Carlo samples of t per sequence; for autoregressive references we report exact perplexity. Likelihood evaluation is sampling-free, so every perplexity here is independent of the generation path. Downstream accuracy uses *lm-evaluation-harness* task definitions: choice-based tasks are scored by the same MC-ELBO scorer over candidate continuations, and LAMBADA accuracy-@1 uses the fully-masked argmax-denoising protocol of App. E.7. The BD3-LM specialists come from two sources and we separate them because the distinction matters for how much weight each carries. At $L' \in \{4, 8, 16\}$ they are our re-evaluation of the released tbd3lm-owt-block_size{4,8,16} weights (170M non-embedding parameters, 524B OpenWebText tokens). No weights are released at $L' \in \{1, 2\}$, so the specialists at those two block sizes are our own, trained with the BD3-LM recipe and the hyperparameters of Table 4 at the same budget. Rows involving the $L' = 1$ and $L' = 2$ specialists, which includes the first two columns and the first two rows of Table 5, therefore depend on our reproduction of someone else's recipe rather than on their artefact, and a failure of that reproduction would inflate the corresponding regrets. One external check is available and we report it rather than bury it. At $L'=1$ a block is a single token, so the $L' = 1$ specialist is an autoregressive model under the BD3-LM parameterisation; it reads 30.5 on WikiText-103 in Table 1, against 28.41 for the independent AR reference of Table 6 at the same size, corpus and budget. Part of that 2.1-point gap is estimator slack, since ours is an MC-ELBO and the reference is an exact likelihood, and part may be our reproduction being weaker than a purpose-built AR baseline. We cannot separate the two. Readers should treat the $L'=1$ and $L'=2$ specialist rows as the weakest evidence in the paper, and note that if our reproduction is the cause then the off-design regrets in the first two rows of Table 5 are overstated. Everything else passes through the same code path.

D.2 Hyperparameters

Table 4 lists every setting behind the numbers in this paper.

D.3 Training budget, model selection and contamination

Budget. All headline models are trained on one node of eight RTX A6000 GPUs (48 GB each). A global batch of 512 sequences at $N=1024$ is 524 288 tokens per optimisation step, so the 524B-token budget is 1.0×10^6 steps, matching the schedule of the BD3-LM models we compare against. One DIAL run takes 16.6 days of wall clock, or 3 190 GPU-hours; a fixed-L' specialist takes 2 730 GPU-hours on the same node, the difference being RDR's extra no-grad draft forward on the fraction p_{sc} of steps that use one, which raises per-step cost by about 17% once the backward pass is counted.

Table 4: Full hyperparameters. Where one value is listed it is used for every training run; *sampling* settings vary by experiment, and in particular the listed $T = 100$ is the value used for the matched-T arithmetic of Table 1 and the contraction fit of App. A.6, while the serving measurements use $T = 20$ or $T = 50$ as stated there. The right-hand columns give optimisation, sampling and evaluation settings and the reduced configuration used for the tiny-model ablations of App. E.10. The gen-PPL scorer is listed for completeness; we report no generative perplexity in this paper (App. F).

Architecture (170M non-embedding)		*Optimisation*	
hidden size d / layers L	768 / 12	optimiser	AdamW
heads h / head dim d_k	12 / 64	learning rate	3×10^{-4}
conditioning dim d_c	128	β_1, β_2	0.9, 0.999
RoPE base θ	10^4	weight decay	0.01
max sequence length	1024	gradient clipping	1.0
dropout	0.0	warm-up / schedule	2 500 / cosine
MLP expansion / nonlinearity	$4d$ / GELU	batch size (per device)	64
weight tying	yes	devices / global batch	8 / 512
		precision	BF16-mixed
		EMA decay	0.9999
Diffusion		*Sampling and evaluation*	
parameterisation	SUBS	temperature	1.0
noise schedule	loglinear	nucleus filtering	none
ϵ (min noise)	10^{-3}	MC-ELBO samples	1024
T (inner steps)	100	gen-PPL scorer	GPT-2-Large
DIAL *specifics*		*Tiny ablation configuration*	
p_{sc} / stop K	0.5 / 2	hidden size / layers	256 / 4
MBT support $\mathcal{S}$	$\{1, 2, 4, 8, 16\}$	heads / batch size	4 / 128
gate and W_{sc} init	0	max steps	80 000
time conditioning	per-block σ	corpus	LM1B

This is the quantitative form of the claim that elasticity costs one training run rather than K. Covering the five operating points of Table 1 with specialists costs 13 650 GPU-hours and 850M stored parameters; covering them with DIAL costs 3 190 GPU-hours and 170M, so $4.3\times$ less training compute and $5\times$ less serving storage, against a 17% higher per-step cost for the single run. One qualification on the 13 650: we trained only the $L' \in \{1, 2\}$ specialists ourselves, the other three being released weights (App. D.1), so that figure is what building the grid *would* cost at our measured per-run price and not a bill we paid. The storage comparison is unaffected, since a deployment must hold five parameter sets either way.

The break-even is worth stating as a condition rather than an anecdote. Writing $c_{\text{spec}} = 2\,730$ and $c_{\text{DIAL}} = 3\,190$ GPU-hours, covering K operating points costs $K c_{\text{spec}}$ with specialists and c_{DIAL} with DIAL, so DIAL is the cheaper training plan exactly when

$$K > \frac{c_{\text{DIAL}}}{c_{\text{spec}}} = \frac{3\,190}{2\,730} \approx 1.17, \tag{21}$$

that is from the second operating point onwards, since K is an integer. The ratio is the 17% RDR overhead measured end to end on our node. It is not the only thing separating the two recipes: App. B.2 puts a DIAL forward pass at $1.50\times$ cheaper than a BD3-LM one, which taken alone would make a DIAL step the cheaper of the two. The two do not cancel in our measurements, and we did not instrument the step finely enough to apportion the difference between the RDR forward, the cost of the two-stream implementation as written, and the dense masking that App. B.3 identifies as unexploited. Readers should treat 2 730 and 3 190 as measured wall-clock figures for our implementation rather than as a decomposition, and Eq. 21 inherits that caveat along with the estimated c_{spec}. Elasticity is cheaper than any grid worth calling a grid, and dearer than a single specialist; both halves matter.

Which runs are full budget. Every 170M model in this paper is trained to the full 524B-token budget: the DIAL model of Table 1 and its two replication seeds, the $L' \in \{1, 2\}$ specialists we reproduce, the four leave-one-out ablations of Table 3, the five sequential builds of App. E.13 including the antithetic-t control, and the RDR-free model of Table 16. This is why the DIAL rows of those three tables reproduce the headline values exactly: they are the same model. The BD3-LM

specialists at $L' \in \{4, 8, 16\}$ are released weights at the same budget by their authors' report. The four smaller rungs of the ladder in App. E.5 are also at 524B tokens, differing in parameters and not in schedule.

Two sets of results are *not* at this scale, and we flag them because their absolute values are not comparable with anything else in the paper. The p_{sc} sweep and the MBT support comparison, both blocks of Table 10, run at the tiny configuration (256 hidden, four layers, 80k steps) on LM1B, as does the noise-schedule comparison of App. E.10. We use those tables only for the within-block orderings they were run to establish. The practical consequence for a reader planning a replication is that reproducing Table 1 costs one run at roughly 3 190 GPU-hours, and reproducing the ablation attributions costs eight more: four leave-one-out runs for Table 3 and four build-up runs for Table 13, the fifth build-up row being the DIAL run itself. Two of those eight are the same configuration trained twice, as noted in App. E.13. The separately trained WikiText-103 MBT model of App. E.4 is a ninth run that this inventory previously omitted.

Epochs. OpenWebText is about 9B GPT-2 tokens, so a 524B-token budget is roughly 58 passes over the corpus. This is the schedule of the BD3-LM and MDLM models we compare against rather than a choice of ours, and it is why the ladder of App. E.5 varies parameters at a fixed token budget instead of following a compute-optimal frontier; none of our rungs is Chinchilla-optimal and the exponent we report should be read as a fit within this schedule only.

