DNACHUNKER: LEARNABLE TOKENIZATION FOR DNA LANGUAGE MODELS

Taewon Kim[1], Jihwan Shin[1], Hyomin Kim[1], Youngmok Jung[2], Jonghoon Lee[2], Won-Chul Lee[2], Sungsoo Ahn[1]†, Insu Han[1]†
[1]Korea Advanced Institute of Science and Technology (KAIST), [2]INOCRAS
{maxkim139, jihwan1205, hyomin126, insu.han, sungsoo.ahn}@kaist.ac.kr
{youngmok.jung, jonghoonlee, wclee47}@inocras.com

ABSTRACT

DNA language models have emerged as powerful tools for decoding the complex language of DNA sequences. However, the performance of these models is heavily affected by their tokenization strategy, i.e., a method used to parse DNA sequences into a shorter sequence of chunks. In this work, we propose DNACHUNKER, which integrates a learnable dynamic DNA tokenization mechanism and is trained as a masked language model. Adopting the dynamic chunking procedure proposed by Hwang et al. (2025), our model learns to segment sequences into variable-length chunks. This dynamic chunking offers two key advantages: it's resilient to shifts and mutations in the DNA, and it allocates more detail to important functional areas. We demonstrate the performance of DNACHUNKER by training it on the human reference genome (HG38) and testing it on the Nucleotide Transformer and Genomic benchmarks. Further ablative experiments reveal that DNACHUNKER learns tokenization that grasps biological "grammar" and uses smaller chunks to preserve detail in important functional elements such as promoters and exons, while using larger chunks for repetitive, redundant regions.

1 INTRODUCTION

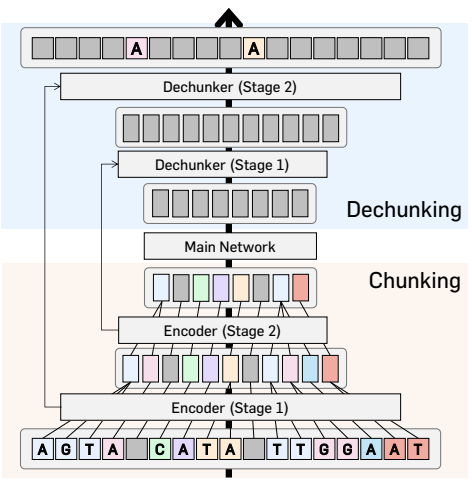
DNA sequences are the fundamental blueprint of life, containing the information that governs complex biological processes such as gene regulation (Moore et al., 2020), protein synthesis (Jia et al., 2024), DNA replication (Ekundayo & Bleichert, 2019), to name a few. Rapid advances in sequencing technology (Behjati & Tarpey, 2013) have made genomic data massively available. However, understanding and predicting the function encoded within these sequences remains a major challenge. The immense length and intricate nature of genomic data, along with a lack of high-quality, task-specific datasets, make it difficult to understand the underlying rules of this biological code.

Inspired by the success of large language models (LLMs; Anil et al., 2023), several recent works have begun investigating DNA language models (Ji et al., 2021; Sanabria et al., 2024; Dalla-Torre et al., 2025), moving beyond traditional rule-based methods to learn the "grammar" and "semantics" of DNA. In particular, the presence of long-range interactions between nucleotides and functional elements such as promoters and enhancers that act as "words" in the genomic language highlights the need for a tokenization strategy that can group DNA sequences into meaningful tokens.

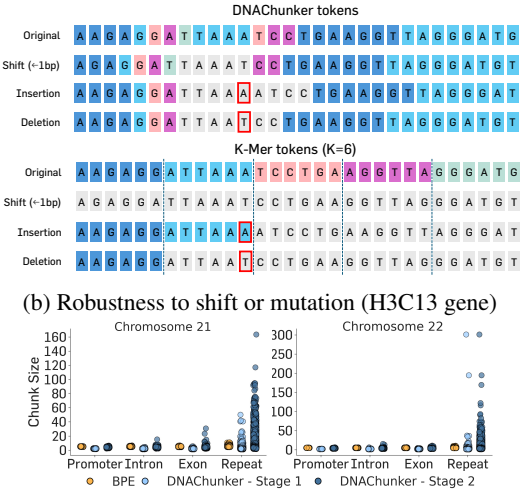
Genomic sequences pose unique challenges for tokenization that differ from natural language, primarily due to the absence of a natural "word" unit. Prior works have largely adopted one of three approaches: single nucleotides (Dalla-Torre et al., 2025; Schiff et al., 2024), fixed-size k-mers (Poli et al., 2023; Ji et al., 2021), or Byte-Pair Encoding (BPE) (Zhou et al., 2024). The single nucleotide approach, while simple, results in excessively long sequences that make it computationally expensive and difficult to model long-range interactions (Dalla-Torre et al., 2025).

To circumvent this length issue, fixed-size k-mers and BPE have been explored, but these methods are inherently fixed and struggle to adapt to the biological context of DNA. K-mer tokenization is highly sensitive to small shifts, where a single insertion, deletion, or mutation can completely alter

†Corresponding authors.



(a) Model architecture of DNACHUNKER



(b) Robustness to shift or mutation (H3C13 gene)

(c) Distribution of chunk size

Figure 1: **Architecture, tokenizer robustness, and distribution of chunk size.** (a) The architecture of DNACHUNKER. (b) Our tokenizer is robust against nucleotide-wise shifts or mutations, where we color the tokens to indicate that they are preserved despite the mutations. (c) Our DNACHUNKER dynamically represents functional elements (promoter, intron, exon) with high-resolution using smaller chunks, while compressing repetitive elements with larger chunks.

the tokenized output, even if the biological function remains unchanged (Dalla-Torre et al., 2025). Next, frequency-driven schemes like BPE fail to capture the functional importance of substrings, since the most frequent substrings are typically simple non-functional repetitive elements.

To this end, we propose DNACHUNKER, a bidirectional masked DNA language model designed to overcome the limitations of fixed tokenization (Figure 1a). Our model integrates a learnable, dynamic tokenizer that segments DNA into variable-length, context-dependent chunks, producing biologically meaningful groupings directly from sequence.

Concretely, we use a two-stage hierarchical encoder: raw base-pair embeddings are first processed with lightweight bidirectional Mamba layers (Schiff et al., 2024), then adjacent positions are merged into chunks using cosine-similarity–based boundary decisions, grouping highly similar representations into a single token. The resulting compressed sequence is modeled by a more expressive main network to capture long-range dependencies, and is then upsampled back to base-pair resolution via a bidirectional dechunking module that reconstructs fine-grained representations while leveraging intermediate encoder features. This architecture enables efficient long-context modeling while retaining nucleotide-level fidelity for masked prediction. With such design choices, our model is able to adaptively allocate chunk sizes to represent functional elements in high resolution, while compressing the non-functional repetitive elements with larger chunks (Figure 1c).

