PRUNING DNA SEQUENCE

Anonymous authors
Paper under double-blind review

ABSTRACT

Data pruning can substantially reduce training cost by removing examples that contribute little additional learning signal, but it has not been studied for DNA language-model pretraining. We introduce Sequence Pruning (SP), a dynamic pruning framework that removes genomic sequences as they become easy for the model to predict. SP uses per-sequence pretraining accuracy computed directly from the model's forward pass, requiring no separate scoring model or additional pass over the corpus. We evaluate SP across five pretraining settings spanning two corpora, three model families, and both MLM and CLM objectives, with downstream evaluation on five genomic benchmarks. SP reduces pretraining data processing by up to 96% while largely preserving downstream performance. At moderate savings, pruning improves performance by up to 1.2 MCC points over full-data pretraining; at the highest savings, pretraining uses only 4% of the baseline GPU hours with little performance loss. SP also outperforms the dynamic pruning methods InfoBatch and BPAS across most comparisons. Beyond accuracy, our framework supports alternative criteria such as perplexity and EL2N, although accuracy yields the strongest downstream performance at matched savings. These results show that genomic pretraining does not require repeatedly processing all available sequences and that dynamic sequence-level pruning can substantially improve its data and computational efficiency.

1 INTRODUCTION

DNA, RNA, and proteins encode biological information as sequences over small alphabets, where the role of each element depends strongly on its surrounding context. DNA remains particularly difficult to interpret: only a small fraction of the human genome encodes proteins (Lander et al., 2001), while the functions of much of the remaining sequence are still incompletely understood (ENCODE Project Consortium, 2020). Functional labels therefore depend on experimental assays that are far more costly and slower to obtain than raw genomic sequence. As sequencing technologies continue to generate data at increasing scale, genomics faces a growing imbalance between abundant unlabeled sequence and scarce functional annotation. Self-supervised language modeling is well suited to this setting because it learns directly from raw sequence and transfers the resulting representations to downstream tasks. Genomic foundation models broadly follow two pretraining paradigms. Encoder-only models, including DNABERT (DB) (Ji et al., 2021), DNABERT-2 (DB2) (Zhou et al., 2024), Nucleotide Transformer (NT) (Dalla-Torre et al., 2025), and GENA-LM (Fishman et al., 2025), use masked language modeling (MLM), while decoder-only models such as HyenaDNA (Hyena) (Nguyen et al., 2023), Evo (Nguyen et al., 2024), and Evo 2 (Brixi et al., 2026) use causal language modeling (CLM).

From a computational perspective, genomic foundation models have followed two broad scaling strategies. The first emphasizes efficiency, aiming to maintain downstream performance with fewer parameters and, in some cases, less pretraining data. For example, DB2, with 117M parameters, remains competitive with the 2.5B-parameter NT, while Hyena reports performance comparable to NT using $1500\times$ fewer parameters and $3200\times$ less pretraining data. The second strategy instead pursues performance through increasingly large models and pretraining corpora. Evo 2 scales to 40B parameters and is pretrained on 9.3T nucleotides, while NTv3 (Boshar et al., 2025) is pretrained on 9T base pairs. Pretraining at this scale is inaccessible to most research groups, yet recent progress increasingly depends on larger computational budgets. At the same time, Vishniakov et al. (2026) found that randomly initialized models can sometimes match or even outperform their pretrained counterparts on downstream tasks, suggesting that additional pretraining does not necessarily translate into better

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.

representations. Together, these observations raise a more fundamental question: How efficiently do genomic foundation models use their pretraining data to learn transferable representations?

Genomic pretraining datasets are themselves large. The Human Reference Genome (HRG) contains 2.75B bases (Dalla-Torre et al., 2025), while the Multi-Species Genome collection (MSG) contains 32.49B (Zhou et al., 2024). Because the computational cost of pretraining scales with the amount of sequence processed, larger corpora directly increase training cost. Yet it is unclear whether all sequences remain equally informative throughout pretraining. Some sequences may continue to provide useful learning signals, whereas others may become redundant once the model can predict them reliably. This motivates a more targeted question: Can a substantial fraction of DNA pretraining data be removed without degrading downstream performance?

Data pruning has been studied extensively in other domains. In vision, EL2N (Paul et al., 2021) scores each example by the ℓ_2 norm of its prediction error and removes low-scoring examples. Marion et al. (2023) evaluate the same criterion for LLM pretraining and compare it against perplexity-based pruning. Using perplexity, they show that training on 30% of a web-text corpus can match full-data training. Both EL2N- and perplexity-based approaches are static: they score the corpus using a reference model before the pruned training run begins and perform pruning only once at initialization. Static pruning has also been studied for other biological sequences. Wu et al. (2026), for example, prune RNA and protein corpora using a post-hoc influence criterion computed from a model already pretrained on the full corpus. Because this criterion itself requires a complete pretraining run, it is less suitable when the primary objective is to reduce pretraining cost. Dynamic pruning avoids this upfront requirement by updating sample importance during training. InfoBatch (Qin et al., 2024) follows this strategy, while Alam et al. (2026) outperform InfoBatch with batch pruning by activation stability (BPAS). We evaluate InfoBatch and BPAS as dynamic pruning baselines for DNA pretraining.

To the best of our knowledge, data pruning has not previously been studied for DNA language-model pretraining. We address this gap with Sequence Pruning (SP), a dynamic sequence-level pruning strategy based on a hypothesis: once a sequence is well learned, continuing to train on it provides diminishing benefit. Equivalently, easy sequences should contribute less useful learning signal than hard sequences. A similar pattern has been observed in web-text pretraining, where Marion et al. (2023) report that removing the easiest sequences can improve performance. This motivates a criterion for distinguishing easy from hard sequences during pretraining. Our approach is related to curriculum learning (Bengio et al., 2009), which assigns examples a notion of difficulty, but differs in how that difficulty is used: rather than determining the order in which examples are presented, we use it to decide which sequences to remove. We define sequence difficulty using per-sequence pretraining accuracy (hereafter, *accuracy*): MLM accuracy for encoder models and CLM accuracy for decoder models. Higher accuracy indicates that a sequence is easier and better learned; pruning such sequences therefore reduces computation spent on examples the model already predicts reliably.

Can accuracy reliably distinguish easy sequences from hard ones? To investigate this, we track pretraining dynamics for DB2 and Hyena, covering both MLM and CLM objectives. Figure 1 follows ten sequences from each setting and shows that per-sequence accuracy is largely stable across epochs: high-accuracy sequences tend to remain high, low-accuracy sequences remain low, and intermediate sequences stay in between. This stability does not mean that sequence accuracies remain unchanged: as the model learns, accuracies gradually shift upward while sequences largely remain within their relative difficulty bands. Figure 2 shows this behavior at the corpus level and makes the gradual improvement across epochs more visible. Each curve shows how many sequences fall into different accuracy ranges within one epoch, while curves from different epochs show how this distribution changes throughout pretraining. The first epoch has a distinct distribution because the model has only just begun learning the corpus. From the second epoch onward, however, the curves retain a similar overall shape while gradually

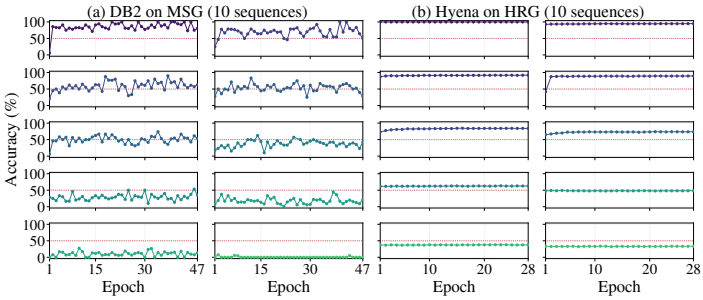

Figure 1: Per-sequence pretraining accuracy trajectories for ten sequences each from DB2 (a) and Hyena (b). In each panel, the sequences are selected to span the observed accuracy range.

shifting toward higher accuracy: fewer sequences remain in the low-accuracy ranges, while more move into the high-accuracy ranges. Thus, sequences identified as easy or hard early in pretraining tend to retain that status even as their accuracies improve. Beyond this coarse separation, accuracy also induces a finer-grained ordering from easiest to hardest, and after a single pass over the corpus, this ordering changes only marginally (Appendix A). If accuracy is stable enough to preserve this finer ordering, it should also be sufficient for the coarser task of separating easy from hard sequences.

From which point should a sequence be considered easy or hard? This requires an accuracy threshold. Since easy sequences are the ones we aim to remove, we define the pruning range using two thresholds. The high threshold is fixed at perfect accuracy, while the low threshold, τ_{low}, determines where pruning begins. We vary only τ_{low}, making it the only tunable parameter introduced by our method. We therefore study how different values of τ_{low} affect downstream performance.

We study sequence pruning (SP) across five pretraining settings spanning two objectives, two corpora, and three model families, and evaluate downstream performance on five benchmarks. SP reduces pretraining data by 3% to 96%. At moderate savings, it not only preserves downstream performance but improves it by up to 1.2 MCC points over the unpruned baseline. At the highest savings, the model pretrains using only 4% of the baseline GPU hours while maintaining comparable downstream performance. SP also outperforms the dynamic methods InfoBatch and BPAS: it achieves greater data savings and yields better downstream performance in most comparisons. We further incorporate two established pruning criteria, perplexity and EL2N, into our framework. Although both were originally designed for static pruning, recomputing them at each epoch boundary remains effective, demonstrating that our framework supports pruning criteria beyond accuracy. At matched savings, however, accuracy yields higher downstream performance than perplexity and EL2N. Our contributions are as follows.

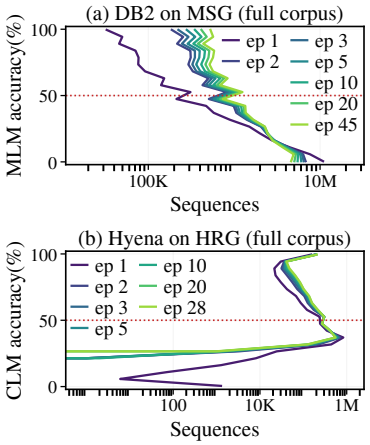

Figure 2: Accuracy distributions across corpus at selected epochs for DB2 (32.4M sequences, 47 epochs) and Hyena (2.7M sequences, 28 epochs). Curves show how sequence counts vary across the accuracy range.

1. We introduce data pruning to DNA pretraining. To the best of our knowledge, data pruning has not previously been applied to DNA corpora. We show that a large fraction of pretraining sequences can be removed without degrading downstream performance and, in several settings, that pruning can outperform full-data pretraining.

2. We introduce accuracy as a pruning criterion. Per-sequence pretraining accuracy consistently separates easy from hard sequences throughout pretraining and is obtained directly from the model's forward pass. It therefore requires no additional pass over the corpus and leaves the underlying pretraining recipe unchanged. Our method introduces only one tunable parameter, the low accuracy threshold, τ_{low}. We identify two values that generalize across all settings we evaluate: one that yields the strongest downstream performance and one that achieves the largest data savings.

3. We propose a general plug-and-play framework for dynamic pruning of DNA sequences. The framework supports both MLM and CLM objectives and accommodates multiple pruning criteria, including accuracy, perplexity, and EL2N.