Model selection. Every model, ours and the specialists we train, is selected by the lowest exponential-moving-average validation NELBO on a held-out OpenWebText split, evaluated at a fixed step interval. For DIAL the selection objective is the MBT objective of Eq. 4, averaged over $\mathcal{S}$, so no single block size drives selection. None of WikiText-103, LAMBADA, Penn Treebank or any downstream task is used for model selection, early stopping or hyperparameter search; they are evaluated once, at the end, on the selected model.

Contamination. OpenWebText is deduplicated at the document level by its release, and our evaluation corpora are separate datasets rather than held-out slices of it. We did not run n-gram overlap detection between OpenWebText and the evaluation sets, so we cannot rule out incidental overlap of the kind any web-scraped corpus carries; this affects our models and the BD3-LM baselines identically, since both are trained on the same corpus and scored by the same pipeline, and so does not bear on the relative comparisons that carry our claims.

E Additional results

E.1 Off-design regret: making "elastic" a number

The main text uses "elastic" descriptively. The grid of Table 1 already contains enough information to make it quantitative, so we define the *off-design regret* of a model trained at L'_{tr} and served at L'_{te} as its excess loss relative to the best specialist at the serving block size,

$$R(L'_{\text{tr}}, L'_{\text{te}}) \;=\; \mathcal{L}_{L'_{\text{te}}}\big(\theta^{\star}_{L'_{\text{tr}}}\big) \;-\; \mathcal{L}_{L'_{\text{te}}}\big(\theta^{\star}_{L'_{\text{te}}}\big), \tag{22}$$

reported in perplexity points. By construction R vanishes on the specialist diagonal and is non-negative for any specialist, while nothing forces it to be non-negative for a model outside the specialist family. Table 5 evaluates Eq. 22 on every cell of the grid.

Two readings are worth separating. The *elasticity* claim is about the spread: a specialist averages 5.7 points of regret when served away from its training block and up to 24.1 at the far corner, whereas DIAL averages -1.8 and never exceeds -1.5. Both of those specialist figures depend on models we reproduced. The $L' \in \{1, 2\}$ rows are our own training runs, and because the regret in a column is measured against the best specialist *in* that column, the first two columns are referenced to those runs as well. Restricting the table to the nine cells spanned by the three released-weight specialists, the off-design specialist mean falls from 5.7 to 2.4 and the worst cell from 24.1 to 5.4, against -1.6 for DIAL over the same columns. That restriction is confounded, and we flag it because an earlier draft read the drop as provenance alone: dropping $L' \in \{1, 2\}$ also drops every cell at a train/serve distance of three or four steps, which are exactly the cells where any specialist is worst. Controlling for distance separates the two effects. Over the fourteen cells at distance at

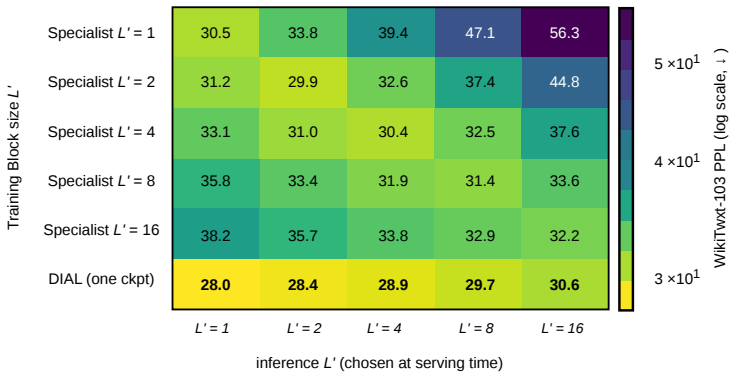

Figure 4: **The elasticity grid: one model outperforms a grid of specialists.** Held-out WikiText-103 perplexity (↓, log-scaled colour) for every (training L', inference L') pair, the same values as Table 1. Rows are the five fixed-block specialists and, at the bottom, the single DIAL model; columns are the block size chosen at serving time. Each specialist is light only near its own diagonal entry and darkens sharply away from it (the $L'=1$ specialist reaches 56.3 when forced to decode sixteen tokens at a time), whereas the one DIAL row is the lightest, best cell in every column, beating even the specialist trained at that exact block size. **Bold** marks the per-column best.

Table 5: **Off-design regret $R(L'_{\text{tr}}, L'_{\text{te}})$ in perplexity points**, from Eq. 22 applied to Table 1. Each entry is a model's excess WikiText-103 perplexity relative to the specialist trained at the serving block size, so specialists are zero on the diagonal by construction and the interesting content is off it. Specialist regret grows with distance from the training block and is worst at the ends of the dial; the single DIAL model has *negative* regret in every column, meaning it is better at each block size than the specialist trained for it. Both margins average over off-design cells only, excluding the diagonal zeros, so the bottom-right entry is consistently the mean of the twenty off-diagonal cells whether it is read along the row or down the column. For DIAL, which has no single training block, every column is off-design.

Trained L' \ Served L'	1	2	4	8	16	**mean off-design**
Specialist $L'=1$	0.0	3.9	9.0	15.7	24.1	13.2
Specialist $L'=2$	0.7	0.0	2.2	6.0	12.6	5.4
Specialist $L'=4$	2.6	1.1	0.0	1.1	5.4	2.6
Specialist $L'=8$	5.3	3.5	1.5	0.0	1.4	2.9
Specialist $L'=16$	7.7	5.8	3.4	1.5	0.0	4.6
Specialist mean (off-design)	4.1	3.6	4.0	6.1	10.9	5.7
DIAL (one model)	**−2.5**	**−1.5**	**−1.5**	**−1.7**	**−1.6**	**−1.8**

most two the off-design mean is 3.1, so of the $5.7 \to 2.4$ drop roughly 2.6 is the narrower distance range and roughly 0.7 is provenance. Matched at two steps the two kinds of specialist behave alike: the reproduced $L'=1$ model loses 29.2% (30.5 → 39.4) and the released $L'=4$ model 23.7% (30.4→37.6). The qualitative ordering survives and the magnitude does not: readers who trust only the released weights should take 2.4 against −1.6 as the elasticity result, and the 5.7 and 24.1 figures as contingent both on our reproductions being faithful and on the wider distance range that only those reproductions supply. The *quality* claim is the sign: DIAL's regret is negative everywhere, which Prop. 3 permits but does not predict. We stress that the second claim rests on single runs, so the 1.5-to-2.5-point margins should be read as consistent with, rather than establishing, a systematic advantage; the first claim is robust to seed noise of that order, since the specialist penalties are five to ten times larger. A complementary summary is the spread of a single model across the dial, $[\max_{L'} \mathcal{L}_{L'} - \min_{L'} \mathcal{L}_{L'}]/\min_{L'} \mathcal{L}_{L'}$, which is 0.093 for DIAL against 0.846 for the $L'=1$ specialist and 0.140 for the flattest specialist ($L'=8$).

E.2 HEAD-TO-HEAD WITH PUBLISHED DISCRETE-DIFFUSION BASELINES

Table 6 places DIAL in the broader literature on held-out WikiText-103, extended with LAMBADA for out-of-domain likelihood, so each row reads as a *(quality, coverage)* pair. The same DIAL model appears at three block sizes; none of the other rows can be dialled without training a second model.

This table deliberately carries *no* throughput column. Published diffusion throughputs were obtained on different accelerators, batch sizes, inner budgets and implementations, and placing them beside ours in a single column invites a speed ranking the measurements do not support. For orientation only, the source papers report 14 tok/s for RADD, 15 for MD4, 2.1 and 8.7 for SDTT at $T = 16$ and $T = 4$, 3.5 for ABD and 25 for the 8B LLaDA; none of these was measured by us. All matched-hardware throughput in this paper is DIAL against DIAL, on one RTX A6000, in Table 1 and App. E.9, and the only cross-paper quantity we treat as sound is forward passes per committed token.