We validate DNACHUNKER by pretraining on the human reference genome (HG38) and fine-tuning on two standard suites: the Nucleotide Transformer benchmark (Dalla-Torre et al., 2025) and the Genomic Benchmarks collection (Grešová et al., 2023). Across these tasks, DNACHUNKER matches the performance of the state-of-the-art GENERATOR model (Wu et al., 2025) (1.2B parameters) while using 172M parameters. In addition to downstream accuracy, we analyze the learned tokenizer and find that it adapts chunk granularity to local genomic context: the model tends to form shorter, higher-resolution chunks around functional elements (e.g., regions with high PhyloP scores, gene bodies, promoters, introns, and exons), while assigning longer chunks to low-information repetitive regions.

Overall, our contributions can be summarized as follows:

1. **Bidirectional masked DNA language model.** We adapt the dynamic tokenization mechanism in the autoregressive field to a bidirectional modeling scheme, targeted for DNA sequences (Section 3.1).

2. **State-of-the-art performance on various DNA benchmarks.** Our proposed DNACHUNKER outperforms the baselines in the Nucleotide Transformer and Genomic benchmarks (Section 4.1).

3. **Robustness and adaptivity of tokenization.** Our tokenization scheme is robust against mutations and adaptively allocates fine-grained representations for functional elements, while compressing non-functional elements with coarse-grained representations (Section 4.2).

2 RELATED WORKS

2.1 DNA LANGUAGE MODELS

Autoregressive generation models. While masked DNA LLMs excel at understanding and predicting DNA sequences, generative capabilities in this domain are still in their early stages. An early preprint on DNAGPT (Yang et al., 2024) demonstrated the ability to learn mammalian genomic structures through next-token prediction and other pre-training tasks. Recent works like HyenaDNA (Nguyen et al., 2023) and megaDNA (Shao & Yan, 2024) have achieved longer context lengths by employing the Hyena (Poli et al., 2023) and multiscale transformer architectures, respectively, though they are limited in their data and model scale. Next, Evo (Nguyen et al., 2024) was trained on an extensive dataset of prokaryotic and viral genomes. Evo2 (Brixi et al., 2025) extends this idea with 7B and 40B parameter models trained on 9.3 trillion DNA base pairs, achieving an unprecedented 1 million token context window with single-nucleotide resolution. Notably, Evo and Evo2 demonstrated practical utility by designing CRISPR-Cas molecular complexes (Nguyen et al., 2024) and bacteriophages (King et al., 2025) in the real world.

Non-autoregressive generation models. In addition, masked language models (MLMs) have been investigated for the representation learning of DNA sequences. MLMs are attractive since they allow reflection of the bidirectional nature of DNA sequences, e.g., regulatory motifs can act in both directions and functional prediction requires context from both upstream and downstream regions. The Nucleotide Transformer (NT; Dalla-Torre et al., 2025) scaled model parameters from 100 million to 2.5 billion and was trained on a diverse set of multispecies genomes. Subsequent studies, such as DNABERT-2 (Zhou et al., 2024) and GROVER (Sanabria et al., 2024) proposed to use Byte Pair Encoding (BPE) over k-mer tokenizers for masked DNA LLMs. A primary limitation of these models has been their insufficient context length, a consequence of the high computational cost of extending context in the standard transformer architecture. To address this, GENA-LM (Fishman et al., 2025) employs sparse attention, while Caduceus (Schiff et al., 2024) utilizes the lightweight BiMamba architecture (Tang et al., 2024).

2.2 LEARNABLE TOKENIZERS

Autoregressive generation models. Tokenization methods have primarily been developed in the context of autoregressive generation models. Existing models often rely on an outside function or module to identify boundaries. This includes delimiter-based methods like SpaceByte (Slagle, 2024), which works well for languages with clear separators, and entropy-based methods like the Byte Latent Transformer (Pagnoni et al., 2024), which identify boundaries based on conditional entropy. Recently, Hwang et al. (2025) proposed H-Net as a module to learn dynamic chunking, which learns optimal segmentation strategies directly from data through training and matches the performance of models based on fixed tokenizers for natural language and DNA tasks.

Non-autoregressive generation models. For non-autoregressive models, similar principles are applied with different design considerations. Charformer (Tay et al., 2022) introduced a gradient-based method for pooling sequences at multiple resolutions. Other approaches, such as eByte (Thawani et al., 2023) and Word-based self-attention fusion (Sreedhar et al., 2023), perform external chunking based on words. Our work, DNACHUNKER, fills a critical gap by being the first model to apply a learnable, dynamic chunking mechanism to a non-autoregressive, masked DNA language model.

3 METHODOLOGY

3.1 ARCHITECTURE DETAILS OF DNACHUNKER

DNACHUNKER is a bidirectional masked language model (MLM) for genomic sequences, structured around a central design principle: the main network serves as the primary locus of long-range contextual reasoning. Operating on adaptively compressed segment sequences, the main network aggregates evidence across distant genomic regions while remaining computationally tractable. The encoder and decoder are designed to support this goal—compressing sequences into the main network and reconstructing base-pair predictions from it—while addressing two key challenges.

First, genomic signals depend on both upstream and downstream context, demanding bidirectionality throughout. Most prior adaptive segmentation methods were introduced in autoregressive settings (Pagnoni et al., 2024; Hwang et al., 2025), making them inherently directional. In contrast, DNACHUNKER employs bidirectional Mamba layers to inform segmentation decisions from both flanks in the encoder, while the decoder applies bidirectional probability-gated smoothing during reconstruction.

Second, `[MASK]` tokens are functional artifacts of pretraining, not semantic nucleotides, and require careful handling to prevent information leakage. In the encoder, we enforce boundaries around masked positions so that segmentation decisions remain mask-invariant and transfer to downstream sequences without masks. In the decoder, we gate residual connections to masked positions, ensuring their reconstruction depends solely on main-network representations rather than leaked encoder context.

Together, these designs concentrate contextual modeling within the main network while enabling adaptive compression that respects the bidirectional and mask-sensitive nature of MLM pretraining. We illustrate the architecture in Figure 1a and provide hyperparameters in Section A. Below, we describe the encoder (Section 3.1.1), main network (Section 3.1.2), and decoder (Section 3.1.3) in detail. We summarize our pretraining and finetuning scheme in Section 3.2 and Section B.

3.1.1 ENCODER: MASK-PROTECTED ADAPTIVE SEGMENTATION

The encoder compresses genomic sequences via *adaptive segmentation*, reducing the effective sequence length in low-information regions while preserving high-information content at an appropriate granularity. DNACHUNKER applies a two-stage hierarchical procedure that progressively maps base pair–level signals into coarser, semantically meaningful representations. Each stage follows three steps: (1) encode the current sequence into contextualized embeddings, (2) infer decision boundaries between adjacent positions, and (3) downsample embeddings according to these boundaries to produce a shorter stage output. This structured workflow retains salient genomic patterns while reducing computational cost.