2 METHODOLOGY

Figure 3 provides an overview of our pruning framework, which consists of two main components: the corpus and the pretraining model. Within the pretraining component, sequences pass through block A for MLM or block C for CLM, depending on the pretraining objective. At each epoch boundary, block B applies the pruning criterion and updates the corpus used in the following epoch. Thus, our method changes only which sequences are presented to the model across epochs. The model architecture, pretraining objective, optimizer, and downstream evaluation protocol remain unchanged from the corresponding unpruned setup.

A corpus contains N sequences indexed by p, and a pretraining run consists of I epochs indexed by i. At epoch i, the model trains on an active corpus $\mathcal{A}_i \subseteq \{1, \ldots, N\}$, initialized with the full

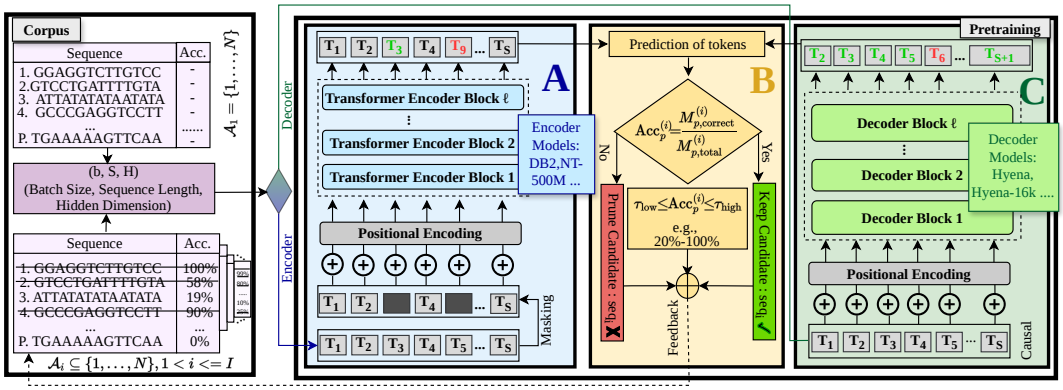

Figure 3: Overview of the proposed SP framework. Left: the active corpus across epochs. Right: the pretraining pipeline, consisting of blocks A and B for MLM or blocks B and C for CLM, depending on the pretraining objective. Block B shows $[20\%, 100\%]$ as an illustrative pruning range. T = Token.

corpus, $\mathcal{A}_1 = \{1, \ldots, N\}$, corresponding to the corpus entering epoch 1 in Figure 3. Sequences in $\mathcal{A}_i$ are tokenized and grouped into batches, producing model inputs of shape (b, S, H), where b is the batch size, S is the sequence length, and H is the hidden dimension. Each batch then follows block A or block C according to the model's pretraining objective. In the encoder branch (A), a fraction of tokens in each sequence is selected and masked. The sequence is passed through ℓ transformer encoder blocks, and the model predicts the original tokens at the masked positions. In the decoder branch (C), the sequence passes through ℓ causal decoder blocks, so each position can access only the tokens that precede it, and the model predicts each token from this context. Despite these different objectives, both branches produce token-level predictions at their respective prediction positions. We use the correctness of these predictions to compute per-sequence MLM accuracy for encoder models and per-sequence CLM accuracy for decoder models.

Let $M^{(i)}_{p,\text{total}}$ denote the number of prediction positions scored for sequence p at epoch i: the masked positions for MLM and every position after the first for CLM. Let $M^{(i)}_{p,\text{correct}}$ denote the number of these positions predicted correctly. The per-sequence accuracy is $\text{Acc}^{(i)}_p = M^{(i)}_{p,\text{correct}} / M^{(i)}_{p,\text{total}}$, where $M^{(i)}_{p,\text{total}} > 0$. Because per-sequence accuracy is computed from the same token predictions used for the training loss, only $M^{(i)}_{p,\text{correct}}$ and $M^{(i)}_{p,\text{total}}$ need to be accumulated, with no additional forward pass. At the end of epoch i, this yields one accuracy value for each scored sequence in $\mathcal{A}_i$.

The accuracy definition is shared across both objectives, but the positions that contribute to it differ. Under MLM, the $80/10/10$ rule replaces selected tokens with `[MASK]`, random tokens, or leaves them unchanged. We score only the `[MASK]` positions to keep the accuracy calculation consistent across sequences. Because masking is resampled at every epoch, the accuracy of a sequence may be computed from different positions over time, introducing variability due to mask sampling. Despite this variability, the stability observed in Figure 1 and 2 indicates that the distinction between easy and hard sequences is generally preserved. Under CLM, every position after the first contributes to the accuracy calculation, resulting in less epoch-to-epoch fluctuation than under MLM (Appendix A.4).

At each epoch boundary, every sequence in $\mathcal{A}_i$ with valid accuracy is compared against the pruning thresholds, $\tau_{\text{low}} \leq \text{Acc}^{(i)}_p \leq \tau_{\text{high}}$, and sequences whose accuracy falls within this range are removed to form $\mathcal{A}_{i+1}$ for the next epoch. We fix $\tau_{\text{high}} = 100\%$ for all SP variants, corresponding to perfect accuracy, making τ_{low} the only parameter we vary. Lowering τ_{low} expands the pruning range, making more sequences eligible for removal and pruning a larger fraction of the corpus. Since τ_{high} remains fixed at 100% accuracy, pruning is restricted to sequences the model already predicts well.

As the active corpus shrinks, each epoch of a pruned run requires fewer optimization steps than the unpruned baseline. Retaining the baseline step budget would therefore cause the remaining sequences to be processed more frequently, confounding data reduction with sample repetition. To avoid this, each pruned run retains the baseline epoch budget I and terminates after at most I epochs, or earlier if the active corpus becomes empty. This ensures that no retained sequence is processed more times than in the baseline. The learning-rate schedule remains tied to the baseline step count and is not rescaled, so the pruned run and baseline use the same learning rate at each matched optimization step. For InfoBatch and BPAS, block B in Figure 3 is replaced by each method's native pruning procedure,

while the corresponding pretraining setting is otherwise kept fixed. For perplexity and EL2N, only the pruning criterion within block B is replaced, leaving the rest of the SP framework unchanged. Additional methodological details are provided in Appendix B.

3 EXPERIMENT

Pretraining settings. A pretraining setting is defined by a specific combination of pretraining corpus and model. Table 1 summarizes the five settings, which vary in corpus, model scale, context length, and pretraining objective. Relative to DB2, NT-500M changes the model scale, Hyena changes the pretraining objective from MLM to CLM, and Hyena-16k increases the context length to 16,384 bp. Each setting follows its published pretraining recipe (Appendix C).

Downstream benchmarks. Figure 4 summarizes the pretraining settings and their corresponding downstream evaluations. DB2 and NT-500M are evaluated on Genome Understanding Evaluation (GUE) (Zhou et al., 2024) and the Nucleotide Transformer benchmark (NT-18) (Dalla-Torre et al., 2025), while Hyena is evaluated on Genomic Benchmarks (GB) (Nguyen et al., 2023). These benchmarks use full-parameter fine-tuning, in which the entire pretrained model is updated for each task. Hyena-16k is evaluated on BEND (Marin et al., 2024) gene finding (B-GF) and DART-Eval (Patel et al., 2024) Task 4 (D-T4) using frozen probing, in which the pretrained model remains fixed and a small convolutional probe is trained on its embeddings. We report MCC for all benchmarks except D-T4, for which we report Spearman correlation. GUE and NT-18 organize their tasks into groups, so we first average performance within each group and then across groups, following their respective reporting protocols. For GB, we average performance directly across its seven tasks. Appendix D provides the full fine-tuning and frozen-probing protocols.

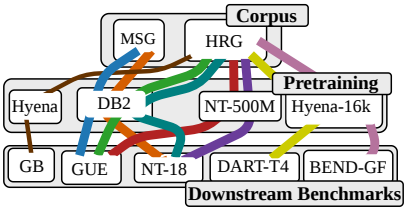

Figure 4: Pretraining settings and corresponding downstream benchmarks; Edges indicating the evaluation paths.

Variants and efficiency measures. Sequence Pruning (SP) variants are named by their τ_{low}. SP_50% sets $\tau_{\text{low}} = 50\%$ and $\tau_{\text{high}} = 100\%$, so at each epoch boundary, sequences with accuracy $50\% - 100\%$ are pruned. For the other variants, the numerical suffixes only distinguish configurations; their parameters are listed in Appendix F. For efficiency measurement, DSI remains our primary measure. Following Alam et al. (2026), if N is the number of sequences in the full corpus, $\mathcal{A}_i$ is the active corpus in epoch i, I is the scheduled number of epochs, and I_s is the stopping epoch, then $\text{DSI}=1-\sum_{i=1}^{I_s}|\mathcal{A}_i|/(I \cdot N)$, where higher values indicate greater savings. $\text{DSI}=50\%$ means that half of the baseline sequence processing is avoided. Additionally, we report GPU hours, computed as the number of GPUs multiplied by wall-clock training time. Pretraining uses 24 NVIDIA A100 GPUs, and fine-tuning uses one.

Table 1: The five pretraining settings. BPE tokenizer denotes byte-pair encoding, and 6-mer denotes non-overlapping groups of six nucleotides.

Corpus	Model	Param.	Toknzr.	Vocab	Seq. Len.	# Seq.	Batch
MSG	DB2	117M	BPE	4096	128	32.4M	3072
HRG	DB2	117M	BPE	4096	128	3M	3072
HRG	NT-500M	500M	6-mer	4105	1000	500k	1008
HRG	Hyena	6.6M	char	12	1024	2.7M	1536
HRG	Hyena-16k	7.2M	char	12	16,384	185.6k	96

3.1 RESULT ANALYSIS

Table 2 reports results for the three encoder pretraining settings: DB2 on MSG, DB2 on HRG, and NT-500M on HRG, while Table 3 reports Hyena on HRG, the decoder pretraining setting. Each setting includes two reference points: the unpruned baseline, which uses the full pretraining corpus throughout the run, and full pruned, which skips pretraining and uses the same fine-tuning protocol as our pruned variants. The downstream performance gap between these two reference points reflects the contribution of pretraining, which is the stage targeted by pruning. This gap is substantial for the encoder settings but smaller for Hyena. Hyena uses character-level tokenization, and Vishniakov et al. (2026) similarly report limited gains from pretraining for character-tokenized models. For Hyena, SP_50% even outperforms the unpruned baseline. Across settings, SP remains substantially closer to the unpruned baseline than to full pruned, indicating that it preserves most of the benefit provided by pretraining. Variant parameters and full pruning sweeps are provided in Appendix E.

Removing easy, well-learned sequences is more effective than removing random or hard sequences. We verify this with two control experiments. The first is random pruning (RP). For RP_20%, we match the DSI of SP_20% by randomly removing the same fraction of the corpus before pretraining and training for the same number of epochs. At matched DSI, RP_20% performs worse than SP_20% in both encoder and decoder settings, showing that data reduction alone does not reproduce the gains from SP. The second control is low-accuracy pruning (LP). Rather than removing easy sequences, LP_0% removes hard sequences at every epoch boundary by setting $\tau_{\text{low}} = \tau_{\text{high}} = 0\%$. Thus, only sequences with 0% accuracy are removed. This is the only variant in the paper for which the high pruning threshold (τ_{high}) differs from 100%. On DB2–MSG, LP_0% achieves performance similar to RP_20%.