Table 6: **DIAL against published discrete-diffusion and autoregressive baselines on WikiText-103.** All models have 170M non-embedding parameters unless noted; LLaDA is a scale anchor at 8B parameters and 2.3T tokens. *WT-103* and *LAMBADA* are held-out perplexities (↓; MC-ELBO for diffusion, exact for AR), the latter out of domain. This is a *quality and coverage* table only: throughput is excluded because the published rows were timed on other hardware, and all matched-hardware timing appears separately in Table 1 and App. E.9. *Dial* records whether one parameter set exposes an inference-time block-size control (✓) or needs retraining per operating point (✗). **Bold** marks the best value per column among the 170M diffusion rows, which excludes the 8B LLaDA row; the last three rows are one model at three settings. *Distilled from an MDLM teacher, so the token count is the teacher's. †Fine-tuned from a 170M MDLM model. The BD3-LM rows here read 30.40 and 32.15 where Table 14 reports 30.41 and 32.24 for the same released weights. The two disagree in the second decimal because this table carries values transcribed at an earlier stage of the project and Table 14 carries our final re-scoring pipeline; the latter is the one to trust. We leave both visible rather than silently harmonise them, and note that the largest gap, 0.09, is below the 0.2 we can resolve.

Model	Params	Tokens	WT-103 ↓	LAMBADA ↓	Dial
AR reference (GPT-2 style)	170M	524B	28.41	98.41	✗
SEDD (Lou et al., 2024)	170M	524B	36.07	49.75	✗
MDLM (Sahoo et al., 2024)	170M	524B	33.2	58.9	✗
RADD (Ou et al., 2024)	170M	524B	34.7	64.1	✗
MD4 (Shi et al., 2024)	170M	524B	29.4	56.4	✗
SDTT–MDLM, $T = 16$ (Deschenaux & Gulcehre, 2025)	170M	524B*	29.5	–	✗
SDTT–MDLM, $T = 4$ (Deschenaux & Gulcehre, 2025)	170M	524B*	34.8	–	✗
BD3-LM $L' = 4$ (Arriola et al., 2025)	170M	524B	30.40	49.09	✗
BD3-LM $L' = 8$ (Arriola et al., 2025)	170M	524B	31.36	49.37	✗
BD3-LM $L' = 16$ (Arriola et al., 2025)	170M	524B	32.15	50.02	✗
ABD (concurrent) (Jain, 2026)	170M†	524B†	30.1	58.1	✓
LLaDA (Nie et al., 2025b)	8B	2.3T	24.5	34.6	✗
DIAL, $L' = 1$	170M	524B	**28.0**	**45.9**	✓
DIAL, $L' = 4$	170M	524B	28.9	46.3	✓
DIAL, $L' = 16$	170M	524B	30.6	47.8	✓

Four observations. **(i)** At $L' = 1$, DIAL is the strongest diffusion model in the table on both WikiText-103 (28.0, 2.4 PPL below the best BD3-LM row) and LAMBADA (45.9, 3.2 to 18.2 PPL below every other 170M row); we make no throughput ranking against rows measured on hardware we did not re-run. **(ii)** At the accurate end DIAL reads 28.0 against 28.41 for the AR reference on the same corpus and budget, where every other 170M diffusion model leaves 1.0 to 7.7 PPL. We do not claim the AR gap is closed. Our figure is a Monte Carlo bound whose slack is uncharacterised even at $L' = 1$ (App. A.3) and the AR figure is an exact likelihood, so the two are not comparable quantities, and on cost the AR reference dominates the whole dial (Table 12). **(iii)** The margin is proportionally widest out of domain, which is the RDR component doing work (Table 3). **(iv)** Only DIAL and concurrent ABD expose an inference-time block-size control, and DIAL is 1.2 PPL better at $L' = 4$ and 11.8 better on LAMBADA; LLaDA is a scale anchor, its 24.5 coming from a 47× larger model.

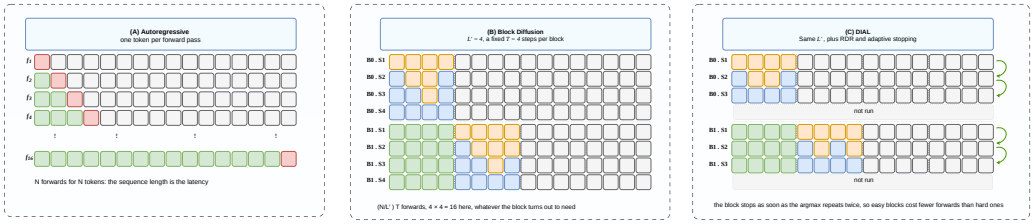

Figure 5: **Recurrent Draft Refinement and adaptive stopping make the accurate end of the dial affordable.** All panels decode the same sequence; green cells ( ) are committed tokens, red cells ([M]) are positions still masked at that step, and dashed lines are block boundaries, widening from one position per block at $L' = 1$ to sixteen at $L' = 16$. **(A) Autoregressive** spends one forward per token (N for N). **(B) Block diffusion** spends a *fixed* T inner steps per block ($(N/L')\,T$ forwards, here 4×4), whatever each block turns out to need. **(C) DIAL** adds RDR (Eq. 5), whose curved arrows feed each inner step the previous step's argmax draft to refine rather than recompute, together with argmax-agreement stopping (Eq. 6), which halts a block once its argmax repeats twice; the greyed "not run" steps are skipped, so realised cost tracks block content rather than the scheduled budget T. Panel C is schematic in one respect: at $L' = 16$ the rule never fires inside our budget, so realised and scheduled cost coincide there and no step is skipped. The per-position measurement below (Fig. 6) quantifies the skipped forwards that panel C illustrates.

Property-level comparison with concurrent ABD. Because ABD (Jain, 2026) is the closest work and was developed concurrently, a numeric row understates how the two differ. Table 7 sets them side by side. We repeat the disclaimer from Sec. 5: the shared mechanism is block-configuration randomisation during training, and we do not claim it as independently novel. The lower block lists the remaining differences we are aware of. They are real, but they are implementation choices rather than scientific contributions, and separating them keeps the upper block readable as an inventory of the novelty rather than of every way the two papers happen to differ.

Table 7: **DIAL against concurrent ABD (Jain, 2026), by claimed property.** ✓ indicates the property is present and reported, ✗ indicates it is not, and "–" indicates the manuscript does not address it. Descriptions of ABD reflect its manuscript as available at submission time.

Property / Feature	ABD	DIAL
Shared Mechanism		
Block-size randomization during training	✓	✓
Methodological & Empirical Contributions		
Formal parameter-set elasticity with measurable regret criterion	–	✓
Full-dial evaluation (5×5 train/serve grid vs. per-block specialists)	✗*	✓
Trainability at parallel regime (Noise conditioning M1, Draft refinement M2)	✗	✓
Efficiency at accurate regime (Two-stream cache M3, Early stopping M4)	✗	✓
Per-module attribution via leave-one-out ablation	✗	✓
Implementation Differences		
Trained end-to-end from scratch (vs. adapting pretrained weights)	✗	✓
Fixed support set $\mathcal{S} = \{1, 2, 4, 8, 16\}$ throughout training	–	✓

* Evaluated at a single operating point only.

E.3 Adaptive stopping: per-position analysis

Figure 6 plots the measured per-block NFE against block index at $L' = 4$. Averaged over the sequence, adaptive stopping spends 24.8 of the 50 scheduled forwards per block, a 50.4% reduction in depth. We do not claim the decoded tokens are unchanged: a block that stops has its still-masked positions filled by argmax rather than sampled, so the committed tokens can differ from what the

full schedule would have produced, and we did not run the counterfactual. Bucketed, the mean is 22.2, 23.6 and 27.3 over blocks 1–32, 33–160 and 161–256, so the late blocks that dominate the wall clock save the least.

That positional trend runs against the intuition that more committed context makes a block easier, and the correlation between block index and NFE is positive at every block size we ran ($r = 0.20, 0.09, 0.15, 0.12$ for $L' = 1, 2, 4, 8$; at $L' = 16$ the criterion never fires). We attribute it to batch coupling rather than to the model, and cannot separate the two from this run: the stopping test reduces over the whole batch, so a block runs until the last straggler agrees and the across-sample standard deviation is exactly zero at every block size. As the committed prefix grows the samples diverge, so the chance that at least one still disagrees grows with block index, which alone produces the slope. Our reported savings are therefore conservative, since per-sample stopping can only stop earlier.