In practice, step (1) is handled by a lightweight *bidirectional* Mamba backbone (Schiff et al., 2024) , which transforms raw tokens into base-pair–resolution features optimized for boundary inference. Steps (2)–(3) are performed by the segmentation module, which predicts boundaries from these processed features and aggregates embeddings to form segment representations. Bidirectionality is important for genomic sequences because boundary evidence can arise from both upstream and downstream context, unlike autoregressive settings that typically constrain encoders to a single direction (Pagnoni et al., 2024; Hwang et al., 2025).

Formally, given an input sequence of length T, let $x^{(0)} = (x_1^{(0)}, \dots, x_T^{(0)})$ denote base pair–level embeddings. These embeddings are processed by the first-stage encoder, producing intermediate representations $\widehat{x}^{(0)}$. A routing network then computes boundary probabilities $p_t^{(0)}$ for each position $t \in [1, T]$ using cosine dissimilarity between projected query and key vectors:

$$
\begin{aligned}
p_t^{(0)} &= \frac{1}{2}\left(1 - \frac{(q_t^{(0)})^\top k_{t-1}^{(0)}}{\|q_t^{(0)}\| \cdot \|k_{t-1}^{(0)}\|}\right), \\
q_t^{(0)} &= W_{\text{enc},q}^{(1)}\, \widehat{x}_t^{(0)}, \qquad k_t^{(0)} = W_{\text{enc},k}^{(1)}\, \widehat{x}_t^{(0)},
\end{aligned}
\tag{1}
$$

where $W^{(s)}_{\text{enc},q}$ and $W^{(s)}_{\text{enc},k}$ are learnable parameters of the encoder routing network at stage $s \in \{1, 2\}$. We obtain hard boundary indicators by thresholding:

$$b_t^{(s)} = \mathbf{1}\Big(p_t^{(s)} \geq 0.5\Big). \tag{2}$$

These indicators define chunk boundaries. The first stage collects $T' = \sum_{t=1}^{T} b_t^{(0)}$ adaptive chunks from $\widehat{x}^{(0)}$, yielding chunked embeddings $x^{(1)} \in \mathbb{R}^{T' \times d}$. The second-stage encoder applies the same procedure to $x^{(1)}$ to produce a more coarse-grained representation $x^{(2)} = (x_1^{(2)}, \ldots, x_{T''}^{(2)})$ with $T'' < T'$, which is used as input to the main network.

Mask protection mechanism. A key complication in MLM pretraining is that `[MASK]` is a *functional* token rather than a semantic nucleotide. Since the routing network predicts boundaries from contextualized features, allowing `[MASK]` to merge with neighboring bases can introduce shortcuts: the model may learn segmentation patterns that exploit mask placement (a pretraining-only artifact) rather than genomic context, which does not transfer to downstream sequences without masked tokens. To prevent mask-conditioned tokenization, we enforce boundaries around every masked base pair so that masked positions are never merged into larger chunks.

Concretely, for each masked index m, we force chunk boundaries immediately before and after it, ensuring that the masked token forms a singleton chunk and its neighbors start new chunks. This mask protection is applied throughout the hierarchical encoder, ensuring that masked tokens remain isolated at every segmentation stage. As a result, adaptive segmentation decisions are driven by genomic context rather than by the presence of `[MASK]`, while the MLM learning signal is preserved through compression.

3.1.2 MAIN NETWORK

The main network consists of 8 Transformer blocks operating on the compressed segment sequence. Each block follows the standard Transformer design with layer normalization, multi-headed self-attention, and a feedforward network with GELU (Hendrycks & Gimpel, 2023) activation. We incorporate Rotary Position Embeddings (RoPE; Su et al., 2024) in the attention mechanism to encode positional information, utilizing the mean index location of each chunk. The main network accounts for the majority of parameters in DNACHUNKER and memory usage during inference. Note that, the main network is intended to be the primary locus of contextual modeling the input DNA sequence.

3.1.3 DECODER: HIERARCHICAL DECHUNKING WITH BIDIRECTIONAL SMOOTHING

Mirroring the encoder's two-stage adaptive segmentation, the decoder reconstructs representations in two hierarchical steps ($z^{(0)} \to z^{(1)} \to z^{(2)}$), progressively expanding the compressed sequence back to base-pair resolution. Unlike autoregressive reconstructions (Hwang et al., 2025; Pagnoni et al., 2024) that must remain causal, our bidirectional MLM decoder leverages context from both directions.

Each dechunking step proceeds as follows. Given compressed representations $z^{(s)} \in \mathbb{R}^{T_s \times d}$ and boundary indicators $b^{(S-s)}$ from the corresponding encoder stage, we first *paste* each segment representation to all positions it governs via cumulative boundary counts:

$$\tilde{z}_t^{(s+1)} = z^{(s)}_{\sum_{k=1}^{t} b_k^{(S-s)}}. \tag{3}$$

Here, S denotes the total number of hierarchical layers. This initial assignment produces piecewise-constant representations. To (1) enable gradient flow through discrete boundary decisions and (2) incorporate bidirectional context, we then apply probability-gated smoothing in *both directions*:

$$z_t^{(s+1)} = \tfrac{1}{2}\big(\textsc{Scan}_{\rightarrow}(\tilde{z}^{(s+1)}, p)_t + \textsc{Scan}_{\leftarrow}(\tilde{z}^{(s+1)}, p)_t\big), \tag{4}$$

where $\textsc{Scan}_{\rightarrow}$ and $\textsc{Scan}_{\leftarrow}$ denote forward and backward linear recurrences gated by boundary probabilities p. The smoothed representations are then combined with gated encoder residuals from the corresponding stage and refined by bidirectional Mamba layers, before proceeding to the next dechunking step or the final language model head.

Masked residual gating. At each dechunking stage, the decoder employs residual connections from the corresponding encoder features to aid reconstruction of fine-grained information. However, allowing these residuals to flow into masked positions creates an undesirable shortcut: since the encoder's bidirectional layers mix information across neighbors, masked tokens could be reconstructed from leaked encoder context alone, bypassing the main network. To enforce contextual compute in the main network, we gate residual connections based on whether a position's assigned segment contains a mask token—positions in masked segments receive zero residual. This design ensures that the main network serves as the primary locus of long-range dependency modeling.