Comparison with dynamic pruning. In Table 2, Info_1 uses the pruning ratio and annealing point recommended by Qin et al. (2024) and reaches about 19% DSI. Info_2 targets higher DSI but performs below the full pruned DB2 baseline, suggesting that InfoBatch may require modifications beyond its recommended recipe for DNA pretraining. BPAS degrades more gradually and provides the stronger dynamic-pruning comparison. Table 2 and Table 3 compare BPAS with SP at similar DSI. On DB2–MSG, SP remains competitive with the unpruned baseline at about 77% DSI, whereas BPAS degrades sharply by 71% DSI. On DB2–HRG, SP_20% achieves greater savings while attaining a higher downstream score than BPAS. On NT-500M, SP performs better at every compared DSI on both benchmarks, while SP_50% remains above the unpruned baseline on GUE. On Hyena, SP matches BPAS at the highest savings while reaching 96% DSI versus 92%.

Table 2: Efficiency and downstream performance across the three encoder pretraining settings.

	Pruning Apprch.	Variant	DSI %↑	GPU↓ Hours	Epch.	Steps (k)	MCC↑ GUE	MCC↑ NT-18
DB2 on MSG	Baseline	Unprun.	0	1012	47	500	**71.6**	**69.8**
		Full prun.	100	0	0	0	60.6	47.1
	Related Work	BPAS_1	6	977	47	470	71.3	68.8
		BPAS_2	19	777	47	400	71.2	69.6
		BPAS_4	36	838	38	310	71.0	68.6
		BPAS_9	71	318	24	140	68.9	67.5
		Info_1	19	709	47	400	69.8	68.6
		Info_2	29	600	47	349	39.4	48.3
	Ours	SP_100%	5	994	47	480	71.5	69.5
		SP_80%	12	905	47	430	70.8	69.2
		SP_50%	32	713	47	330	71.5	69.0
		SP_20%	77	226	47	110	70.5	69.1
	Ablation	RP_20%	77	215	47	110	68.1	65.6
		LP_0%	55	378	47	230	68.5	64.9
DB2 on HRG	Baseline	Unprun.	0	47	32	30	**66.6**	64.1
		Full prun.	100	0	0	0	43.6	47.3
	Related Work	BPAS_1	0	47	32	30	65.8	62.7
		BPAS_4	29	32	31	21	65.0	62.4
		BPAS_5	40	23	29	18	64.4	62.8
	Ours	SP_100%	2	46	32	29	65.2	63.8
		SP_80%	5	45	32	28	65.4	**65.7**
		SP_50%	18	39	32	25	66.1	63.6
		SP_20%	46	25	32	16	65.2	63.2
	Ablation	EL2N_1	15	39	32	25	64.2	64.7
		EL2N_2	25	34	32	22	65.3	64.6
		EL2N_3	37	32	32	19	65.8	64.2
		EL2N_4	44	27	32	16	65.0	63.0
NT-500M on HRG	Baseline	Unprun.	0	1137	99	50	67.0	**74.2**
		Full prun.	100	0	0	0	53.0	63.7
	Related Work	BPAS_1	1	1087	99	49	66.4	73.8
		BPAS_2	7	1033	99	46	65.6	73.7
		BPAS_4	40	658	99	29	65.6	73.9
	Ours	SP_100%	4	1081	99	47	67.4	73.9
		SP_80%	6	1063	99	46	67.1	73.9
		SP_50%	10	969	99	44	**68.2**	74.0
		SP_20%	37	668	99	30	67.4	74.1

Other pruning criteria. For a fair comparison, we integrate perplexity and EL2N into our framework and recompute them at every epoch boundary. For CLM, we follow Marion et al. (2023) to compute per-sequence perplexity and evaluate it on Hyena. For MLM, we evaluate per-sequence EL2N on DB2–HRG following Paul et al. (2021). For both criteria, we use four thresholds chosen to match the DSI of four SP variants in Table 2 and Table 3; the corresponding thresholds are listed in Appendix F. Table 3 shows that Perp_4 performs worse than SP and BPAS at comparable DSI. At matched DSI, EL2N yields slightly lower downstream performance than SP at nearly every comparison point. Downstream results show that the framework is not limited to accuracy, as both perplexity and EL2N can also serve as pruning criteria. Among the three criteria, accuracy achieves the strongest downstream performance, with a small but consistent advantage over the other two. Accuracy thresholds are also directly interpretable: a threshold of 20% or 50% specifies the per-sequence pretraining accuracy above which sequences become eligible for pruning. In contrast, InfoBatch relies on loss-based selection, BPAS on activation differences, perplexity on likelihood-based scores, and EL2N on error norms. These quantities are less directly interpretable and generally require tuning across settings.

Robustness on harder benchmarks. To test robustness, we add two benchmarks that differ from the others in important ways: B-GF requires a prediction at every nucleotide over sequences up to $14{,}000$ bp, while D-T4 is a regression task evaluated by correlation rather than classification. To accommodate the longer B-GF sequences, we pretrain Hyena-16k at $16{,}384$ bp and evaluate it on both benchmarks. Hyena-16k is pretrained for only two epochs, allowing a single pruning decision after the first epoch. Table 4 shows that all SP variants remain close to the unpruned baseline on both tasks, indicating that SP preserves the information required for these tasks despite having only one opportunity to prune. This result is consistent with Figure 2, where the separation between easy and hard sequences emerges early in pretraining.

Table 3: Efficiency and downstream performance for decoder pretraining setting (Hyena on HRG).

Pruning Apprch.	Variant	DSI GPU ↓ %↑	Hours	Epch.	Steps (k)	MCC↑ GB
Baseline	Unprun.	0	76	28	50	77.0
	Full prun.	100	0	0	0	73.3
Related Work	BPAS_4	2	73	28	48	77.2
	BPAS_6	9	68	28	45	**77.3**
	BPAS_9	63	27	23	18	<u>77.2</u>
	BPAS_12	92	4	2	4	76.1
Ours	SP_100%	4	71	28	47	76.4
	SP_80%	11	65	28	43	76.4
	SP_50%	47	39	28	26	**77.3**
	SP_20%	96	2	28	1.7	76.1
Ablation	Perp_1	6	71	28	47	75.9
	Perp_2	20	51	28	39	76.5
	Perp_3	45	36	28	27	76.7
	Perp_4	96	2	28	1.7	75.4
	RP_20%	96	3	28	1.7	74.8

Are these differences real? We measure run-to-run variation at both the fine-tuning and pretraining stages. Across repeated runs, the largest MCC standard deviation is 0.6 points, while SP improves over the unpruned baseline by up to 1.2 MCC points in Table 2 and 3. Data savings are also stable, with DSI standard deviations of at most 0.15 percentage points across repeated SP_20% pretraining runs. Together, these results show that both downstream performance and data savings are consistent across runs, and that the largest observed performance gain is larger than the measured run-to-run variation (Appendix G).

How far pruning can go depends on the pretraining setting. The savings achieved by a given τ_{low} are not a property of the pruning method alone. The two DB2 settings in Table 2 use the same architecture and differ only in corpus, yet the same parameter yields substantially different savings. This allows us to isolate the effect of the corpus on data savings while holding the model architecture fixed. Conversely, the three HRG settings in Table 2 and 3 share the same corpus but use different models, and their savings also vary widely. The corpus and model therefore jointly determine how much of the data becomes well learned enough to prune, and how quickly this occurs. What remains consistent across settings is the useful range of the parameter. SP_50% gives the strongest downstream performance across settings, while SP_20% is the most aggressive setting before performance begins to degrade meaningfully, although the savings achieved at these settings differ across pretraining configurations.

Table 4: Results on harder downstream benchmarks under frozen probing (Hyena-16k on HRG).

Pruning Apprch.	Variant	DSI GPU↓ %↑	Hours	Epch.	Steps (k)	MCC↑ B-GF	Sprmn↑ D-T4
Baseline	Unprun.	0	7	2	4	**35.0**	36.6
Ours	SP_100%	2	7	2	4	34.5	**38.2**
	SP_80%	3	7	2	4	34.4	34.6
	SP_50%	10	6	2	3.5	<u>34.9</u>	34.6
	SP_20%	49	4	2	2	33.0	37.7
	SP_10%	49	4	2	2	32.6	<u>37.8</u>

Table 5: Effect of averaging-window size on efficiency and downstream performance (DB2 on MSG).

Pruning Apprch.	Variant	Wndw.	DSI GPU↓ %↑	Hours	Epch.	Steps (k)	MCC↑ GUE	NT-18
Baseline	Unprun.	-	0	1012	47	500	<u>71.6</u>	**69.8**
Ours	SP_20%	1	**77**	226	47	110	70.5	69.1
Ablation	SP_20%	2	<u>63</u>	330	47	180	70.5	<u>69.4</u>
	SP_20%	3	56	359	47	210	**71.7**	**69.8**
	SP_20%	4	51	433	47	237	71.3	**69.8**
	SP_20%	5	48	479	47	253	70.6	69.2

Effect of averaging window. Instead of pruning based on per-sequence pretraining accuracy from a single epoch, we average each sequence's accuracy over a window of consecutive epochs before applying the pruning criterion. Table 5 shows the trade-off for SP_20% on DB2–MSG. The one-window setting is identical to SP_20% in Table 2. Increasing the window stabilizes the pruning decision but reduces DSI. A window of three provides the clearest trade-off: DSI decreases to 56%, while performance on

both benchmarks exceeds that of the unpruned DB2 baseline. Thus, windowed accuracy provides an operating mode when stronger downstream performance is preferred over maximizing data savings.

DSI as control parameter. For a fixed τ_{low}, the achieved DSI is known only after the run. Across our variants, however, DSI follows a similar trajectory: it rises rapidly early in pretraining and then gradually flattens. We use this empirical pattern to define a default target DSI curve, turning DSI from an observed outcome into a controllable parameter (Figure 5). The curve specifies the target DSI after each epoch and, consequently, how many sequences may remain active. At each epoch boundary, we rank the remaining sequences by their per-sequence pretraining accuracy and remove the highest-scoring sequences until the target DSI is reached. Thus, the DSI curve determines how much to prune, while accuracy determines which sequences to remove.

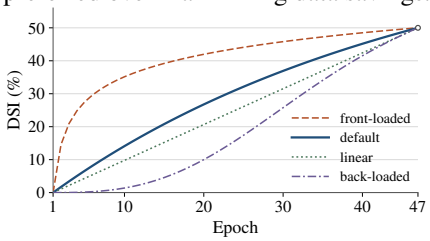

Figure 5: Four pruning budgets that all reach DSI $= 50\%$ at epoch 47 by different routes. The default is the trajectory measured in our SP runs on DB2–MSG.

Table 6 shows that the default curve reproduces the SP results at comparable savings. To test whether DSI alone determines performance, we compare the default curve with three alternative curves that reach the same final DSI but distribute pruning differently across epochs. At DSI $=$ 50%, the default curve outperforms the unpruned baseline on both tasks while avoiding half of the baseline sequence processing. All three alternative schedules perform worse, showing that when sequences are removed matters in addition to how many are removed. Appendix H provides details.