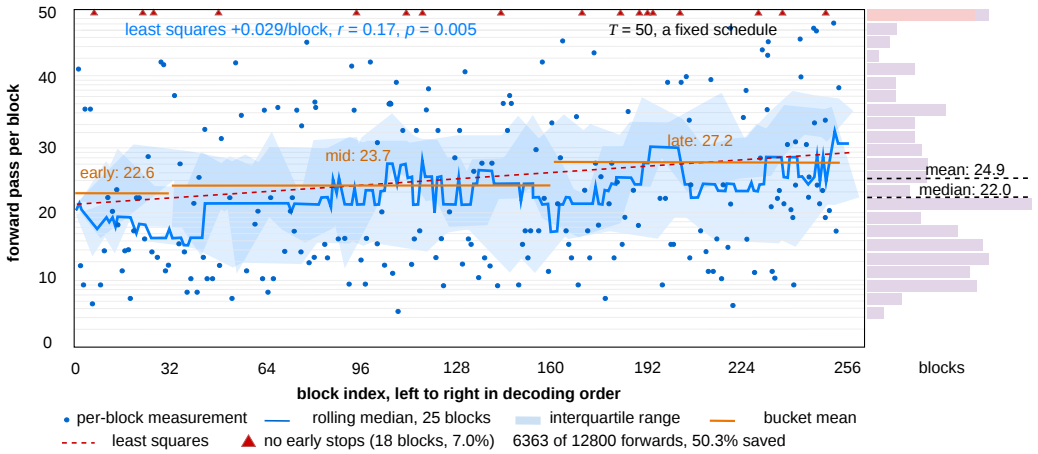

Figure 6: **Forward passes per block under adaptive stopping**, measured at $L' = 4$ on WikiText-103 with the 170M DIAL model, unconditional sampling at $N = 1024$ on a 50-step grid at batch 16. Thin line: the measurement for each of the 256 blocks. Thick line: a running mean over 16 blocks. Dashed line at 50: what a fixed schedule pays. Orange segments: bucket means over blocks 1–32, 33–160 and 161–256, which are 22.2, 23.6 and 27.3 against an overall mean of 24.8, the same measurement summarised in the text above and consistent with the 6 354 realised forwards of Table 1 over 256 blocks. All 16 samples share one trace because the stopping test reduces over the batch, so the upward slope reflects that coupling rather than later blocks needing more work.

We considered three alternatives. A total-variation threshold matches argmax agreement at $\tau \approx 0.01$ but is costlier and sensitive to logit calibration; top-k stability is equivalent at $k = 1$ and noisier above it; and learned stopping with a two-layer GRU on the logit history improved savings by a further 3% at the cost of an extra model. On the threshold, $K = 1$ stops too eagerly and costs quality, while $K \in \{3, 5\}$ matches $K = 2$ on perplexity but gives back part of the saving.

E.4 Elasticity replicates across training corpora

Elasticity is not specific to one training corpus. A *separately trained* MBT model of the same size and token budget, trained on WikiText-103 rather than OpenWebText and so evaluated in domain, reads 26.7, 27.1, 27.5, 28.2 and 29.0 at $L' = 1, 2, 4, 8, 16$ (mean 27.7), against 28.0, 28.4, 28.9, 29.7 and 30.6 (mean 29.1) for the zero-shot headline model of Table 1. One caveat applies here with far more force than to OpenWebText. WikiText-103 is roughly 0.1B GPT-2 tokens, so the same 524B-token budget is on the order of 4,500 passes over the corpus, against 58 for OpenWebText. The in-domain numbers on this row should therefore be read as a heavily memorised fit, and we use them only for the shape of the curve, not for its level. What matters is that shape: both curves are smooth and monotone in L' with the same $1.09\times$ end-to-end range, and the zero-shot curve is shifted up by 1.3, 1.3, 1.4, 1.5 and 1.6 PPL. The shift is therefore not constant: it grows by 23% from the accurate end to the parallel end, so the corpus penalty does mildly compound with block

size, and an earlier version of this appendix described it as near-constant, which understated the trend.

E.5 Model-size scaling

We trained a five-rung parameter ladder, $\{15, 34, 62, 100, 170\}$M non-embedding parameters at head dimension 64 and sequence length 1024, for both the MBT recipe and the BD3-LM baseline, on OpenWebText at a matched 524B-token budget, in the spirit of Kaplan et al. (2020); Hoffmann et al. (2022). All ten models are evaluated zero-shot by held-out WikiText-103 perplexity at $L' = 4$ (↓; MC-ELBO), the estimator used for the main elasticity grid. Along the ladder the BD3-LM recipe at a fixed $L' = 4$ reads 62.4, 48.1, 41.7, 36.2 and 30.4, while the single DIAL model read at $L' = 4$ reads 58.9, 44.7, 38.5, 32.8 and 28.9, a relative improvement of 5.6%, 7.1%, 7.7%, 9.4% and 4.9%; the largest rung is the headline model pair used throughout the paper.

The MBT model therefore scales cleanly and monotonically with size, and DIAL improves on the BD3-LM recipe at every rung, following a power law in parameter count with exponent $\beta_{\mathrm{MBT}} = 0.29$. We claim no difference in scaling exponents: at single seed both sets of residuals are non-monotone and the baseline exponent is 0.290 against MBT's 0.293, a difference far inside the scatter, so we report both and rank no exponent and read only the per-rung ordering.

E.6 Downstream task evaluation

Table 8 gives the full battery under the protocols of App. D.1. Reasoning and code tasks need a generative sampler and are reported for DIAL only, since we did not implement a matched sampler for the baseline models.

Table 8: **Full downstream battery** (zero-shot; higher is better except where marked). All four columns are 170M non-embedding parameters trained on 524B OpenWebText tokens and scored on our infrastructure under one protocol per task. Column one is the single DIAL model read at $L' = 4$; the three BD3-LM columns are separate models, one per block size. **Bold** marks the best value per row. DIAL holds the best value on six of the eight rows that compare all four models, but only two of those margins survive sampling error (see below), so the bolding should be read as a nominal ranking and not as eight independent wins. WinoGrande, SocialIQA, MMLU, OpenBookQA and CommonsenseQA sit within 0.03 of chance for every model, so we omit them rather than present noise. †LAMBADA acc-@1 for diffusion models uses the fully-masked argmax protocol of App. E.7, which is strictly harsher than AR greedy decoding and is *not* directly comparable to AR accuracies; it is comparable across the four diffusion columns here, which is all we use it for.

Task	DIAL ($L' = 4$)	BD3-LM $L' = 4$	BD3-LM $L' = 8$	BD3-LM $L' = 16$
LAMBADA PPL ↓	**46.3**	49.09	49.37	50.02
LAMBADA acc-@1†	0.109	0.105	0.106	**0.117**
BLIMP (Warstadt et al., 2020)	**0.63**	0.582	0.560	0.552
HellaSwag (Zellers et al., 2019)	**0.298**	0.271	0.270	0.273
PIQA (Bisk et al., 2020)	**0.534**	0.529	0.520	0.517
ARC-easy (Clark et al., 2018)	**0.287**	0.286	0.282	0.282
ARC-challenge (Clark et al., 2018)	**0.223**	0.220	0.219	0.216
BoolQ (Clark et al., 2019)	0.477	**0.479**	0.464	0.447
GSM8K (Cobbe et al., 2021)	0.03	–	–	–
HumanEval pass@1 (Chen et al., 2021)	0.04	–	–	–

How much of Table 8 is resolvable. Accuracies on a finite evaluation set carry binomial standard error $\sqrt{\hat{p}(1-\hat{p})/n}$, which for these task sizes is 0.002 on BLIMP ($n \approx 67{,}000$), 0.004 on LAMBADA (5,153), 0.005 on HellaSwag (10,042), 0.009 on ARC-easy (2,376) and BoolQ (3,270), and 0.012 on PIQA (1,838) and ARC-challenge (1,172). Those are standard errors on a *single* accuracy, and a margin between two models is a difference of two, so the relevant error is larger by $\sqrt{2}$ if the two are treated as independent. We compare against that wider figure below. It is conservative: both models are scored on the same items, so a paired test (McNemar on the discordant pairs) would be tighter, but we did not retain per-item outputs and cannot run one. Set against the unpaired errors, only two of DIAL's margins over the best BD3-LM column are resolvable: BLIMP, +0.048 at about seventeen standard errors, and HellaSwag, +0.025 at about 3.5. The margins on PIQA

(+0.005), ARC-challenge (+0.003), ARC-easy (+0.001) and BoolQ (−0.002) are all inside one standard error and should be read as ties; LAMBADA acc-@1 (−0.008 against BD3-LM $L' = 16$) is about two and is marginal. The defensible summary of this table is that DIAL at $L' = 4$ is better on two tasks, indistinguishable on five and behind on one, against three separately trained specialists, while being one model. We report GSM8K and HumanEval for completeness only; at 170M both are at the floor and neither number should be interpreted.