3.2 Model Pretraining

Loss function. DNAChunker is pretrained with masked language modeling, with down-weighting of repetitive regions of DNA by 0.1, in line with prior works (Brixi et al., 2025). The loss is formulated as follows:

$$\begin{aligned} \mathcal{L}_{\text{MLM}} &= \sum_{t \in M} w_t \, \mathcal{L}_{\text{CE}}(t), \\ w_t &= \begin{cases} 0.1 & \text{if position } t \text{ is in a repetitive region,} \\ 1.0 & \text{otherwise.} \end{cases} \end{aligned} \tag{5}$$

where $\mathcal{L}_{\text{CE}}(t)$ denotes the cross entropy loss for predicting the masked nucleotide at position t. Additionally, to control the degree of compression from the chunking layers, we use the ratio loss proposed by Hwang et al. (2025):

$$\begin{aligned} \mathcal{L}_{\text{ratio}}^{(s)} &= \frac{\overline{b}^{(s)} \, \overline{p}^{(s)}}{\alpha^{(s)}} + \frac{\big(1 - \overline{b}^{(s)}\big)\big(1 - \overline{p}^{(s)}\big)}{1 - \alpha^{(s)}}, \\ \overline{b}^{(s)} &= \frac{1}{T} \sum_{t=1}^{T} b_t^{(s)}, \qquad \overline{p}^{(s)} = \frac{1}{T} \sum_{t=1}^{T} p_t^{(s)}. \end{aligned} \tag{6}$$

where $\overline{b}^{(s)}$ and $\overline{p}^{(s)}$ are the fraction of selected tokens and the average boundary probability, respectively, and $\alpha^{(s)} \in (0, 1)$ is the target compression ratio of the encoder, which is a controllable parameter. Note that $\overline{b}^{(s)}$ is non-differentiable, but the network can be trained towards the target compression ratio through tuning $\overline{p}^{(s)}$. Together, we train the model to minimize the loss $\mathcal{L} = \mathcal{L}_{\text{MLM}} + \lambda \mathcal{L}_{\text{ratio}}^{(0)} + \lambda \mathcal{L}_{\text{ratio}}^{(1)}$, where λ is the weighting coefficient. More details about pretraining can be found in Section B.

Dataset. We pretrain our model on the Human Reference Genome, adopting the data partitioning strategy from Enformer (Avsec et al., 2021). The genome is first divided into non-overlapping regions of 2^{20} (1,048,576) base pairs (bp), which will be allocated to the training, validation, and test sets. These regions are subsequently segmented into input sequences with a maximum length of 8192 bp. During the preprocessing, ambiguous nucleotides ('N') are mapped to a padding token and are excluded from the loss computation. Following the methodology of BERT (Devlin et al., 2019), for each input sequence, 15% of all nucleotides are randomly selected for prediction. Of this selection, 80% are replaced with a `[MASK]` token, 10% are substituted with a random nucleotide, and the remaining 10% are left unchanged.

Fine-tuning on downstream tasks. For fine-tuning on the downstream tasks, we remove the language model head and perform average pooling over the valid tokens, *i.e.* excluding `[PAD]` tokens. The pooled output is subsequently passed through a linear layer.

4 Experiments

In what follows, we demonstrate the experimental results for evaluating DNAChunker upon two benchmark datasets: Nucleotide Transformer benchmark (Dalla-Torre et al., 2025) and Genomic benchmark (Grešová et al., 2023). We show that, despite the small number of parameters (172M),

Table 1: **Nucleotide Transformer Benchmark.** The reported values represent the Matthews Correlation Coefficient (MCC; mean ± standard error) averaged over 10-fold cross-validation. Best results are **bold**; second best are underlined.

	Enformer (252M)	DNABERT-2 (117M)	HyenaDNA (55M)	NT-multi (2.5B)	NT-v2 (500M)	Caduceus-Ph (8M)	Caduceus-PS (8M)	GROVER (87M)	GENErator (1.2B)	DNACHUNKER (172M)
Histone Markers										
H3	0.724 ± 0.018	0.785 ± 0.012	0.781 ± 0.015	0.793 ± 0.013	0.788 ± 0.010	0.794 ± 0.012	0.772 ± 0.022	0.768 ± 0.008	0.806 ± 0.005	**0.817** ± 0.011
H3K14ac	0.284 ± 0.024	0.515 ± 0.009	0.608 ± 0.020	0.538 ± 0.009	0.538 ± 0.015	0.564 ± 0.033	0.596 ± 0.038	0.548 ± 0.020	0.605 ± 0.008	**0.711** ± 0.021
H3K36me3	0.345 ± 0.019	0.591 ± 0.005	0.614 ± 0.014	0.618 ± 0.011	0.618 ± 0.015	0.590 ± 0.018	0.611 ± 0.048	0.563 ± 0.017	0.657 ± 0.007	**0.677** ± 0.003
H3K4me1	0.291 ± 0.016	0.512 ± 0.008	0.512 ± 0.008	0.541 ± 0.005	0.544 ± 0.009	0.468 ± 0.015	0.487 ± 0.029	0.461 ± 0.018	0.553 ± 0.009	**0.631** ± 0.009
H3K4me2	0.207 ± 0.021	0.333 ± 0.013	0.455 ± 0.028	0.324 ± 0.014	0.302 ± 0.020	0.332 ± 0.034	0.431 ± 0.016	0.403 ± 0.042	0.424 ± 0.013	**0.599** ± 0.011
H3K4me3	0.156 ± 0.022	0.353 ± 0.021	0.550 ± 0.015	0.408 ± 0.011	0.437 ± 0.028	0.490 ± 0.042	0.528 ± 0.033	0.458 ± 0.022	0.512 ± 0.009	**0.660** ± 0.045
H3K79me3	0.498 ± 0.013	0.615 ± 0.010	0.669 ± 0.014	0.623 ± 0.010	0.621 ± 0.012	0.641 ± 0.028	0.682 ± 0.018	0.626 ± 0.026	0.670 ± 0.011	**0.731** ± 0.012
H3K9ac	0.415 ± 0.020	0.545 ± 0.009	0.586 ± 0.021	0.547 ± 0.011	0.567 ± 0.020	0.575 ± 0.024	0.564 ± 0.018	0.581 ± 0.015	0.612 ± 0.006	**0.678** ± 0.007
H4	0.735 ± 0.023	0.797 ± 0.008	0.763 ± 0.012	0.808 ± 0.007	0.795 ± 0.008	0.788 ± 0.010	0.799 ± 0.010	0.769 ± 0.017	**0.815** ± 0.008	0.813 ± 0.012
H4ac	0.275 ± 0.022	0.465 ± 0.013	0.564 ± 0.011	0.492 ± 0.014	0.502 ± 0.025	0.548 ± 0.027	0.585 ± 0.018	0.530 ± 0.017	0.592 ± 0.015	**0.687** ± 0.027
Average MCC (↑)	0.393	0.551	0.610	0.569	0.571	0.579	0.606	0.571	0.625	**0.701**
Regulatory Annotation										
Enhancer	0.454 ± 0.029	0.525 ± 0.026	0.520 ± 0.031	0.545 ± 0.028	0.561 ± 0.029	0.522 ± 0.024	0.511 ± 0.026	0.516 ± 0.018	**0.580** ± 0.015	0.558 ± 0.011
Enhancer Type	0.312 ± 0.043	0.423 ± 0.018	0.403 ± 0.056	0.444 ± 0.022	0.444 ± 0.036	0.403 ± 0.028	0.410 ± 0.026	0.433 ± 0.029	0.477 ± 0.017	**0.519** ± 0.005
Promoter All	0.910 ± 0.004	0.945 ± 0.003	0.919 ± 0.003	0.951 ± 0.004	0.952 ± 0.002	0.937 ± 0.002	0.941 ± 0.003	0.926 ± 0.004	0.962 ± 0.002	**0.967** ± 0.013
Promoter NonTATA	0.910 ± 0.006	0.944 ± 0.003	0.919 ± 0.004	0.969 ± 0.003	0.952 ± 0.003	0.935 ± 0.007	0.940 ± 0.002	0.925 ± 0.006	0.962 ± 0.001	**0.971** ± 0.007
Promoter TATA	0.920 ± 0.012	0.911 ± 0.011	0.881 ± 0.020	0.919 ± 0.008	0.933 ± 0.009	0.895 ± 0.010	0.903 ± 0.010	0.891 ± 0.009	0.948 ± 0.008	**0.961** ± 0.015
Average MCC (↑)	0.701	0.750	0.728	0.766	0.768	0.738	0.741	0.738	0.786	**0.796**
Splice Site Annotation										
Splice Acceptor	0.772 ± 0.007	0.909 ± 0.004	0.935 ± 0.005	0.973 ± 0.002	0.973 ± 0.004	0.918 ± 0.017	0.907 ± 0.015	0.912 ± 0.010	**0.981** ± 0.002	0.969 ± 0.013
Splice Site All	0.831 ± 0.012	0.950 ± 0.003	0.917 ± 0.006	0.974 ± 0.004	0.975 ± 0.002	0.935 ± 0.011	0.953 ± 0.005	0.919 ± 0.005	**0.976** ± 0.011	0.968 ± 0.030
Splice Donor	0.813 ± 0.015	0.927 ± 0.003	0.894 ± 0.013	0.974 ± 0.002	0.977 ± 0.007	0.912 ± 0.009	0.930 ± 0.010	0.888 ± 0.012	**0.978** ± 0.001	0.960 ± 0.007
Average MCC (↑)	0.805	0.929	0.915	0.974	0.975	0.922	0.930	0.906	**0.979**	0.965
Total Average MCC (↑)	0.547	0.669	0.694	0.690	0.693	0.680	0.697	0.673	0.728	**0.772**
Total Average Rank (↓)	9.67	6.72	6.00	4.83	4.56	6.33	5.61	7.22	2.06	**1.67**