Table 6: Effect of the DSI target and schedule on efficiency and downstream performance (DB2 on MSG).

Pruning Apprch.	Variant	DSI Schedule	DSI %↑	GPU↓ Hours	Epch.	Steps (k)	MCC↑ GUE	MCC↑ NT-18
Baseline	Unprun.	-	0	1012	47	500	71.6	69.8
Ablation	DSI_20%	default	20	699	47	396	71.8	69.4
Ablation	DSI_50%	default	50	450	47	247	**72.0**	**70.4**
Ablation	DSI_50%	front-loaded	50	418	47	247	71.3	69.0
Ablation	DSI_50%	back-loaded	50	448	47	247	69.7	69.7
Ablation	DSI_50%	linear	50	452	46	247	70.2	68.4
Ablation	DSI_80%	default	80	166	47	98	69.6	69.3

3.2 Why SP Works

Figure 2 shows that accuracy separates easy and hard sequences. Figure 6 then examines how pruning changes this separation in the same settings, comparing the DB2–MSG unpruned baseline with SP_20% and the Hyena–HRG unpruned baseline with SP_50%.

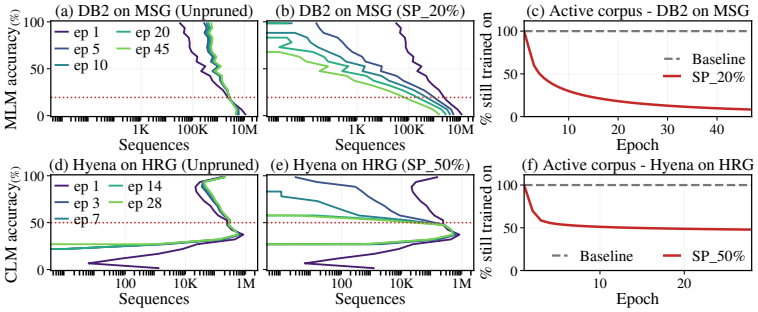

Figure 6: What pruning removes, and when. Panels (a) and (d) show how sequence counts vary across the accuracy range when nothing is pruned, and panels (b) and (e) show the same under sequence pruning. Panels (c) and (f) show how much of the corpus is still being pretrained on at each epoch. Dotted lines mark the τ_{low}.

The model's continued learning makes pruning dynamic. Although pruning removes high-accuracy sequences, prunable sequences do not run out after the first few epochs. Figure 6(c) and (f) show that the active corpus continues to shrink throughout pretraining. On DB2, epoch 1 removes 3.24% of the corpus, epoch 46 still removes 0.24%, and roughly 40% of all pruning occurs after epoch 10. This continued pruning arises because per-sequence accuracy gradually increases as the model learns to predict sequences better. As additional sequences cross τ_{low}, they become eligible for pruning in later epochs. Thus, although τ_{low} is fixed, pruning remains dynamic because the model's continued learning causes new sequences to cross the threshold over time. The same upward shift in accuracy explains why the number of sequences below τ_{low} decreases in the pruned distributions (Figure 6(b,e)), even though pruning directly removes only sequences above τ_{low}.

Pruning is safe at both extremes. If a sequence has temporarily low accuracy early in pretraining, when the model is still close to its initialization, it is simply retained because pruning acts only on high-accuracy sequences. As training progresses and its accuracy stabilizes, the sequence can still become eligible for pruning later. One pass over the corpus is largely sufficient for the sequence ranking to begin stabilizing. We verify this by measuring the rank correlation between sequence accuracy at a given epoch and its average accuracy over the final three epochs. Between the first and second epochs, this correlation rises from 0.36 to 0.63 for unpruned DB2 and from 0.75 to 0.93 for unpruned Hyena, and continues to increase thereafter (Appendix A.3). Thus, initially low accuracy does not permanently exclude a sequence from later pruning. At the other extreme, hard sequences remain far below τ_{low} throughout pretraining and are therefore never at risk of removal under any parameter setting we test. Sequences near τ_{low} are the main exception: a small epoch-to-epoch fluctuation (Figure 6(a,b)), rather than genuine improvement, may move a sequence into the pruning range. Averaging accuracy over several epochs before applying the pruning criterion reduces this effect, but at the cost of some data savings (Table 5).

4 Related work

Data pruning methods differ in their scoring signal, computational cost, and refresh frequency. Static methods score examples once using difficulty or uncertainty (Cho et al., 2025; Xia et al., 2023), influence or generalization (Tan et al., 2023; Yang et al., 2023), or information-theoretic criteria (Tan et al., 2025; Chen, 2025). Because these scores are not updated, their rankings may become stale as training progresses. Dynamic methods refresh scores during training (Kim & hwan Oh, 2025; Yang et al., 2025; Jin et al., 2026; Azeemi et al., 2024; Yu et al., 2025), with Zhou et al. (2026) showing that stale decisions can bias the retained distribution. Influence-based methods estimate an example's effect through gradients, influence functions, or Bayesian approximations (Yu et al., 2024; Nikdan et al., 2025; Gurbuz et al., 2025; Deng et al., 2023; Chhabra et al., 2024), but are expensive at scale and have been studied primarily for fine-tuning. Coreset and diversity methods instead optimize coverage (Tong et al., 2025; Cheng et al., 2025; Zhang et al., 2024; Nguyen et al., 2025; Li et al., 2025; Du et al., 2025), while another line of work targets label noise, calibration, or spurious correlations (Lin et al., 2023; Vysogorets et al., 2025; Park et al., 2023). These latter approaches rely on labels or class structure, which are unavailable in self-supervised genomic pretraining.

At pretraining scale, data selection typically operates over documents, batches, or fixed subsets to satisfy distributional or compute objectives (Xie et al., 2023; Yin & Rush, 2025; Bai et al., 2025; Zheng et al., 2025; Shum et al., 2025; He et al., 2026). These approaches often require a separate scoring procedure or reference model, and their selection decisions are made at a coarser granularity than individual sequences. To the best of our knowledge, none has been applied to DNA pretraining. SP instead dynamically prunes individual genomic sequences at every epoch boundary using a signal already produced by the model's forward pass, without requiring a separate reference model.

5 Conclusion and Limitations

We introduced data pruning to DNA pretraining. Per-sequence accuracy, obtained directly from the forward pass the model already performs, separates easy sequences from hard ones well enough to guide pruning. Removing easy sequences preserves most of the benefit of pretraining across five settings and both MLM and CLM objectives. Two control experiments further show that the gains depend on which sequences are removed, rather than on data reduction alone. The method is also simple to adopt: it requires no changes to the model architecture, pretraining objective, or optimizer.

Our method requires completing an epoch before pruning can begin, so extending it to single-pass pretraining would require additional changes. One possibility is to begin with a smaller corpus, allowing the same compute budget to cover multiple passes and giving the criterion an epoch boundary at which to act. Another is to make pruning decisions before a sequence is passed through the model. Our framework also treats pruning criteria as alternatives; combining them, such that a sequence is removed only when multiple criteria agree, is a natural direction for future work. Another direction is to adapt the learning-rate schedule to the reduced pretraining budget, allowing pruned runs to reach lower learning rates and potentially improve downstream performance at the same savings. Finally, the biological properties shared by pruned sequences remain an open question. Does the pruning criterion identify genuine redundancy in the genome, or does it instead reveal that DNA pretraining continues to spend computation on sequences the model has already learned?

AI USE STATEMENT

We used large language models for writing assistance only, including grammar and phrasing. No part of the methodology, experiments, analysis, or code was generated by AI tools.

REPRODUCIBILITY STATEMENT

All corpora and benchmarks used here are publicly available. Pretraining configurations are in Appendix C, downstream protocols in Appendix D, and implementations of the compared methods in Appendix B.2. All primary experiments use seed 42; Appendix G reports additional multi-seed variance experiments.

ETHICS STATEMENT

This work uses publicly available reference genomes and established benchmarks. It involves no human subjects and no private or identifiable data.

REFERENCES

Md Mustakin Alam, Shaker Islam, and Aminul Islam. Batch pruning by activation stability. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=TUADW7db5n.

Abdul Hameed Azeemi, Ihsan Ayyub Qazi, and Agha Ali Raza. Language model-driven data pruning enables efficient active learning, 2024. URL https://arxiv.org/abs/2410.04275.

Tianyi Bai, Ling Yang, Zhen Hao Wong, Fupeng Sun, Jiahui Peng, Xinlin Zhuang, Chi Zhang, Lijun Wu, Jiantao Qiu, Wentao Zhang, Binhang Yuan, and Conghui He. Efficient pretraining data selection for language models via multi-actor collaboration, 2025. URL https://arxiv.org/abs/2410.08102.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. ICML '09, pp. 41–48, New York, NY, USA, 2009. Association for Computing Machinery. ISBN 9781605585161. doi: 10.1145/1553374.1553380. URL https://doi.org/10.1145/1553374.1553380.

Sam Boshar, Benjamin Evans, Ziqi Tang, Armand Picard, Yanis Adel, Franziska K. Lorbeer, Chandana Rajesh, Tristan Karch, Shawn Sidbon, David Emms, Javier Mendoza-Revilla, Fatimah Al-Ani, Evan Seitz, Yair Schiff, Yohan Bornachot, Ariana Hernandez, Marie Lopez, Alexandre Laterre, Karim Beguir, Peter Koo, Volodymyr Kuleshov, Alexander Stark, Bernardo P. de Almeida, and Thomas Pierrot. A foundational model for joint sequence-function multi-species modeling at scale for long-range genomic prediction. *bioRxiv*, 2025. doi: 10.64898/2025.12.22.695963. URL https://www.biorxiv.org/content/early/2025/12/25/2025.12.22.695963.

Garyk Brixi, Matthew G. Durrant, Jerome Ku, Mohsen Naghipourfar, Michael Poli, Gwanggyu Sun, Greg Brockman, Daniel Chang, Alison Fanton, Gabriel A. Gonzalez, Samuel H. King, David B. Li, Aditi T. Merchant, Eric Nguyen, Chiara Ricci-Tam, David W. Romero, Jonathan C. Schmok, Ali Taghibakhshi, Anton Vorontsov, Brandon Yang, et al. Genome modelling and design across all domains of life with evo 2. *Nature*, 652(8112):1349–1361, 2026. doi: 10.1038/s41586-026-10176-5.

Jianting Chen. Efficient core-set selection for deep learning through squared loss minimization. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 8065–8076. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/chen25q.html.

Zihao Cheng, Binrui Wu, Zhiwei Li, Yuesen Liao, Su Zhao, Shuai Chen, Yuan Gao, and WEIZHONG ZHANG. Efficient representativeness-aware coreset selection. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=CrdrVdXCui.

Anshuman Chhabra, Peizhao Li, Prasant Mohapatra, and Hongfu Liu. "what data benefits my classifier?" enhancing model performance and interpretability through influence-based data selection. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=HE9eUQlAvo.

Yeseul Cho, Baekrok Shin, Changmin Kang, and Chulhee Yun. Lightweight dataset pruning without full training via example difficulty and prediction uncertainty, 2025. URL https://arxiv.org/abs/2502.06905.

Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, et al. Nucleotide transformer: Building and evaluating robust foundation models for human genomics. *Nature Methods*, 22:287–297, 2025. doi: 10.1038/s41592-024-02523-z.

Zhijie Deng, Peng Cui, and Jun Zhu. Towards accelerated model training via bayesian data selection, 2023. URL https://arxiv.org/abs/2308.10544.

Yupei Du, Yingjin Song, Hugh Mee Wong, Daniil Ignatev, Albert Gatt, and Dong Nguyen. Disentangling the roles of representation and selection in data pruning, 2025. URL https://arxiv.org/abs/2507.03648.

ENCODE Project Consortium. Expanded encyclopaedias of DNA elements in the human and mouse genomes. *Nature*, 583(7818):699–710, 2020. doi: 10.1038/s41586-020-2493-4.

Veniamin Fishman, Yuri Kuratov, Aleksei Shmelev, Maxim Petrov, Dmitry Penzar, Denis Shepelin, Nikolay Chekanov, Olga Kardymon, and Mikhail Burtsev. Gena-lm: a family of open-source foundational dna language models for long sequences. *Nucleic Acids Research*, 53(2):gkae1310, 01 2025. ISSN 1362-4962. doi: 10.1093/nar/gkae1310. URL https://doi.org/10.1093/nar/gkae1310.

Mustafa Burak Gurbuz, Xingyu Zheng, and Constantine Dovrolis. Peaks: Selecting key training examples incrementally via prediction error anchored by kernel similarity, 2025. URL https://arxiv.org/abs/2504.05250.

Bowei He, Lihao Yin, Hui-Ling Zhen, Xiaokun Zhang, Mingxuan Yuan, and Chen Ma. Paser: Post-training data selection for efficient pruned large language model recovery, 2026. URL https://arxiv.org/abs/2502.12594.

Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics*, 37(15):2112–2120, 08 2021. ISSN 1367-4803. doi: 10.1093/bioinformatics/btab083. URL https://doi.org/10.1093/bioinformatics/btab083.

Chenhan Jin, Shengze Xu, Qingsong Wang, Fan JIA, Dingshuo Chen, and Tieyong Zeng. OrderDP: A theoretically guaranteed lossless dynamic data pruning framework. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=e77QyyRQPz.

Gyu Yeol Kim and Min hwan Oh. Adam optimization with adaptive batch selection, 2025. URL https://arxiv.org/abs/2512.06795.

Eric S. Lander, Lauren M. Linton, Bruce Birren, Chad Nusbaum, Michael C. Zody, Jennifer Baldwin, Keri Devon, Ken Dewar, Michael Doyle, William FitzHugh, et al. Initial sequencing and analysis of the human genome. *Nature*, 409(6822):860–921, 2001. doi: 10.1038/35057062.

Jiazheng Li, Lu Yu, Qing Cui, Zhiqiang Zhang, Jun Zhou, Yanfang Ye, and Chuxu Zhang. Mass: Mathematical data selection via skill graphs for pretraining large language models, 2025. URL https://arxiv.org/abs/2503.14917.

Yong Lin, Chen Liu, Chenlu Ye, Qing Lian, Yuan Yao, and Tong Zhang. Optimal sample selection through uncertainty estimation and its application in deep learning, 2023. URL https://arxiv.org/abs/2309.02476.

Frederikke Isa Marin, Felix Teufel, Marc Horlacher, Dennis Madsen, Dennis Pultz, Ole Winther, and Wouter Boomsma. BEND: Benchmarking DNA language models on biologically meaningful tasks. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=uKB4cFNQFg.

Max Marion, Ahmet Üstün, Luiza Pozzobon, Alex Wang, Marzieh Fadaee, and Sara Hooker. When less is more: Investigating data pruning for pretraining llms at scale, 2023. URL https://arxiv.org/abs/2309.04564.

Dang Nguyen, Wenhan Yang, Rathul Anand, Yu Yang, and Baharan Mirzasoleiman. Mini-batch coresets for memory-efficient language model training on data mixtures, 2025. URL https://arxiv.org/abs/2407.19580.

Eric Nguyen, Michael Poli, Marjan Faizi, Armin W Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton M. Rabideau, Yoshua Bengio, Stefano Ermon, Christopher Re, and Stephen Baccus. HyenaDNA: Long-range genomic sequence modeling at single nucleotide resolution. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ubzNoJjOKj.

Eric Nguyen, Michael Poli, Matthew G. Durrant, Brian Kang, Dhruva Katrekar, David B. Li, Liam J. Bartie, Armin W. Thomas, Samuel H. King, Garyk Brixi, Jeremy Sullivan, Madelena Y. Ng, Ashley Lewis, Aaron Lou, Stefano Ermon, Stephen A. Baccus, Tina Hernandez-Boussard, Christopher Ré, Patrick D. Hsu, and Brian L. Hie. Sequence modeling and design from molecular to genome scale with evo. *Science*, 386(6723):eado9336, 2024. doi: 10.1126/science.ado9336. URL https://www.science.org/doi/abs/10.1126/science.ado9336.

Mahdi Nikdan, Vincent Cohen-Addad, Dan Alistarh, and Vahab Mirrokni. Efficient data selection at scale via influence distillation, 2025. URL https://arxiv.org/abs/2505.19051.

Dongmin Park, Seola Choi, Doyoung Kim, Hwanjun Song, and Jae-Gil Lee. Robust data pruning under label noise via maximizing re-labeling accuracy, 2023. URL https://arxiv.org/abs/2311.01002.

Aman Patel, Arpita Singhal, Austin Wang, Anusri Pampari, Maya Kasowski, and Anshul Kundaje. DART-eval: A comprehensive DNA language model evaluation benchmark on regulatory DNA. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2024. URL https://openreview.net/forum?id=qR0x6H5WUX.

Mansheej Paul, Surya Ganguli, and Gintare Karolina Dziugaite. Deep learning on a data diet: Finding important examples early in training. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=Uj7pF-D-YvT.

Ziheng Qin, Kai Wang, Zangwei Zheng, Jianyang Gu, Xiangyu Peng, xu Zhao Pan, Daquan Zhou, Lei Shang, Baigui Sun, Xuansong Xie, and Yang You. Infobatch: Lossless training speed up by unbiased dynamic data pruning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=C61sk5LsK6.

Kashun Shum, Yuzhen Huang, Hongjian Zou, Qi Ding, Yixuan Liao, Xiaoxin Chen, Qian Liu, and Junxian He. Predictive data selection: The data that predicts is the data that teaches, 2025. URL https://arxiv.org/abs/2503.00808.

Haoru Tan, Sitong Wu, Fei Du, Yukang Chen, Zhibin Wang, Fan Wang, and Xiaojuan Qi. Data pruning via moving-one-sample-out, 2023. URL https://arxiv.org/abs/2310.14664.

Haoru Tan, Sitong Wu, Wei Huang, Shizhen Zhao, and Xiaojuan Qi. Data pruning by information maximization, 2025. URL https://arxiv.org/abs/2506.01701.

Ruilin Tong, Yuhang Liu, Javen Qinfeng Shi, and Dong Gong. Coreset selection via reducible loss in continual learning. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=mAztx8QO3B.

Kirill Vishniakov, Karthik Viswanathan, Aleksandr Medvedev, Praveenkumar Kanithi, Marco AF Pimentel, Ronnie Rajan, and Shadab Khan. Tokenization to transfer: Do genomic foundation models learn good representations? In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=4UY1NHG5Ge.

Artem Vysogorets, Kartik Ahuja, and Julia Kempe. Drop: Distributionally robust data pruning, 2025. URL https://arxiv.org/abs/2404.05579.

Yifan Wu, Jiyue Jiang, Xichen Ye, Yiqi Wang, Chang Zhou, Yitao Xu, Jiayang Chen, He Hu, Weizhong Zhang, Cheng Jin, Jiao Yuan, and Yu Li. Investigating data pruning for pretraining biological foundation models at scale, 2026. URL https://arxiv.org/abs/2512.12932.

Xiaobo Xia, Jiale Liu, Jun Yu, Xu Shen, Bo Han, and Tongliang Liu. Moderate coreset: A universal method of data selection for real-world data-efficient deep learning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=7D5EECbOaf9.

Sang Michael Xie, Shibani Santurkar, Tengyu Ma, and Percy Liang. Data selection for language models via importance resampling, 2023. URL https://arxiv.org/abs/2302.03169.

Shuo Yang, Zeke Xie, Hanyu Peng, Min Xu, Mingming Sun, and Ping Li. Dataset pruning: Reducing training data by examining generalization influence, 2023. URL https://arxiv.org/abs/2205.09329.

Suorong Yang, Peng Ye, Furao Shen, and Dongzhan Zhou. When dynamic data selection meets data augmentation, 2025. URL https://arxiv.org/abs/2505.03809.

Junjie Oscar Yin and Alexander M. Rush. Compute-constrained data selection, 2025. URL https://arxiv.org/abs/2410.16208.

Zichun Yu, Spandan Das, and Chenyan Xiong. Mates: Model-aware data selection for efficient pretraining with data influence models, 2024. URL https://arxiv.org/abs/2406.06046.

Zichun Yu, Fei Peng, Jie Lei, Arnold Overwijk, Wen tau Yih, and Chenyan Xiong. Group-level data selection for efficient pretraining, 2025. URL https://arxiv.org/abs/2502.14709.

Chi Zhang, Huaping Zhong, Kuan Zhang, Chengliang Chai, Rui Wang, Xinlin Zhuang, Tianyi Bai, Jiantao Qiu, Lei Cao, Ju Fan, Ye Yuan, Guoren Wang, and Conghui He. Harnessing diversity for important data selection in pretraining large language models, 2024. URL https://arxiv.org/abs/2409.16986.

Haizhong Zheng, Elisa Tsai, Yifu Lu, Jiachen Sun, Brian R. Bartoldson, Bhavya Kailkhura, and Atul Prakash. Elfs: Label-free coreset selection with proxy training dynamics, 2025. URL https://arxiv.org/abs/2406.04273.

Qing Zhou, Tao Yang, Bingxuan Zhao, Hongyuan Zhang, Junyu Gao, and Qi Wang. Inconsistency biases in dynamic data pruning. In *The Fourteenth International Conference on Learning Representations*, 2026. URL https://openreview.net/forum?id=Zw1Uw7u6Su.

Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana V Davuluri, and Han Liu. DNABERT-2: Efficient foundation model and benchmark for multi-species genomes. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oMLQB4EZE1.

A Per-Sequence Accuracy as a Difficulty Signal

This appendix provides additional evidence supporting the claims associated with Figure 1 and 2. For both unpruned baselines, we track every sequence in the corpus at every epoch: 32.4M MSG sequences for DB2 over 47 epochs and 2.7M HRG sequences for Hyena over 28 epochs. When comparing early and late training windows, we use the first and last twenty epochs for DB2 and the first and last ten epochs for HyenaDNA, reflecting the different run lengths.

Table 7: Per-sequence accuracy across the unpruned baselines, over every sequence in each corpus. Fluctuation is the standard deviation of a sequence's accuracy across epochs, reported as the median over sequences. Accuracy is bounded by 0 and 1 in both settings.