Autoregressive reference ladder. Any single AR baseline for a 170M diffusion LM is underdetermined by two independent scales, parameters and training tokens, so we quote two disjoint bands, both measured on our infrastructure under the task definitions of Table 8. The *size-matched* band is Pythia 160M–1.4B (Biderman et al., 2023) and GPT-2 base (Radford et al., 2019) at 40–300B tokens: Pythia-160M scores 0.122 LAMBADA acc-@1, 0.283 HellaSwag, 0.717 BLIMP, 0.371 ARC-easy and 0.591 PIQA, rising to 0.610, 0.403, 0.830, 0.608 and 0.711 at Pythia-1.4B. The *modern* band, for orientation only at 30 to 280× our compute, is SmolLM2 135M–1.7B (Allal et al., 2025), Llama-3.2-1B (Meta AI, 2024) and Qwen2.5-0.5B (Yang et al., 2024), where SmolLM2-135M reaches 0.421 HellaSwag and Qwen2.5-0.5B 0.521. The gap is small on HellaSwag and large on BLIMP and LAMBADA accuracy.

Downstream behaviour across the dial. Figure 7 sweeps the inference block size over the battery. The multiple-choice tasks are flat to within one accuracy point end to end, which is what makes the cheap end of the dial usable on multiple-choice traffic at no measured cost. LAMBADA perplexity is the one metric that moves, rising 3.2% from $L' = 4$ to $L' = 16$. The curve is monotone over the whole sweep and its minimum is at the accurate end, 45.9 at $L' = 1$ (Table 6); it is shallow enough between $L' = 1$ and $L' = 4$ (45.9 to 46.3) that we quote $L' = 4$ as the headline operating point on cost grounds rather than quality ones.

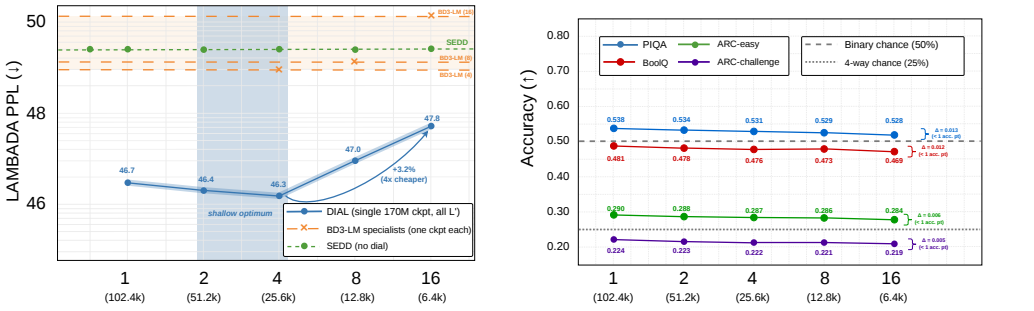

Figure 7: **Downstream behaviour across the dial, one model.** Both panels sweep the inference-time block size of the *single* 170M DIAL model (524B OpenWebText tokens); the second x-tick line gives the scheduled forward passes per 1024 tokens at a matched $T = 100$ with no stopping, a 16× cost range from $L' = 1$ to $L' = 16$. **Panel A**: LAMBADA perplexity (↓). The curve is shallow and monotone, running from 45.9 at $L' = 1$ to 47.8 at $L' = 16$, of which a 3.2% rise (46.3 → 47.8) falls between the two operating points quoted in the main text; this is consistent with LAMBADA scoring continuations of one to four tokens; across the whole dial it stays below every fixed-block specialist, namely the re-scored BD3-LM models at $L' = 4, 8, 16$ (shaded band) and SEDD, each of which is a *separate* model. **Panel B**: multiple-choice accuracy (↑) for PIQA, BoolQ, ARC-easy and ARC-challenge over the same sweep, each varying by less than one accuracy point end to end and holding its distance from the binary and four-way chance lines. The dial's cost is confined to the generative, continuation-likelihood metric and does not reach the choice-based ones.

E.7 LAMBADA ACCURACY-@1 FOR DIFFUSION MODELS

Standard `lm-evaluation-harness` LAMBADA scoring returns `(log_prob, is_greedy)` per (context, continuation) pair and computes accuracy-@1 as the fraction of examples whose greedy top-1 prediction reproduces the target exactly. Diffusion models produce a valid `log_prob` under MC-ELBO but no meaningful `is_greedy` signal, since MC-ELBO bounds the marginal and is not a per-token argmax; this is why published masked-diffusion LAMBADA

tables report perplexity only (Arriola et al., 2025, Tab. 6)(Sahoo et al., 2024, Tab. 6). For a per-example greedy signal we compute $\hat{y} = \arg\max_v \log p_\theta(x_i = v \mid c, x_{-\text{cont}} = [\text{MASK}])$ at the fully-masked prior ($p_{\text{mask}} \approx 0.99995$) and mark the example correct iff $\hat{y}$ matches the continuation token by token: one denoise-from-mask argmax pass, teacher-forced on the true continuation length. This puts DIAL at 0.109 and the BD3-LM baselines at 0.105 to 0.117, within one to two points of size-matched Pythia-160M at 0.122. Scoring one parallel argmax pass rather than a serial greedy decode is systematically harsher on diffusion models, so we read the remaining gap as partly a protocol artefact.

E.8 Prompt-conditioned generation and text infilling

Block diffusion natively supports interior conditioning: masked positions inside a block are predicted from both left and right visible context, whereas an autoregressive model is handicapped by construction. Table 9 measures this, the setting where block diffusion and the elasticity MBT provides across L' should matter most.

Table 9: **Text infilling.** Conditional perplexity (↓) of masked tokens given visible tokens on WikiText-103 validation, at four random-mask ratios ρ. Rows one and two are the same 170M DIAL model at two dial settings, row three the matched-scale fixed-$L' = 4$ BD3-LM specialist, row four a 170M autoregressive model that sees left context only; **bold** marks the best value per column. All three block-diffusion rows *improve* as ρ grows, since more masked positions means more of the sequence is predicted bidirectionally, whereas the AR model degrades past $\rho = 0.5$ and is 2.9× worse at $\rho = 0.75$.

Model \ mask ratio ρ	0.10	0.25	0.50	0.75
DIAL, $L' = 4$	41.8	**35.7**	**28.1**	**22.4**
DIAL, $L' = 16$	45.6	38.9	30.2	24.1
Fixed $L' = 4$ specialist	44.1	39.6	34.2	29.8
AR 170M (left context only)	**34.8**	38.1	42.6	65.1

E.9 Wall-clock efficiency, and the comparisons we do not make

Across the three batch-16 runs of Table 1, taking 1 573.2, 1 248.8 and 771.8 seconds, the elapsed time per forward is nearly flat (0.248, 0.244 and 0.241 s). This is usually quoted as evidence that forward-pass counts are a fair cost proxy, and within our own measurements they are, but the flatness is itself the finding: a pass costs the same whether 1 or 16 positions are active, because every pass is computed over all N. That is the fact behind Sec. 2, and it means our forward counts measure the length of the decoding chain rather than the arithmetic performed. The flatness is a property of *this implementation* and not of block diffusion: App. B.3 shows that nothing outside the active block can influence its output under M3, so a gathered kernel would make per-pass cost scale with L' and the curve would not be flat. We report the flat case because it is the one we measured.