Table 2: **Genomics Benchmark.** The reported values represent the Top-1 Accuracy (Acc; mean ± standard error) averaged over 10-fold cross-validation. Best results are **bold**; second best are underlined.

	DNABERT-2 (117M)	HyenaDNA (55M)	NT-v2 (500M)	Caduceus-Ph (8M)	Caduceus-PS (8M)	GROVER (87M)	GENErator (1.2B)	GENErator-All (1.2B)	DNACHUNKER (172M)
Coding vs. Intergenomic	0.951 ± 0.002	0.902 ± 0.004	0.955 ± 0.001	0.933 ± 0.001	0.944 ± 0.002	0.919 ± 0.002	**0.963** ± 0.000	0.959 ± 0.001	0.955 ± 0.012
Drosophila Enhancers Stark	0.774 ± 0.011	0.770 ± 0.016	0.797 ± 0.009	**0.827** ± 0.010	0.816 ± 0.015	0.761 ± 0.011	0.821 ± 0.005	0.768 ± 0.015	0.779 ± 0.021
Human Enhancers Cohn	0.758 ± 0.005	0.725 ± 0.009	0.756 ± 0.006	0.747 ± 0.003	0.749 ± 0.003	0.738 ± 0.003	**0.763** ± 0.002	0.754 ± 0.006	0.761 ± 0.011
Human Enhancers Ensembl	0.918 ± 0.003	0.901 ± 0.003	0.921 ± 0.004	**0.924** ± 0.002	0.923 ± 0.002	0.911 ± 0.004	0.917 ± 0.002	0.912 ± 0.002	0.922 ± 0.007
Human Ensembl Regulatory	0.874 ± 0.007	0.932 ± 0.001	**0.941** ± 0.001	0.938 ± 0.004	**0.941** ± 0.002	0.897 ± 0.001	0.928 ± 0.001	0.926 ± 0.001	0.935 ± 0.005
Human non-TATA Promoters	0.957 ± 0.008	0.894 ± 0.023	0.932 ± 0.006	0.961 ± 0.003	0.961 ± 0.002	0.950 ± 0.005	0.958 ± 0.001	0.955 ± 0.005	**0.962** ± 0.001
Human OCR Ensembl	0.806 ± 0.003	0.774 ± 0.004	0.813 ± 0.001	0.825 ± 0.004	**0.826** ± 0.003	0.789 ± 0.002	0.823 ± 0.002	0.812 ± 0.003	0.810 ± 0.007
Human vs. Worm	0.977 ± 0.001	0.958 ± 0.004	0.976 ± 0.001	0.975 ± 0.001	0.976 ± 0.001	0.966 ± 0.001	**0.980** ± 0.000	0.978 ± 0.001	0.969 ± 0.001
Mouse Enhancers Ensembl	0.865 ± 0.014	0.756 ± 0.030	0.855 ± 0.018	0.788 ± 0.028	0.826 ± 0.021	0.742 ± 0.025	0.871 ± 0.015	0.784 ± 0.027	**0.874** ± 0.020
Average Acc (↑)	0.876	0.846	0.883	0.880	0.885	0.853	**0.892**	0.872	0.885
Average Rank (↓)	5.11	8.22	4.17	3.89	3.33	8.11	**2.89**	5.44	3.29

DNACHUNKER demonstrates state-of-the-art performance (Section 4.1). Next, we describe ablative experiments comparing DNACHUNKER with prior DNA-specific tokenization methods, and provide extensive analysis of the learned tokenizer, demonstrating its robustness and inherent biological understanding (Section 4.2).

4.1 DOWNSTREAM TASKS

Nucleotide transformer benchmark. We evaluate our model on the Nucleotide Transformer benchmark (Dalla-Torre et al., 2025), which aggregates 18 datasets spanning three task families: (i) *histone mark prediction* from chromatin profiling assays, (ii) *regulatory annotation* such as promoter and enhancer classification, and (iii) *splice-site* annotation at donor/acceptor boundaries. Following the evaluation protocol of Wu et al. (2025), we perform 10-fold cross-validation and report the Matthews Correlation Coefficient (MCC) for each dataset and the average rank among 10 models. Specific finetuning details of DNACHUNKER are reported in C.1, while scores of previous baseline models are taken from Wu et al. (2025).