	DB2–MSG (MLM)	Hyena–HRG (CLM)
Early window, mean accuracy	0.233	0.530
Late window, mean accuracy	0.274	0.551
Change in mean	$+0.042$	$+0.020$
Sequences that improved	74.8%	84.2%
Fluctuation across epochs	0.096	0.009

A.1 Sequences Stay Within a Band

A sequence's accuracy changes slightly from one epoch to the next, as also shown in Figure 1. On a 0–1 accuracy scale, the median fluctuation in per-sequence accuracy is 0.096 for DB2 and 0.009 for Hyena (Table 7). These fluctuations are small enough that a sequence typically remains within the same accuracy band throughout pretraining.

A.2 Continued Training Improves Accuracy Slightly

From the early to the late window, mean accuracy increases by 4.2 percentage points for DB2 and 2.0 percentage points for Hyena (Table 7). Accuracy improves for 74.8% and 84.2% of sequences, respectively, showing that both models continue to learn throughout pretraining. However, the magnitude of this improvement is small, so sequences generally remain within similar accuracy bands. As a result, accuracy continues to separate easy sequences from hard ones even as the model improves.

Table 8: Spearman rank correlation between per-sequence accuracy at a single epoch and each sequence's eventual difficulty, taken as its mean accuracy over the final three epochs.

Epoch	DB2 (MLM) ρ	Hyena (CLM) ρ
1	0.36	0.75
2	0.63	0.93
3	0.68	0.96
5	0.72	0.98
10	0.77	0.99
20	0.81	1.00

A.3 Accuracy Recovers the Eventual Ordering Early

Beyond separating easy sequences from hard ones, per-sequence pretraining accuracy also ranks sequences by difficulty, and this ordering becomes reliable early enough to guide pruning. We use each sequence's mean accuracy over the final three epochs as its reference difficulty. We then compare this ranking with the ranking obtained from a single earlier epoch using Spearman's rank correlation ρ. A value of 1 indicates identical rankings, while 0 indicates no relation between them. Because this measure depends only on the relative ordering of sequences, the overall increase in accuracy during pretraining does not affect it.

Both settings exhibit the same pattern (Table 8). At epoch 1, accuracy provides only a weak estimate of the eventual ordering, consistent with the model still being close to its initialization. After one additional pass over the corpus, agreement increases substantially and continues to strengthen as pretraining proceeds. By the time most pruning occurs, the ordering from a single epoch already closely agrees with the eventual ordering, supporting its use as the pruning criterion.

A.4 The Two Objectives Differ in Measurement Precision

The stability measures in Appendix A are consistently stronger for HyenaDNA than for DB2: the median fluctuation is 0.009 compared with 0.096, and the rank correlations are consistently higher. The same pattern appears in Figure 1, where the trajectories in panel (b) are nearly flat, while those in panel (a) fluctuate around their typical levels. CLM accuracy is computed over every prediction position in a sequence, whereas MLM accuracy uses only the masked positions, which may comprise only a few positions in a short sequence. Moreover, the same sequence receives a different mask at each epoch, so its MLM accuracy may be computed from different positions over time. CLM therefore uses more prediction positions per sequence and yields a more stable accuracy estimate. The two objectives differ in how precisely per-sequence accuracy is measured, not in whether accuracy separates easy and hard sequences.

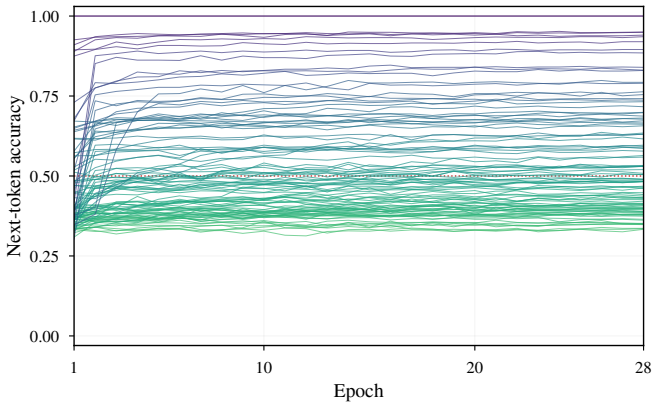

Figure 7: 100 randomly sampled sequences from the Hyena–HRG unpruned baseline. Colour encodes each sequence's mean accuracy. The dotted line marks 0.5 for reference.

Because Hyena trajectories are highly stable, 100 randomly sampled sequences can be displayed together while remaining easy to distinguish. Figure 7 shows that each sequence remains at a similar accuracy level throughout pretraining, with most crossings occurring during the first few epochs, when the model is still close to its initialization.

B Additional Method Details

This appendix provides additional methodological details for the pruning framework illustrated in Figure 3. We use the same notation as in Section 2: N is the number of sequences, p indexes a sequence, I is the scheduled number of epochs, i indexes an epoch, and $\mathcal{A}_i \subseteq \{1, \ldots, N\}$ is the active corpus entering epoch i. The batch size is b, and ℓ denotes the number of model blocks. For sequence p, $M_{p,\text{total}}^{(i)}$ denotes the number of scored positions in epoch i, and $\text{Acc}_p^{(i)}$ denotes its per-sequence accuracy when $M_{p,\text{total}}^{(i)} > 0$. We first provide additional details of the SP framework, followed by InfoBatch and BPAS, and then the perplexity and EL2N variants.

B.1 SP Framework Details

The main method defines the pruning criterion; this subsection provides the additional details needed to maintain $\mathcal{A}_i$ across epochs and handle sequences for which no valid score is available. Sequence identifiers are preserved across epochs so that each sequence can be tracked as $\mathcal{A}_i$ changes and batches are reconstructed. This ensures that sequences are correctly retained or excluded after each pruning decision. Once sequence p is removed, it does not return to $\mathcal{A}_{i+1}$ or any later active corpus.

At each epoch, the sequences in $\mathcal{A}_i$ are shuffled and regrouped into batches of size b. If $|\mathcal{A}_i|$ is not divisible by b, the remaining sequences do not form a complete batch and are therefore not processed or scored in epoch i. Because $\mathcal{A}_i$ is reshuffled and rebatched at the next epoch, the sequences in this remainder may differ across epochs. Thus, a sequence omitted in one epoch can be processed in a later epoch, preventing the same subset of the corpus from being systematically excluded from the pruning criterion. Under MLM, a processed sequence may also receive no `[MASK]` positions, in which case $M_{p,\text{total}}^{(i)} = 0$ and $\text{Acc}_p^{(i)}$ is undefined. Such sequences are excluded from the pruning decision for epoch i and reconsidered in a later epoch.

B.2 Comparison Methods and Alternative Pruning Criteria

We run InfoBatch and BPAS using the same corpus, step budget, and termination rule as SP in the corresponding pretraining setting. Perplexity and EL2N instead replace the pruning criterion used in block B of Figure 3, while the remaining SP procedure and step budget remain unchanged.

B.2.1 InfoBatch

Following Qin et al. (2024), InfoBatch dynamically scores sequence p by its mean cross-entropy loss over epoch i and temporarily drops low-loss sequences rather than removing them permanently. At each epoch, the mean of all N sequence scores is used as the threshold. Sequences at or above the threshold are always retained, while those below it are independently dropped with probability r. Selection is resampled at every epoch, so no sequence is permanently removed.

To correct the gradient bias introduced by dropping part of the low-loss group, the loss of retained sequences from that group is scaled by $1/(1-r)$, restoring the expected full-data gradient. Pruning is applied only during the first δI epochs; afterward, all N sequences are pretrained without rescaling. Scores are initialized to 1 before epoch 1, and the score of sequence p is updated only in epochs in which p is processed; otherwise, its previous score is retained. Table 16 gives (r, δ) for the InfoBatch variants.

B.2.2 BPAS

Following Alam et al. (2026), BPAS partitions the N sequences into fixed global batches before pretraining begins. Each sequence remains assigned to the same batch throughout the run. Sequences are shuffled within each batch, and the order of batches is shuffled, but sequences never move between batches. The trailing incomplete batch is dropped so that all global batches have the same size.

For each batch and transformer block, we flatten the block output and record its standard deviation. Averaging these values over the ℓ blocks produces one statistic per batch at epoch i. From epoch 2 onward, each active batch is compared with its statistic from epoch $i-1$ and pruned when the change falls below $\delta(i) = \delta_s e^{\alpha i}$, where $\alpha = \frac{1}{I}\ln(\delta_e/\delta_s)$. The threshold is more restrictive early in pretraining, when representations are changing rapidly, and relaxes as pretraining progresses. Once a batch is pruned, all of its sequences are permanently removed, causing $\mathcal{A}_i$ to shrink monotonically. Table 16 gives (δ_s, δ_e) for each BPAS variant.

B.2.3 Perplexity

Following Marion et al. (2023), perplexity scores sequence p at epoch i as the exponential of the mean negative log-likelihood over its prediction positions. Perplexity is bounded below by 1, attained when every position is predicted with certainty, and is unbounded above.

Low perplexity therefore identifies sequences that are already predicted well. The low threshold is fixed at 1, while the high threshold varies across variants; sequence p is pruned when its perplexity is at or below the high threshold. The score is obtained from the same forward pass, pruning is applied at the epoch boundary, and removed sequences remain excluded from subsequent $\mathcal{A}_i$. A sequence without a valid score in epoch i is excluded from that epoch's pruning decision and reconsidered in a later epoch. Table 16 gives the high threshold for each perplexity variant.

B.2.4 EL2N

Following Paul et al. (2021), EL2N scores sequence p at epoch i by the ℓ_2 norm of its prediction error. At each masked position, we compute the norm between the predicted softmax distribution and the one-hot target and average these values over the masked positions of p. The score lies in $[0, \sqrt{2}]$, where 0 corresponds to a perfect prediction, a value near 1 to a uniform prediction over the vocabulary, and $\sqrt{2}$ to a confident incorrect prediction.

Low EL2N therefore identifies sequences that are already predicted well. Its threshold orientation is opposite to that of accuracy: accuracy fixes $\tau_{\text{high}} = 1$ and varies τ_{low}, whereas EL2N fixes its low threshold at 0 and varies its high threshold. The score is obtained from the same forward pass, pruning is applied at the epoch boundary, and removed sequences remain excluded from subsequent

$\mathcal{A}_i$. If $M^{(i)}_{p,\text{total}} = 0$, sequence p is excluded from the pruning decision for epoch i and reconsidered in a later epoch. Table 16 gives the high threshold for each EL2N variant.

C Pretraining Details

Table 9 gives the optimizer and learning-rate schedule for every setting. All runs train in bf16 with gradient clipping at 1.0. The subsections below give the corpus, tokenizer, and model configuration for each setting in turn.

Table 9: Optimizer hyperparameters and learning-rate schedules for each pretraining setting. Hyena denotes the 1,024-bp Hyena setting.