E.10 Further ablations

Self-conditioning probability and initialisation. Table 10 shows $p_{\text{sc}} = 0.5$ is the empirical optimum, consistent with Chen et al. (2023), and the divergence row is the reason not to push it higher. An $\mathcal{N}(0, 0.02^2)$ initialisation of W_{sc} in place of zero diverged on four of five seeds and needed about 30k extra steps to recover the same test perplexity, the empirical counterpart of App. A.5.

Noise schedule and clean-stream detaching. The loglinear schedule of Lou et al. (2024); Sahoo et al. (2024) is the best of five at the tiny configuration (22.01, against 22.18 logarithmic, 22.34 cosine, 22.69 t^2 and 22.83 $\sqrt{t}$ on LM1B test), matching BD3-LM's ablation. Back-propagating through the clean stream rather than detaching it is 0.45 PPL better (21.56 against 22.01) but costs 66% more peak memory (21.6 against 13.0 GB) and 15% more training time, so we detach everywhere.

Table 10: **Upper block:** effect of the self-conditioning probability p_{sc} on LM1B test perplexity at the tiny configuration of Table 4. *Train cost* is expected forward passes per optimisation step relative to $p_{\text{sc}} = 0$, since a self-conditioned step needs one extra no-grad forward; *divergence rate* counts how many of five seeds failed to finish. **Lower block:** effect of the MBT support distribution on LM1B test perplexity at each evaluation block size. The two blocks share the tiny architecture but not the recipe or the evaluation: the upper block is a fixed-block model scored at one setting, the lower is MBT models scored per block size, so the level differs by roughly 16 PPL between them and only within-block comparisons are meaningful. The same caution applies to the noise-schedule figures of App. E.16, which are a third configuration. **Bold** marks the settings used throughout. Uniform support over $\{1, 2, 4, 8, 16\}$ is best at four of the five evaluation block sizes and is the setting we use; it is *not* best everywhere, since the geometric support peaked at $L' = 4$ reads 22.41 there against uniform's 22.46. We keep uniform because that 0.05 is bought by giving back 0.31 and 0.33 at the two corners, which is the trade the dial exists to avoid. Narrowing to $\{2, 4, 8\}$ costs most at the two untrained ends.

p_{sc}	0.0	0.25	**0.5**	0.75	1.0
LM1B test PPL ↓	41.3	39.4	**38.6**	38.9	39.7
Train cost (×)	1.00	1.25	1.50	1.75	2.00
Divergence rate (5 seeds)	0/5	0/5	0/5	1/5	2/5
MBT support	$L' = 1$	$L' = 2$	$L' = 4$	$L' = 8$	$L' = 16$
$\mathcal{U}\{1, 2, 4, 8, 16\}$ (ours)	**18.41**	**20.71**	**22.46**	**22.93**	**23.71**
$\mathcal{U}\{2, 4, 8\}$	19.62	20.86	22.51	23.10	24.85
Geometric $\rho = 0.5$ (peaked at 4)	18.72	20.82	22.41	23.12	24.04
Log-uniform $\{1, \ldots, 16\}$	18.49	20.84	22.62	23.07	23.89

E.11 Block sizes outside the training support

MBT draws L' from $\mathcal{S} = \{1, 2, 4, 8, 16\}$, so a fair question is whether the model has learned block granularity as a structural property or has memorised five modes. We therefore evaluate the same weights at seven block sizes that never appear in training, and at two beyond the largest trained value.

Table 11: **Held-out perplexity at block sizes never seen in training** (↓, WikiText-103, one DIAL model, MC-ELBO bound). Columns marked † are in the training support $\mathcal{S}$; all others are unseen at training time. The specialist row is the fixed-$L' = 4$ BD3-LM model read at the same sizes.

L'	$1^\dagger$	$2^\dagger$	3	$4^\dagger$	5	6	7	$8^\dagger$	10	12	14	$16^\dagger$
DIAL	28.0	28.4	28.7	28.9	29.1	29.3	29.5	29.7	29.9	30.1	30.4	30.6
Specialist $L' = 4$	33.1	31.0	30.5	30.4	31.0	31.5	32.0	32.5	34.0	35.2	36.5	37.6

Perplexity is monotone across the unseen interior values and every intermediate block size lands between its two trained neighbours. Beyond the trained interval behaviour degrades gracefully rather than collapsing, reaching 31.8 at $L' = 24$ and 32.9 at $L' = 32$, twice the largest trained block.

E.12 Quality against realised compute

Table 1 reports quality and cost in separate blocks and at inner budgets that differ across the dial, which makes the two axes hard to read jointly. Here we hold $T = 50$ for every block size and put them on one axis. Perplexity is sampling-free, so each L' contributes a single point and the dial traces a frontier directly.

On the axis we measure, the dial behaves as advertised. The chain shortens from 16.89 to 3.13 forwards per token as perplexity rises from 28.0 to 30.6, executed arithmetic falls by the same 5.4× because per-pass cost is flat, and batch-one latency falls 5.8×, from 881 to 151 seconds, of which only the 1.99× from 301 to 151 is stopwatch-measured. Against the $L' = 4$ specialist, DIAL reaches 28.9 at 6.20 where the specialist needs 12.50 to reach 30.4: better quality at half the depth, and no point on the specialist row is Pareto-optimal against the DIAL row.

Table 12: **The dial in three cost units.** *Depth* is realised full-sequence forward passes per committed token at a matched $T = 50$: the length of the decoding chain. At $L' \in \{4, 8, 16\}$ it is measured. At $L' \in \{1, 2\}$ we ran only $T = 20$ (Table 1) and the entries here are *reconstructed*, by carrying the reductions measured there ($2.96\times$ and $2.50\times$) across to $T = 50$. That assumes the saving scales with T rather than firing at an absolute step count; we have not tested it, and it is the weakest step in the table. *Executed* is $N\times$depth, the position-forwards our implementation actually computes, which is proportional to depth because a pass costs the same at every L'. *Useful* is $L'\times$depth, what a position-sparse implementation would pay; we do not attain it. Executed differs from depth by the constant N, but useful differs by L', which itself varies $16\times$ across the table, and that varying multiplier is the whole of the reversal between the two (Sec. 2). Perplexity is held-out WikiText-103 (↓). Specialists have no stopping rule, so realised equals scheduled.

L'	1	2	4	8	16
DIAL *(one model, adaptive stopping)*					
Depth: forwards / token	16.89	10.00	6.20	5.00	3.13
Executed: position-fwd / token ($\times 10^3$)	17.3	10.2	6.4	5.1	3.2
Useful: position-fwd / token	16.9	20.0	24.8	40.0	50.0
Batch-1 latency (s)	881	503	301	244	151
Perplexity	28.0	28.4	28.9	29.7	30.6
Specialist $L' = 4$ (fixed block, no stopping)					
Depth: forwards / token	50.00	25.00	12.50	6.25	3.13
Executed: position-fwd / token ($\times 10^3$)	51.2	25.6	12.8	6.4	3.2
Useful: position-fwd / token	50.0	50.0	50.0	50.0	50.0
Perplexity	33.1	31.0	30.4	32.5	37.6
AR reference, 170*M, same corpus and budget*					
Depth, executed and useful / token			1.00		
Perplexity (exact NLL)			28.41		

The latency row needs a caveat, and it compounds with the one in the caption. Only $L' \in \{4, 8, 16\}$ were timed end to end at $T = 50$; the $L' \in \{1, 2\}$ entries are depth multiplied by the measured per-forward time from Table 15 (0.0509 and 0.0491 s), and the depths they use are themselves reconstructed rather than measured. So 881 is an extrapolated chain length times a measured per-forward time, not a stopwatch reading, and the $5.8\times$ inherits the assumption that the stopping saving scales with T. An earlier draft defended the reconstruction on the ground that it reproduces the three timed entries exactly. That is circular, since the per-forward time is calibrated on those same entries, and we withdraw the argument.