Results are summarized in Table 1. DNACHUNKER achieves state-of-the-art performance on 10 out of 18 datasets, bypassing the previous best model, GENERATOR (Wu et al., 2025), by a large margin in both average MCC and average rank. Our gains are especially more pronounced upon the histone mark prediction tasks, showing an average gain of 14.2%. Note that DNACHUNKER is trained only on the human reference genome, using only 13% of the Generator's number of parameters.

Table 3: **Nucleotide Transformer Benchmark Revised.** Each entry reports the average MCC (mean ± average standard error) across the subtasks within each group. The final two rows report the overall average MCC and average rank across *all* subtasks. Best results are **bold**; second best are underlined.

	HyenaDNA (6.6M)	Caduceus-PS (7.7M)	GENA-LM-Base (110M)	NT-MS-100M (100M)	MxDNA (100M)	PatchDNA (19.2M)	DNACHUNKER (172M)
Histone markers	0.509 ± 0.011	0.543 ± 0.011	0.537 ± 0.019	0.552 ± 0.007	0.555 ± 0.013	0.560 ± 0.009	**0.562** ± 0.006
Regulatory annotations	0.635 ± 0.015	0.650 ± 0.015	0.623 ± 0.021	0.660 ± 0.015	0.664 ± 0.017	0.689 ± 0.010	**0.690** ± 0.008
Splice site annotations	0.836 ± 0.017	0.777 ± 0.013	0.768 ± 0.007	**0.960** ± 0.002	0.868 ± 0.020	0.740 ± 0.028	0.936 ± 0.009
Total Avg. MCC (↑)	0.599	0.612	0.600	0.650	0.637	0.626	**0.660**
Total Avg. Rank (↓)	6.06	4.89	5.67	3.06	3.06	3.00	**2.05**

Genomic benchmark. The Genomic benchmark suite (Grešová et al., 2023) consists of nine different tasks centered on human genomic data. This includes *enhancer* and *promoter* recognition, *coding vs. intergenic* discrimination, and a small *species* control (human vs. worm). We once again follow the evaluation protocol of Wu et al. (2025) and report the top-1 accuracy averaged over 10-fold cross-validation. Specific finetuning details are reported in Section C.2, while scores from baseline models are taken from (Wu et al., 2025).

Our DNACHUNKER achieves the second-highest rank and performance on-par with GENERATOR, while utilizing 7× fewer parameters than the best model Generator (Wu et al., 2025) with 1.2B parameters. Additionally, our model was trained solely upon the human reference genome, while GENERATOR used various types of genomes beyond human.

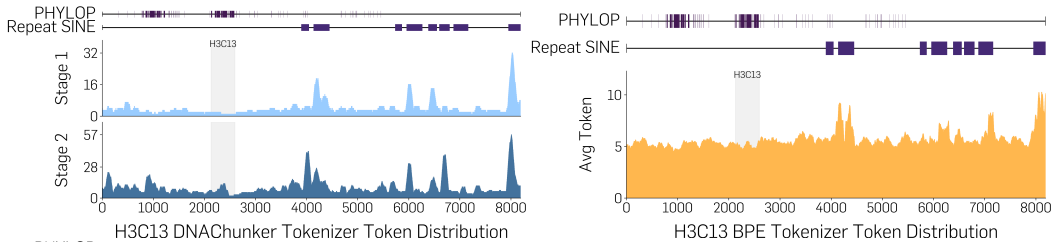
4.2 ABLATIVE STUDIES

Comparison with DNA-targeted tokenization. Several prior studies have proposed tokenization strategies tailored to genomic sequences. However, these approaches often rely on hand-crafted signals (e.g., using external conservation scores to define boundaries (Del Vecchio et al., 2025)) or impose architectural constraints that implicitly cap the effective token granularity (e.g., fixed-kernel CNN tokenizers (Qiao et al., 2024)). Such design choices can restrict flexibility and lead to degraded performance when the appropriate resolution varies across genomic contexts. We therefore compare DNACHUNKER against these DNA-targeted baselines - MXDNA (Qiao et al., 2024) and PATCHDNA (Del Vecchio et al., 2025) on the revised Nucleotide Transformer benchmark (Dalla-Torre et al., 2025) (summary in Table 3; full results in Section D). Across all task groups, DNACHUNKER consistently outperforms both methods, with the largest gains observed on splice-site annotation. Prior analysis in (Qiao et al., 2024), citing (Lindsey et al., 2024), argues that splice-site prediction particularly benefits from uniform, fine-grained tokenization, and reports that segmentation-based schemes can underperform fixed BPE or k-mer tokenizers on these tasks. In contrast, DNACHUNKER remains competitive with fixed tokenizers on splice-site benchmarks while retaining the benefits of adaptive segmentation on other task families, suggesting that it preserves base-level sensitivity when needed without sacrificing broader contextual modeling elsewhere.

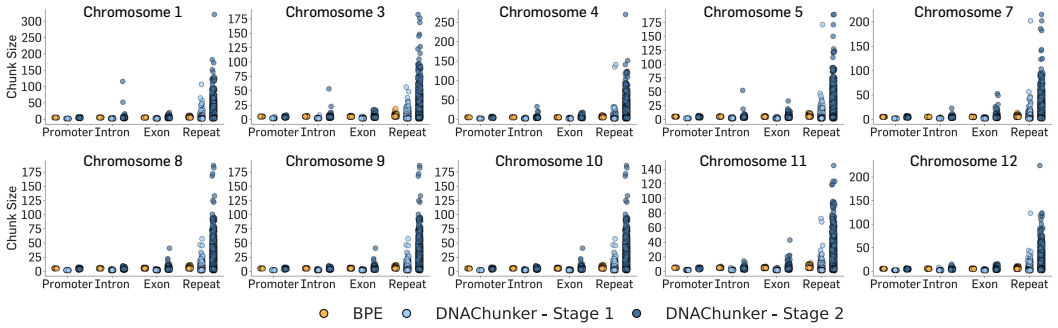
Biological meaning of chunks. To assess whether DNACHUNKER produces biologically informed adaptive segmentation, we compare its token-size patterns against the fixed BPE tokenizer (here, we used DNABERT-2 (Zhou et al., 2024) as representative). We focus on whether token granularity changes systematically with functional annotations, rather than remaining uniform across the sequence.

We examine three genes in the histone cluster of H3 family (H3C13, H3C14, and H3C15), which contain both (i) highly conserved coding segments with elevated phyloP scores (PHYLOP) and (ii) repetitive Short Interspersed Nuclear Elements (Repeat SINE). A desirable tokenizer should allocate higher resolution (shorter segments) to conserved and coding regions, while using longer segments to summarize low-complexity repetitive sequence.

Figure 2 shows that the BPE baseline exhibits a largely uniform distribution of token sizes across annotations, applying similar granularity to conserved elements and repetitive SINEs. In contrast, DNACHUNKER adapts segment length to local genomic context: it assigns longer segments in SINE regions, where sequence is more redundant, and produces shorter segments within gene bodies and



(a) H3C13 gene views with genomic tracks.