	DB2 MSG	DB2 HRG	NT-500M HRG	Hyena HRG	Hyena-16k HRG
Optimizer	AdamW	AdamW	Adam	AdamW	AdamW
Peak LR	5×10^{-4}	5×10^{-4}	1×10^{-4}	6×10^{-4}	6×10^{-4}
LR Schedule	linear	linear	inverse sqrt	cosine	cosine
Warmup steps	30,000	1,800	16,000	500	500
Adam β	(0.9, 0.98)	(0.9, 0.98)	(0.9, 0.999)	(0.9, 0.999)	(0.9, 0.999)
Adam ϵ	10^{-6}	10^{-6}	10^{-8}	10^{-8}	10^{-8}
Weight decay	10^{-5}	10^{-5}	0	0.1	0.1

C.1 DB2 on MSG

The multi-species genome collection, segmented into sequences of roughly 1 kb. Tokenization uses byte-pair encoding with a 4,096-token vocabulary, under which a 1 kb sequence becomes roughly 128 tokens. Ambiguous bases are dropped during corpus preparation. The model is the released 117M-parameter DB2 architecture, pretrained from scratch with 15% MLM masking under the standard $80/10/10$ corruption rule.

C.2 DB2 on HRG

The human reference genome, segmented into 1 kb sequences and prepared with the same pipeline as MSG: the same byte-pair tokenizer and vocabulary size, and ambiguous bases dropped. The model and masking are identical to Appendix C.1. This setting differs from that one only in corpus, which is what isolates the effect of the data.

C.3 NT-500M on HRG

The HRG, segmented into 6,100 bp windows with 50 bp of overlap on each side, following the NT-500M protocol. Tokenization uses non-overlapping 6-mers with a 4,105-token vocabulary. Ambiguous bases are retained, since the vocabulary carries an explicit `N`. The model follows the NT-500M-HumanRef configuration, with 24 layers, hidden size 1280, 20 attention heads, and intermediate size 5120, pretrained from scratch with 15% masking under the same $80/10/10$ rule. Its learning rate warms up linearly from 5×10^{-5} to peak, then decays as the inverse square root of the step count with a floor at one percent of peak.

C.4 Hyena on HRG

The human reference genome, segmented into fixed windows of 1,024 bp for Hyena and 16,384 bp for Hyena-16k. Tokenization is single-character with a 12-token vocabulary, so tokens and base pairs correspond one to one, and ambiguous bases are retained as an explicit `N` token. Both settings use the 8-layer, width-256 architecture with Hyena order 2 and implicit filter order 64, pretrained from scratch on next-token prediction. They share an optimizer and differ only in context length and step budget. Parameter counts differ slightly, 6.6M at 1,024 bp and 7.2M at 16,384 bp, because each Hyena filter carries a learned positional embedding sized to the context length. Every other weight is identical between the two. We hold the context fixed at the stated length rather than applying the sequence-length warmup of the original recipe. This keeps a sequence the same length at every epoch, so its per-sequence accuracy stays comparable across epochs.

D DOWNSTREAM PROTOCOLS

Our downstream evaluation uses two protocols: full-parameter fine-tuning for GUE, NT-18, and GB, and frozen probing for B-GF and D-T4. This appendix provides the training and evaluation details for both settings.

D.1 FINE-TUNED BENCHMARKS

Table 10: Fine-tuning settings by benchmark.

	GUE	NT-18	GB
Pretrained models	DB2, NT-500M	DB2, NT-500M	Hyena
Tasks	27 tasks, 6 groups	18 tasks, 3 groups	7 tasks
Learning rate	3×10^{-5}	3×10^{-5}	6×10^{-4}
Batch size	16	16	128
Weight decay	0.01	0.01	0.1
Warmup	100 steps	100 steps	1% of steps
Epochs	per task	per task	20

We follow each benchmark's published fine-tuning protocol and we omit the optional Low-Rank Adaptation component and update all parameters. Table 10 gives the settings shared across tasks within a benchmark, and Table 11 the per-task epochs and maximum sequence lengths. Lengths follow each task's own sequence-length distribution, so they differ between tokenizers at equal base-pair length. Every run uses seed 42, and we evaluate the best checkpoint by validation performance: lowest validation loss on GUE and GB, and highest validation MCC on NT-18.

Table 11: Per-task training epochs and maximum token lengths for the fine-tuned benchmarks. TF denotes transcription factor.

Task group	Task	Epochs	Maximum token length	
			DB2	NT-500M
GUE				
Epigenetic marks	10 histone datasets	3	132	112
TF binding, mouse	5 datasets	30	47	24
TF binding, human	5 datasets	3	29	24
Promoter	all	4	77	64
	no TATA	4	77	64
	TATA	10	82	64
Core Promoter	all	4	21	16
	no TATA	4	21	16
	TATA	10	27	20
Splice site	reconstruction	5	110	96
NT-18				
Regulatory	promoter (all)	4	100	64
	promoter (no TATA)	4	100	64
	promoter (TATA)	10	100	64
	enhancers	4	128	96
	enhancer types	4	128	96
Splice site	all	5	192	128
	acceptors	5	192	128
	donors	5	192	128
Chromatin	10 histone datasets	3	256	192
GB				Hyena
demo_coding_vs_intergenomic_seqs		20		202
demo_human_or_worm		20		202
human_nontata_promoters		20		253
human_enhancers_cohn		20		502
human_enhancers_ensembl		20		550
human_ocr_ensembl		20		579
human_ensembl_regulatory		20		802

D.2 Frozen probing benchmarks

On BEND gene finding and DART-Eval Task 4 the pretrained model is used only as a feature extractor. We follow each benchmark's released protocol without modification: embeddings are computed in a single forward pass with all pretrained weights frozen, and only the benchmark's own probe is trained on them. For BEND this is a nine-class convolutional probe over per-nucleotide embeddings, evaluated by MCC on the test split. For DART-Eval Task 4, it is a convolutional regression probe trained on peak, non-peak, and IDR regions of the K562 cell line and evaluated on held-out chromosomes by Spearman correlation.

E Full Pruning Sweeps

This section reports the full pruning sweeps for each pretraining setting, including variants omitted from Section 3.1. The tables show how different pruning thresholds affect data savings, pretraining cost, and downstream performance. The full sweeps also reveal how far each method can be pushed before downstream performance deteriorates. For the encoder settings, SP remains effective over a wider range of DSI before downstream performance deteriorates. On DB2–MSG, BPAS degrades sharply beyond roughly 55% DSI, whereas SP remains competitive with the unpruned baseline at 77.77% DSI (Table 12). Across the encoder sweeps, the more aggressive SP_10% setting marks the point where downstream performance begins to drop substantially.

The Hyena sweep shows a different pattern. BPAS performs better at low DSI, but the performance gap narrows as DSI increases. SP_50% achieves the best downstream result among the SP variants and exceeds the unpruned baseline while using roughly half of the baseline pretraining data. The sweep also exposes a structural limit of BPAS. Because BPAS measures the change in activation statistics between consecutive epochs, it requires two completed epochs before pruning can begin. These first two full epochs cap its maximum DSI at 92.86%. SP can prune after the first epoch and reaches 96.42% DSI while remaining close to the unpruned baseline (Table 15).

Table 12: Detailed pruning sweep for DB2 pretraining on MSG.

Pruning Apprch.	Variant	Threshold Low	Threshold High	DSI (%)↑	GPU↓ Hours	Epch.	Steps	MCC↑ GUE	MCC↑ NT-18
Baseline	Unpruned	-	-	0	1012.13	47	500k	<u>71.58</u>	**69.79**
	Full pruned	-	-	100	0	0	0	60.57	47.12
Related Work	BPAS_1	2×10^{-6}	2×10^{-5}	6.67	977.99	47	470k	71.31	68.83
	BPAS_2	1×10^{-5}	1×10^{-4}	19.81	777.84	47	400k	71.17	<u>69.59</u>
	BPAS_3	7×10^{-5}	7×10^{-4}	34.61	652.99	35	320k	71.02	68.70
	BPAS_4	1×10^{-4}	1×10^{-3}	36.25	838.74	38	310k	71.03	68.59
	BPAS_5	2×10^{-4}	2×10^{-3}	40.72	752.49	33	290k	70.67	68.67
	BPAS_6	3×10^{-4}	3×10^{-3}	45.36	689.09	30	270k	70.19	68.34
	BPAS_7	4×10^{-4}	4×10^{-3}	46.35	666.24	29	270k	70.30	68.69
	BPAS_8	1×10^{-3}	1×10^{-2}	54.69	473.71	25	220k	70.12	68.16
	BPAS_9	5×10^{-3}	5×10^{-2}	71.23	318.55	24	140k	68.88	67.52
	BPAS_10	1×10^{-2}	1×10^{-1}	89.91	100.61	9	50k	66.30	63.34
Ours	SP_100%	100	100	5.53	994.88	47	480k	71.54	69.54
	SP_90%	90	100	8.05	946.69	47	460k	70.55	69.28
	SP_80%	80	100	12.07	905.06	47	430k	70.76	69.15
	SP_70%	70	100	16.67	834.11	47	410k	70.54	68.77
	SP_60%	60	100	23.14	786.63	47	380k	**72.00**	69.10
	SP_50%	50	100	32.42	713.61	47	330k	71.49	69.00
	SP_40%	40	100	42.29	494.22	47	280k	71.35	68.60
	SP_30%	30	100	57.19	438.88	47	210k	70.30	69.28
	SP_20%	20	100	77.77	226.51	47	110k	70.47	69.05
	SP_10%	10	100	93.20	72.52	47	30k	65.86	62.08
	SP_5%	5	100	96.65	30.12	47	16k	63.32	60.70
	SP_1%	1	100	96.76	27.80	47	15k	63.65	60.39
Ablation	RP_20%	-	-	77.97	215.92	47	110k	68.12	65.57
	LP_0%	0	0	55.84	378.18	47	230k	68.53	64.85

Table 13: Selected pruning configurations for DB2 pretraining on HRG.

Pruning Apprch.	Variant	Threshold Low	Threshold High	DSI (%)↑	GPU↓ Hours	Epch.	Steps	MCC↑ GUE	MCC↑ NT-18
Baseline	Unpruned	-	-	0	47.16	32	30k	**66.61**	64.14
	Full pruned	-	-	100	0	0	0	43.56	47.27
Related Work	BPAS_1	2×10^{-6}	2×10^{-5}	0.72	47.15	32	30k	65.75	62.72
	BPAS_4	1×10^{-4}	1×10^{-3}	29.63	32.95	31	21k	64.99	62.41
	BPAS_5	2×10^{-4}	2×10^{-3}	40.40	23.57	29	18k	64.37	62.83
	BPAS_6	3×10^{-4}	3×10^{-3}	59.21	19.24	25	12k	64.70	61.78
	BPAS_8	1×10^{-3}	1×10^{-2}	91.03	4.31	8	3k	62.42	61.82
	BPAS_9	5×10^{-3}	5×10^{-2}	93.75	2.99	2	2k	61.40	60.96
Ours	SP_100%	100	100	2.88	46.78	32	29k	65.20	63.80
	SP_80%	80	100	5.93	45.04	32	28k	65.42	**65.70**
	SP_50%	50	100	18.12	39.16	32	25k	<u>66.12</u>	63.57
	SP_20%	20	100	46.29	25.78	32	16k	65.20	63.24
	SP_10%	10	100	72.00	13.47	32	8k	62.44	60.17
	SP_5%	5	100	89.22	5.23	32	3k	61.54	60.09
Ablation	EL2N_1	0	0.70	15.93	39.91	32	25k	64.16	<u>64.70</u>
	EL2N_2	0	0.80	25.98	34.99	32	22k	65.32	64.61
	EL2N_3	0	0.90	37.65	32.57	32	19k	65.75	64.21
	EL2N_4	0	0.95	44.96	27.83	32	16k	64.98	63.03

Table 14: Detailed pruning sweep for NT-500M pretraining on HRG.