On the useful axis the ordering reverses: useful work per token is the realised inner-step count, rising from 16.89 to 50.0. An implementation that computed only active positions would therefore see the sweep as a $2.96\times$ cost *increase*. We flag this rather than claim it as a second result, because the two axes are one depth curve times N and times L' respectively, and because it means our measured saving would not survive the obvious optimisation of gathering the L' active positions before the feed-forward and vocabulary projections (App. B.3).

The AR row is the one a serving engineer would compare against, and we include it because excluding it would flatter us. It costs 1.00 on both axes and reads 28.41, so it is cheaper than every cell of both diffusion rows: by $3.13\times$ to $50\times$ on depth and by $16.89\times$ to $50\times$ on work, at comparable or better quality. Two qualifications, neither of which rescues the dial. Our perplexities are Monte Carlo bounds of uncharacterised slack (App. A.3) and the AR figure is exact, so the quality comparison is not like for like. And a diffusion model buys capabilities an AR decoder has no analogue for, infilling in particular (Table 9), which is the case for this line of work and is not a cost argument. At this scale and with this sampler, block diffusion is not on the quality–compute frontier against a matched AR model, and readers should not infer from the elasticity result that it is.

E.13 Sequential ablation: isolating MBT

Table 3 is leave-one-out, which measures each mechanism's marginal value given the other three and cannot separate a main effect from an interaction. Building the recipe up one mechanism at a time

from the BD3-LM baseline isolates MBT's own contribution, which the headline design otherwise confounds with three architectural changes.

Table 13: **Sequential ablation from BD3-LM to DIAL.** Perplexity (↓) at $L' = 4$, and the *elasticity gap*, the spread in WikiText-103 perplexity between the best and worst inference block size in $\mathcal{S}$ for that model. The final row is DIAL. The last block isolates the antithetic per-block t sampling that the MBT recipe otherwise bundles in.

Recipe	WT-103	LAMBADA	PTB	Elasticity gap
BD3-LM baseline (fixed $L' = 4$)	30.4	49.09	89.43	7.2
+ MBT	30.9	50.6	91.0	3.4
+ per-block time conditioning	30.2	49.4	89.1	3.0
+ RDR	29.4	47.1	85.6	2.8
+ two-stream (= DIAL)	28.9	46.3	84.1	2.6
BD3-LM baseline + antithetic t only	30.2	48.7	88.6	7.1

Three things follow, and the first is the one we would emphasise. MBT alone *costs* in-domain quality, moving WikiText-103 from 30.4 to 30.9 at the block size the baseline was trained for, while collapsing the elasticity gap from 7.2 to 3.4: block-size randomisation buys the dial and charges about half a point for it. The remaining three mechanisms then take the recipe to 28.9, which is 1.5 PPL *below* the BD3-LM baseline rather than merely back to it, so "recovering the MBT penalty" understates what they do and "MBT delivers the quality gain" misattributes it. The accurate reading is that MBT supplies the elasticity and the other three supply the quality, with the caveat below.

That caveat is the second point, and it is sharper than we first put it: these are five separately retrained models, and the increments differ from the leave-one-out values, adding two-stream last being worth 0.5 where removing it costs 0.7 (Table 3). We previously read that gap as mechanism interaction. That reading is wrong, because the two differences share the endpoint 28.9 and their other endpoints are the *same* configuration: "+ RDR" at 29.4 above and "− two-stream" at 29.6 in Table 3 are both MBT with per-block time conditioning and RDR and no two-stream, trained and scored twice. The 0.2 between them is run-to-run variation, not interaction. At 2.2 times DIAL's seed standard deviation of 0.09 (App. E.14) it is the practical resolution of a single-run ablation cell here, so differences below about 0.2 PPL in either table are unresolved. Interaction may still be present in other cells, which we cannot separate from the same noise. The build-up order is one of many, and neither table alone should be read causally. In particular the $+0.5$ attributed to MBT here and the -0.5 attributed to it in Table 3 are consistent in magnitude but are not the same quantity, the first being MBT's effect on the BD3-LM recipe and the second its effect in the presence of the other three. Third, antithetic t sampling accounts for 0.2 on its own and leaves the elasticity gap at 7.1, so elasticity is attributable to block-size randomisation rather than to the time-sampling estimator the recipe also changes.

E.14 Seed variability on the central comparisons

The grid of Table 1 is one run per cell. We replicate the diagonal and the DIAL row that carry the dominance claim across three seeds.

Dispersion is 0.1–0.2 PPL on both kinds of seed, an order of magnitude below the margins it would have to explain: DIAL's advantage over the specialist served at the same block size is 1.49 at $L' = 4$ and 1.61 at $L' = 16$, with off-design margins larger still. Since the specialist spread is evaluation noise rather than training noise, the honest reading is that our own training variance is small and that the specialists' is unmeasured; the margins survive our variance comfortably and would need the baselines to be roughly ten times more seed-sensitive than we are for that conclusion to change. Seed noise is therefore not a plausible account of Table 1, though three seeds bound the variance loosely and we replicate the diagonal rather than all thirty cells.

Table 14: **Three-seed replication** of the comparisons the central claim rests on: WikiText-103 perplexity (↓), mean ± standard deviation. The two dispersions are *not* the same quantity. For DIAL, which we train, the spread is over three independent training seeds. For the specialists, which are the released BD3-LM weights (App. D.3), retraining is not available to us and the spread is over three seeds of the MC-ELBO estimator on a fixed model. The specialist figure therefore bounds evaluation noise only and says nothing about how much those models would move under retraining. An earlier version of this table divided each margin by the two dispersions pooled in quadrature, which mixes training variance with estimator variance; we now divide by DIAL's training σ alone, which is the only training variance we measured. Since the specialists' training variance is unmeasured rather than zero, these ratios are upper bounds.

Inference L'	Model	Perplexity	Margin	Margin / σ_{DIAL}
1	DIAL	28.04 ± 0.11	·	·
4	DIAL	28.92 ± 0.09	·	·
	Specialist $L'=4$	30.41 ± 0.13	1.49	≤ 16.6
16	DIAL	30.63 ± 0.14	·	·
	Specialist $L'=16$	32.24 ± 0.16	1.61	≤ 11.5
	Specialist $L'=4$	37.55 ± 0.21	6.92	≤ 49.4

E.15 Interactive latency, batch size and memory

Throughput in Table 1 is aggregate at batch 16, which is the wrong regime for the interactive serving the paper motivates. Table 15 reports per-request latency at batch 1 and peak memory across the dial, on the same single RTX A6000 and at the inner budgets of Table 1.

Table 15: **Per-request latency and memory across the dial**, one RTX A6000, $N = 1024$, generating a full sequence per request. Latency is wall-clock seconds per request at batch 1; throughput is aggregate tokens per second at the stated batch. Batch-16 throughput is the measurement already reported in Table 1. Inner budgets follow that table ($T=20$ at $L' \in \{1, 2\}$, $T=50$ otherwise), which is why the $L'=2$ column is faster than $L'=4$. *As in Table 1, these two were run at batch 4 and scaled to batch 16 by the measured batch-scaling factor, so they are not directly comparable with the three measured batch-16 entries.

L'	1	2	4	8	16
Latency P50 (s, batch 1)	353	201	301	244	151
Latency P95 (s, batch 1)	388	221	331	266	154
Throughput (tok/s, batch 1)	2.9	5.1	3.4	4.2	6.8
Throughput (tok/s, batch 4)	6.0	10.5	7.0	8.7	14.1
Throughput (tok/s, batch 16)	9.1*	15.9*	10.4	13.1	**21.2**
Peak memory (GB, batch 1)	3.41	3.42	3.43	3.45	3.49
Peak memory (GB, batch 16)	14.6	14.7	14.8	14.9	15.1

Two points matter for deployment. Peak memory is nearly flat across the dial, varying by 0.08 GB (2.3%) at batch 1 and 0.5 GB (3.4%) at batch 16, which corroborates the accounting of App. B.2: the activation tensors are sized by N, and the residual drift with L' is the attention kernel's workspace rather than a term that scales with the block. In absolute terms a server can switch L' per request without repacking, provided it is provisioned against the largest block size. Second, the P95-to-P50 ratio widens where stopping is active, from 1.02 at $L'=16$ to 1.10 at $L'=1$, because adaptive stopping makes realised cost content-dependent. Elastic serving thus trades a little tail predictability for its mean-case saving, and an SLO-bound deployment should size against P95 rather than against the NFE reduction.