(b) Chromosome-wide distributions by genomic feature.

Figure 2: **Token size distributions of DNACHUNKER and BPE.** (a) The BPE tokenizer is compared against the two-stage DNACHUNKER tokenizer on the H3C13, H3C14 and H3C15 gene. Tracks include gene bodies (shaded), conserved elements (PHYLOP), and SINE repeats. (b) Human chromosome-wide segment-size distributions for BPE and both stages of DNACHUNKER, categorized by Promoter, Intron, Exon, and Repeat.

in high-conservation intervals. Importantly, this contrast is driven by the tokenization mechanism itself: BPE is fixed after preprocessing and cannot condition token boundaries on biological context, whereas DNACHUNKER learns boundaries jointly with representation learning, allowing segmentation decisions to vary with the local sequence and its inferred function. This behavior suggests that the learned tokenization is sensitive to biologically meaningful structure, even though training uses only a masked language modeling objective.

We further quantify this trend at a genome-wide scale in Figures 1c and 2b by stratifying segment lengths by annotation (Promoters, Introns, Exons, and Repeats) across chromosomes 1, 3, 4, 5, 7, 21, and 22. DNACHUNKER yields a broad distribution of segment lengths that varies across annotations, including substantially longer segments in repeat regions (up to 320 bp in our setting), whereas BPE produces predominantly fixed-length tokens across categories. Taken together, these results indicate that adaptive segmentation can align representational resolution with biological content, allocating finer granularity to information-dense regions and coarser granularity to repetitive sequence, which plausibly supports downstream predictive performance.

Robustness to mutations. We evaluate whether a tokenizer produces *stable segmentations* under small genetic perturbations by comparing the tokenizations of a reference sequence and its mutated counterpart for variants sampled from ClinVar (Landrum et al., 2016). We draw 1,000 examples for each category: benign SNVs, pathogenic SNVs, benign InDels, and pathogenic InDels. For each pair, we compute a similarity score

$$S(x^{\text{ref}}, x^{\text{mut}}) = (1-\gamma)\, S_{\text{boundary}} + \gamma\, S_{\text{content}},$$

where x^{ref} and x^{mut} are the two token sequences. S_{boundary} is the Jaccard similarity

Table 4: **Robustness of tokenizers against mutations.** Similarity scores between tokenizations of a reference sequence and its mutated counterpart on ClinVar variants. Higher is more stable under mutation.

		BPE	DNACHUNKER	
			Stage 1	Stage 2
SNV	Benign	0.9993	0.9987	0.9940
	Pathogenic	0.9994	0.9985	0.9934
InDel	Benign	0.7506	0.8512	0.7932
	Pathogenic	0.7434	0.8492	0.7900

between the *sets of token boundary positions* induced by the tokenizer, capturing how much the segmentation structure is preserved. S_{content} measures how similar the resulting token sequences are, implemented as a normalized edit-similarity (i.e., one minus the normalized edit distance) so that larger values indicate closer agreement. We set $\gamma = 0.5$ to weight structural and content stability equally.

Table 4 shows that both tokenizers are nearly invariant under SNVs. In contrast, for insertions and deletions, DNACHUNKER yields much higher similarity, especially at Stage 1, indicating that its context-dependent segmentation is less prone to cascading boundary shifts downstream of the mutation than fixed, frequency-driven merges.

5 CONCLUSION

In this work, we tackle a central bottleneck in genomic language modeling: unlike natural language, DNA lacks an intrinsic notion of discrete "words," making tokenization an ill-posed design choice. We therefore frame tokenization as a learnable component and introduce DNACHUNKER, which performs context-dependent adaptive segmentation jointly with masked sequence modeling. Across two widely used benchmark suites, DNACHUNKER improves over strong fixed-tokenization baselines while remaining parameter-efficient. Beyond accuracy, analyses of the learned segments indicate systematic structure rather than incidental compression: the model allocates finer granularity to functionally enriched regions (e.g., gene bodies and conserved elements) and coarser granularity to low-complexity or repetitive sequence. Together, these results suggest that end-to-end, data-driven segmentation can serve as a practical route toward genomic language models that better reflect biological organization.

ETHICS STATEMENT

The genomic data used for pre-training our model is the Human Reference Genome (HG38), which is a publicly available and fully anonymized resource widely used by the international scientific community. The use of this public reference genome ensures that our work does not involve private or identifiable genetic information, thereby posing no direct risk to individual privacy. While our research focuses on developing a foundational language model for understanding DNA sequences, we acknowledge that powerful genomic models could have future applications with broader ethical implications.

REPRODUCIBILITY STATEMENT

To ensure the reproducibility of our results, we have provided a detailed account of our methodology and experimental setup. Our model was pre-trained on the Human Reference Genome (HG38), using the public data partitioning strategy from the Enformer study. For downstream evaluation, we used two publicly available collections: the Nucleotide Transformer benchmark and the Genomic benchmark. Comprehensive details regarding the model architecture, pre-training configuration, and fine-tuning hyperparameters for every task are documented in the Appendices.

ACKNOWLEDGEMENT

This work was partly supported by Institute for Information & communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (RS-2019-II190075, Artificial Intelligence Graduate School Support Program(KAIST)), National Research Foundation of Korea(NRF) grant funded by the Ministry of Science and ICT(MSIT) (No. RS-2022-NR072184), GRDC(Global Research Development Center) Cooperative Hub Program through the National Research Foundation of Korea(NRF) grant funded by the Ministry of Science and ICT(MSIT) (No. RS-2024-00436165), and the Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (RS-2025-02304967, AI Star Fellowship(KAIST)).

REFERENCES

Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R Ledsam, Agnieszka Grabska-Barwinska, Kyle R Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature methods*, 18 (10):1196–1203, 2021.

Sam Behjati and Patrick S Tarpey. What is next generation sequencing? *Archives of Disease in Childhood-Education and Practice*, 98(6):236–238, 2013.

Garyk Brixi, Matthew G Durrant, Jerome Ku, Michael Poli, Greg Brockman, Daniel Chang, Gabriel A Gonzalez, Samuel H King, David B Li, Aditi T Merchant, et al. Genome modeling and design across all domains of life with evo 2. *BioRxiv*, pp. 2025–02, 2025.

Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P de Almeida, Hassan Sirelkhatim, et al. Nucleotide transformer: building and evaluating robust foundation models for human genomics. *Nature Methods*, 22(2):287–297, 2025.

Alice Del Vecchio, Chantriolnt-Andreas Kapourani, Abdullah M. Athar, Agnieszka Dobrowolska, Andrew Anighoro, Benjamin Tenmann, Lindsay Edwards, and Cristian Regep. Patchdna: A flexible and biologically-informed alternative to tokenization for dna. *bioRxiv*, 2025. doi: 10.1101/2025.11.28.691095. URL https://www.biorxiv.org/content/early/2025/11/29/2025.11.28.691095.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pp. 4171–4186, 2019.