Pruning Apprch.	Variant	Threshold Low	Threshold High	DSI (%)↑	GPU↓ Hours	Epch.	Steps	MCC↑ GUE	MCC↑ NT-18
Baseline	Unpruned	-	-	0	1137.60	99	50k	66.98	**74.23**
	Full pruned	-	-	100	0	0	0	53.00	63.74
Related Work	BPAS_1	2×10^{-6}	2×10^{-5}	1.56	1087.51	99	49k	66.37	73.75
	BPAS_2	1×10^{-5}	1×10^{-4}	7.10	1033.89	99	46k	65.60	73.70
	BPAS_4	1×10^{-4}	1×10^{-3}	40.59	658.58	99	29k	65.61	73.90
	BPAS_8	1×10^{-3}	1×10^{-2}	73.71	284.79	56	13k	63.38	72.61
	BPAS_9	5×10^{-3}	5×10^{-2}	82.61	215.70	31	9k	61.14	71.43
Ours	SP_100%	100	100	4.78	1081.86	99	47k	67.37	73.90
	SP_80%	80	100	6.66	1063.21	99	46k	67.09	73.88
	SP_50%	50	100	10.31	969.95	99	44k	**68.19**	74.01
	SP_20%	20	100	37.94	668.83	99	30k	<u>67.44</u>	<u>74.12</u>
	SP_10%	10	100	65.41	379.00	99	17k	64.46	73.18

Table 15: Detailed pruning sweep for Hyena pretraining on HRG.

Pruning Apprch.	Variant	Threshold Low	Threshold High	DSI (%)↑	GPU↓ Hours	Epch.	Steps	MCC↑ GB
Baseline	Unpruned	-	-	0	76.41	28	50k	76.97
	Full pruned	-	-	100	0	0	0	73.27
Related Work	BPAS_1	2×10^{-6}	2×10^{-5}	0.001	75.20	28	49k	76.65
	BPAS_4	1×10^{-4}	1×10^{-3}	2.67	73.14	28	48k	77.18
	BPAS_6	3×10^{-4}	3×10^{-3}	9.26	68.09	28	45k	<u>77.31</u>
	BPAS_8	1×10^{-3}	1×10^{-2}	37.42	46.96	28	30k	76.83
	BPAS_9	5×10^{-3}	5×10^{-2}	63.52	27.54	23	18k	77.19
	BPAS_10	1×10^{-2}	1×10^{-1}	74.65	19.14	23	13k	**77.64**
	BPAS_11	1×10^{-1}	1×10^{0}	84.13	15.81	7	7.8k	76.70
	BPAS_12	7×10^{1}	1×10^{2}	92.86	4.65	2	4k	76.05
Ours	SP_100%	100	100	4.37	71.73	28	47k	76.42
	SP_80%	80	100	11.79	65.94	28	43k	76.37
	SP_50%	50	100	47.07	39.75	28	26k	<u>77.31</u>
	SP_30%	30	100	96.38	2.8	28	1.7k	75.37
	SP_20%	20	100	96.42	2.8	28	1.7k	76.11
	SP_10%	10	100	96.43	2.78	28	1.7k	75.28
Ablation	Perp_1	1.00	1.13	6.92	71.20	28	47k	75.85
	Perp_2	1.00	2.19	20.52	51.77	28	39k	76.47
	Perp_3	1.00	3.00	45.86	36.52	28	27k	76.70
	Perp_4	1.00	12.00	96.42	2.19	28	1.7k	75.36
	RP_20%	-	-	96.43	3.62	28	1.7k	74.81

F HYPERPARAMETERS AND COMPUTATION

Table 16 lists the variant-specific parameters used across the pretraining settings. For BPAS, (δ_s, δ_e) are the starting and ending values of its epoch-dependent activation-change threshold. For InfoBatch, (r, δ) are the drop probability and annealing fraction. SP, EL2N, Perp, and LP use low and high thresholds on the pruning score range; RP has no threshold parameters.

Table 16: Variant-specific parameters across pretraining settings.

Related Work			Ours			Ablation		
Variant	Parameters		Variant	Threshold		Variant	Threshold	
	δ_s/r	δ_e/δ		τ_{low}	τ_{high}		Low	High
BPAS_1	2×10^{-6}	2×10^{-5}	SP_100%	100	100	EL2N_1	0	0.70
BPAS_2	1×10^{-5}	1×10^{-4}	SP_90%	90	100	EL2N_2	0	0.80
BPAS_3	7×10^{-5}	7×10^{-4}	SP_80%	80	100	EL2N_3	0	0.90
BPAS_4	1×10^{-4}	1×10^{-3}	SP_70%	70	100	EL2N_4	0	0.95
BPAS_5	2×10^{-4}	2×10^{-3}	SP_60%	60	100	Perp_1	1.00	1.13
BPAS_6	3×10^{-4}	3×10^{-3}	SP_50%	50	100	Perp_2	1.00	2.19
BPAS_7	4×10^{-4}	4×10^{-3}	SP_40%	40	100	Perp_3	1.00	3.00
BPAS_8	1×10^{-3}	1×10^{-2}	SP_30%	30	100	Perp_4	1.00	12.00
BPAS_9	5×10^{-3}	5×10^{-2}	SP_20%	20	100	RP_20%	-	-
BPAS_10	1×10^{-2}	1×10^{-1}	SP_10%	10	100	LP_0%	0	0
BPAS_11	1×10^{-1}	1×10^{0}	SP_5%	5	100			
BPAS_12	7×10^{1}	1×10^{2}	SP_1%	1	100			
Info_1	0.5	0.875						
Info_2	0.7	0.90						

G RUN-TO-RUN VARIANCE

We separately measure variation introduced by downstream fine-tuning and by pretraining. This allows us to assess whether the observed performance and data-savings differences are robust to random seeds.

G.1 FINE-TUNING SEED VARIANCE

We fine-tune each checkpoint with three different random seeds while keeping the downstream settings fixed. Table 17 reports the mean and standard deviation of downstream MCC for the unpruned baseline and SP_20%. This isolates fine-tuning variation without changing the pretrained checkpoint.

Table 17: Fine-tuning seed variance. Each entry is the mean ± standard deviation over three fine-tuning seeds using a single pretrained checkpoint.

Corpus	Model	Variant	MCC↑		
			GUE	NT-18	GB
MSG	DB2	Unpruned	70.80 ± 0.52	68.98 ± 0.52	—
		SP_20%	70.22 ± 0.29	68.85 ± 0.18	—
HRG	DB2	Unpruned	66.29 ± 0.57	63.53 ± 0.60	—
		SP_20%	64.98 ± 0.36	62.99 ± 0.11	—
HRG	NT-500M	Unpruned	66.46 ± 0.52	74.10 ± 0.13	—
		SP_20%	66.98 ± 0.40	73.67 ± 0.14	—
HRG	Hyena	Unpruned	—	—	76.51 ± 0.50
		SP_20%	—	—	75.34 ± 0.31

Table 18: Pretraining seed variance for SP_20%. Each entry is the mean ± standard deviation over three independent pretraining runs, each fine-tuned once.

Corpus	Model	Variant	DSI (%)	MCC↑		
				GUE	NT-18	GB
MSG	DB2	SP_20%	77.61 ± 0.15	70.18 ± 0.28	68.60 ± 0.17	—
HRG	DB2		46.18 ± 0.13	64.34 ± 0.50	62.82 ± 0.59	—
HRG	NT-500M		37.85 ± 0.08	67.15 ± 0.25	73.77 ± 0.38	—
HRG	Hyena		96.42 ± 0.00	—	—	75.77 ± 0.34

G.2 Pretraining Seed Variance

We pretrain SP_20% with three different random seeds while keeping the pretraining settings fixed, then fine-tune each checkpoint once. Table 18 reports the mean and standard deviation of DSI and downstream MCC. We focus these repeats on SP_20% because its lower computation cost makes repeated pretraining substantially less expensive than for the unpruned baselines.

H DSI-Targeted Sequence Pruning

Let N be the number of sequences in the unpruned corpus and I the scheduled number of epochs, indexed by $i \in \{1, \dots, I\}$. Let $\mathcal{A}_i \subseteq \{1, \dots, N\}$ be the active set entering epoch i, with $\mathcal{A}_1 = \{1, \dots, N\}$, and let p index an individual sequence. Finally, let $I_s \leq I$ denote the last executed epoch. The running DSI through epoch i is

$$\mathrm{DSI}_i = 1 - \frac{\sum_{j=1}^{i} |\mathcal{A}_j|}{i\,N}, \tag{1}$$

the average fraction of the corpus left untouched through epoch i. Reading this equation backwards turns a target DSI curve into a pretraining budget. At the boundary after epoch i, the target DSI_{i+1} determines how many sequences may remain in the next epoch. We remove the sequences with the highest valid $\mathrm{Acc}_p^{(i)}$ from $\mathcal{A}_i$ until $\mathcal{A}_{i+1}$ meets that budget.

The default curve comes from measurement rather than choice. Across our SP runs, the retained fraction in epoch i is well described by the geometric decay

$$\frac{|\mathcal{A}_i|}{N} = \rho^{\,i-1}. \tag{2}$$

For a target final DSI D, we fit ρ so that the final DSI under the equation in section 3 equals D. The other three curves change when pruning occurs while keeping the same final target. Replacing the exponent $i-1$ with $(i-1)^{1/g}$ gives

$$\frac{|\mathcal{A}_i|}{N} = \rho^{(i-1)^{1/g}}, \tag{3}$$

where $g = 4$ gives the front-loaded curve and $g = 0.35$ gives the back-loaded curve. The linear curve instead uses

$$\frac{|\mathcal{A}_i|}{N} = 1 - 2m(i-1), \qquad m = \frac{D}{I-1}. \tag{4}$$

For $D = 0.5$ and $I = 47$, the resulting values are $\rho = 0.9657$ for the default curve, $\rho = 0.7086$ for the front-loaded curve, $\rho = 0.9999$ for the back-loaded curve, and $m = 0.0109$ for the linear curve. Under the linear schedule, $|\mathcal{A}_{47}| = 0$, so the last executed epoch is $I_s = 46$. The running DSI at that point is $\mathrm{DSI}_{46} = 0.489$, but the final DSI reported is 0.5 because the scheduled horizon remains $I = 47$.

During epoch i, we accumulate $M_{p,\mathrm{correct}}^{(i)}$ and $M_{p,\mathrm{total}}^{(i)}$, producing $\mathrm{Acc}_p^{(i)}$ as defined in the Equation in section 2. Sequences that are not processed because they fall outside a complete batch, or for which $M_{p,\mathrm{total}}^{(i)} = 0$, are ranked last and are therefore removed only when the budget cannot be met from the scored sequences alone, as happens when a schedule targets an empty active set. The learning-rate schedule remains tied to the original total step count and is never rescaled, so the pruned and unpruned runs remain comparable step for step.