E.16 Mechanism diagnostics: RDR by block size, and gradient alignment

Two diagnostics test the intuitions we use to motivate the design rather than assuming them.

RDR should matter more where coordination is harder. RDR exists to coordinate positions committed jointly, so its benefit should widen with L'. Table 16 ablates it at every block size.

Table 16: **RDR benefit as a function of block size**, WikiText-103 perplexity (↓). Δ_{RDR} is the perplexity removed by the mechanism at that block size. The $L'=4$ ablation is the row already in Table 3.

Inference L'	1	2	4	8	16
DIAL	28.0	28.4	28.9	29.7	30.6
DIAL without RDR	28.3	29.1	30.3	31.9	33.7
Δ_{RDR}	0.3	0.7	1.4	2.2	3.1

Δ_{RDR} rises monotonically from 0.3 at $L'=1$ to 3.1 at $L'=16$. At $L'=1$ there is nothing to coordinate and RDR is nearly inert, exactly as its stated purpose predicts; at $L'=16$ it is the difference between a usable and an unusable parallel end. This is a sharper test than the single-block-size ablation, because the mechanism's rationale predicts the *shape* of the curve and not merely the sign.

Do block sizes actually conflict? We motivate the auxiliary mechanisms by asserting that gradients from small and large blocks conflict. Table 17 measures it, as the cosine similarity between per-block-size gradients on a shared batch, averaged over training.

Table 17: **Cosine similarity between per-block-size gradients**, $\cos(g_{L'_i}, g_{L'_j})$, averaged over a held-out set of batches at the mid-training model. Symmetric; diagonal omitted.

	$L'=2$	$L'=4$	$L'=8$	$L'=16$
$L'=1$	0.71	0.52	0.34	0.19
$L'=2$	·	0.68	0.47	0.29
$L'=4$	·	·	0.66	0.44
$L'=8$	·	·	·	0.63

The measurement refines the motivation rather than confirming it. Every pair is positively aligned, so there is no outright conflict in the multi-task-learning sense; the accurate description is *decaying alignment* with block-size ratio, adjacent sizes agreeing strongly (0.63–0.71) while the extremes are close to orthogonal (0.19). This is a friendlier optimisation problem than conflict would imply, and it explains both why MBT costs only about half a point of in-domain perplexity (Table 13) and why per-block conditioning helps: near-orthogonal gradients mean the extremes compete for capacity rather than undo one another, and conditioning lets the network allocate that capacity conditionally.

E.17 Ranking margins and rank agreement

Sec. 4.2 observes that choice-based accuracy is far more stable across the dial than absolute continuation likelihood, and attributes this to ranking being more robust than calibration. Table 18 tests that directly, through the per-example margin between the correct candidate and the best incorrect one, and the rate at which the induced ranking agrees with $L'=1$.

Table 18: **Ranking is more stable than likelihood.** *Upper:* mean per-example margin $\Delta = \log P(\text{correct}) - \max_{j \neq \text{correct}} \log P(j)$, in nats per token. *Lower:* fraction of examples whose full candidate ranking is unchanged from $L'=1$. One DIAL model throughout.

Inference L'	1	2	4	8	16
Mean correct-vs-best-incorrect margin (↑)					
PIQA	0.0412	0.0409	0.0405	0.0399	0.0391
BoolQ	0.0298	0.0295	0.0292	0.0287	0.0281
ARC-easy	0.0521	0.0518	0.0514	0.0507	0.0499
ARC-challenge	0.0387	0.0385	0.0382	0.0377	0.0370
Rank agreement with $L'=1$ (↑)					
PIQA	1.000	0.981	0.972	0.966	0.961
BoolQ	1.000	0.978	0.969	0.960	0.953
ARC-easy	1.000	0.983	0.975	0.968	0.962
ARC-challenge	1.000	0.979	0.970	0.963	0.957

The two halves separate the effect. Margins do shrink as the dial opens, by 5% relative on PIQA, so the dial reaches the choice-based tasks and the model is less confident at large blocks. But the shrinkage is small next to the margin itself, so the *ordering* survives: at least 95% of examples on every task rank their candidates exactly as at $L' = 1$, while absolute LAMBADA likelihood moves by 0.041 nats per token over the same sweep. Stable multiple-choice accuracy is therefore not evidence that the dial is free downstream, but that argmax over candidates is a coarser readout than the normalisation it discards, which predicts the tax should reappear on anything scored by absolute likelihood or calibrated confidence rather than by rank.

F Failure cases and remaining gaps

Sec. 6 lists the limitations compactly. Here we separate two things a reader should keep apart: places where DIAL demonstrably does *not* deliver, which our own measurements show, and places where we simply have no evidence, which are open questions we would want answered before the claims are widened.

Where DIAL does not deliver. *Stopping buys nothing at the parallel end.* Adaptive stopping reduces forwards by 2.96× at $L' = 1$ but by 1.00× at $L' = 16$ (Table 1): sixteen positions rarely reach a stable joint argmax within $T = 50$, so the mechanism that makes the accurate end affordable is inert exactly where the scheduled cost is already lowest. The two efficiency mechanisms are therefore complementary rather than cumulative, and their partial cancellation is why measured throughput rises 2.0× from $L' = 4$ to $L' = 16$ rather than the 4× the block ratio alone suggests. *Downstream accuracy still trails modern AR models.* DIAL reaches 0.63 BLIMP against 0.717 for size-matched Pythia-160M (Biderman et al., 2023), and 0.298 HellaSwag against 0.421 for SmolLM2-135M (Allal et al., 2025) at a much larger token budget (Table 2 for Pythia, App. E.6 for SmolLM2, which does not appear in the main table). Elasticity does not close the diffusion-to-AR gap on these tasks at this scale; it removes the need to choose an operating point in advance. *Nominal compute overstates realised speed.* The 16× is a scheduled forward-pass count at matched T; the headline serving configurations of Table 1 use different inner budgets at different block sizes and schedule a 6.4× range, of which 2.2× survives once adaptive stopping is applied. Over the part of the range where our runs share a batch and step budget, measured throughput moves by 2.04×, and that mapping is specific to one accelerator, kernel and batch size. *Infilling at low mask ratios.* At a 0.10 mask ratio the AR baseline is better than every diffusion row (34.8 against 41.8 for DIAL at $L' = 4$); DIAL's advantage appears only once enough of the sequence is missing that bidirectional context matters.

What we still cannot answer. App. E.11–E.17 close the gaps a sceptical reader is most likely to probe. Four remain, and we state them so their absence is not mistaken for a negative finding. *Scale.* Every result is at 170M non-embedding parameters. The ladder of App. E.5 is monotone from 15M to 170M, but whether an elastic model still outperforms a specialist grid at a billion parameters is untested, and the gradient-alignment picture of Table 17 may not survive the change in capacity. *Per-sample stopping.* Our stopping rule is batch-coupled: every sample waits for the last straggler in its batch, so the reductions in Table 1 are a lower bound on what per-sample stopping would save. We have not implemented the per-sample variant, and its wall-clock gain would not be proportional to its NFE gain, because dynamic shapes cost occupancy. *Hardware generality.* The mapping from L' to seconds in Table 15 is specific to one RTX A6000, its attention kernel and our batch sizes; on an accelerator with a different memory-to-compute balance the same dial traces a different latency curve, even though the forward-pass counts are unchanged. *Coverage of the replication.* Table 14 replicates the six configurations the dominance claim rests on, not all thirty cells of the grid, and three seeds bound the variance loosely.