Babatunde Ekundayo and Franziska Bleichert. Origins of dna replication. *PLoS genetics*, 15(9): e1008320, 2019.

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, 2025.

Katarína Grešová, Vlastimil Martinek, David Čechák, Petr Šimeček, and Panagiotis Alexiou. Genomic benchmarks: a collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24(1):25, 2023.

Dan Hendrycks and Kevin Gimpel. Gaussian error linear units (gelus), 2023. URL `https://arxiv.org/abs/1606.08415`.

Sukjun Hwang, Brandon Wang, and Albert Gu. Dynamic chunking for end-to-end hierarchical sequence modeling. *arXiv preprint arXiv:2507.07955*, 2025.

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, 2021.

Xuechao Jia, Xinyu He, Chuntian Huang, Jian Li, Zigang Dong, and Kangdong Liu. Protein translation: biological processes and therapeutic strategies for human diseases. *Signal Transduction and Targeted Therapy*, 9(1):44, 2024.

Samuel H King, Claudia L Driscoll, David B Li, Daniel Guo, Aditi T Merchant, Garyk Brixi, Max E Wilkinson, and Brian L Hie. Generative design of novel bacteriophages with genome language models. *bioRxiv*, pp. 2025–09, 2025.

Diederik P Kingma. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 2015.

Melissa J Landrum, Jennifer M Lee, Mark Benson, Garth Brown, Chen Chao, Shanmuga Chitipiralla, Baoshan Gu, Jennifer Hart, Douglas Hoffman, Jeffrey Hoover, et al. Clinvar: public archive of interpretations of clinically relevant variants. *Nucleic acids research*, 44(D1):D862–D868, 2016.

LeAnn M. Lindsey, Nicole L. Pershing, Anisa Habib, W. Zac Stephens, Anne J. Blaschke, and Hari Sundar. A comparison of tokenization impact in attention based and state space genomic language models. *bioRxiv*, 2024. doi: 10.1101/2024.09.09.612081. URL `https://www.biorxiv.org/content/early/2024/09/14/2024.09.09.612081`.

Jill E Moore, Michael J Purcaro, Henry E Pratt, Charles B Epstein, Noam Shoresh, Jessika Adrian, Trupti Kawli, Carrie A Davis, Alexander Dobin, et al. Expanded encyclopaedias of dna elements in the human and mouse genomes. *Nature*, 583(7818):699–710, 2020.

Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, et al. HyenaDNA: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in neural information processing systems*, 36:43177–43201, 2023.

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, et al. Sequence modeling and design from molecular to genome scale with evo. *Science*, 386(6723):eado9336, 2024.

Artidoro Pagnoni, Ram Pasunuru, Pedro Rodriguez, John Nguyen, Benjamin Muller, Margaret Li, Chunting Zhou, Lili Yu, Jason Weston, Luke Zettlemoyer, et al. Byte latent transformer: Patches scale better than tokens. *arXiv preprint arXiv:2412.09871*, 2024.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pp. 28043–28078. PMLR, 2023.

Lifeng Qiao, Peng Ye, Yuchen Ren, Weiqiang Bai, Chaoqi Liang, Xinzhu Ma, Nanqing Dong, and Wanli Ouyang. Model decides how to tokenize: Adaptive dna sequence tokenization with mxdna, 2024. URL https://arxiv.org/abs/2412.13716.

Melissa Sanabria, Jonas Hirsch, Pierre M Joubert, and Anna R Poetsch. DNA language model GROVER learns sequence context in the human genome. *Nature Machine Intelligence*, 6(8): 911–923, 2024.

Yair Schiff, Chia Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range DNA sequence modeling. In *Forty-first International Conference on Machine Learning*, 2024. URL https://openreview.net/forum?id=mk3A5IUdn8.

Bin Shao and Jiawei Yan. A long-context language model for deciphering and generating bacteriophage genomes. *Nature Communications*, 15(1):9392, 2024.

Kevin Slagle. Spacebyte: Towards deleting tokenization from large language modeling. *Advances in Neural Information Processing Systems*, 37:124925–124950, 2024.

Makesh Narsimhan Sreedhar, Xiangpeng Wan, Yu Cheng, and Junjie Hu. Local byte fusion for neural machine translation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7199–7214, 2023.

Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

Shengkun Tang, Liqun Ma, Haonan Li, Mingjie Sun, and Zhiqiang Shen. Bi-mamba: Towards accurate 1-bit state space models. *arXiv preprint arXiv:2411.11843*, 2024.

Yi Tay, Vinh Q. Tran, Sebastian Ruder, Jai Gupta, Hyung Won Chung, Dara Bahri, Zhen Qin, Simon Baumgartner, Cong Yu, and Donald Metzler. Charformer: Fast character transformers via gradient-based subword tokenization. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=JtBRnrlOEFN.

Avijit Thawani, Saurabh Ghanekar, Xiaoyuan Zhu, and Jay Pujara. Learn your tokens: Word-pooled tokenization for language modeling. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 9883–9893, 2023.

Wei Wu, Qiuyi Li, Mingyang Li, Kun Fu, Fuli Feng, Jieping Ye, Hui Xiong, and Zheng Wang. GENERator: A long-context generative genomic foundation model, 2025. URL https://arxiv.org/abs/2502.07272.

Xianjun Yang, Wei Cheng, Yue Wu, Linda Ruth Petzold, William Yang Wang, and Haifeng Chen. DNA-GPT: Divergent n-gram analysis for training-free detection of GPT-generated text. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=Xlayxj2fWp.

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 ARCHITECTURE DETAILS

Table 5 summarizes the architectural configuration of DNACHUNKER, comprising 171.59M parameters in total. The model follows a hierarchical encoder–decoder design with explicit routing modules and lightweight bidirectional smoothing components. The encoder is organized into two stages, each built from a 2-layer BiMamba backbone (5.61M params per stage) paired with a DifferentiableRoutingModule (819.20K params per stage), operating in a 640-dimensional representation space to produce routed query/key projections before passing the compressed representation downstream.

The main compute backbone is a 30-layer TransformerBlock (147.50M params) using RoPE attention with 20 heads and 32 dimensions per head. Note that, the main network takes up most of the parameters. On the decoder side, DNACHUNKER applies two BiMamba stages (Stage 1/2; 5.61M each), followed by two extremely lightweight BidirectionalGatedSmoothing dechunkers and a final RMSNorm. Together, these components concentrate capacity in the Transformer trunk while keeping the hierarchical routing and smoothing pathways parameter-efficient.

Table 5: Hyperparameters of DNACHUNKER architecture (171.59M parameters in total).

Component	**Architecture / Details**	**#Params**
Token embedding	16 vocab size, 640 dim	10.24K
Encoder (Stage 1)	2-layer BiMamba (bidirectional=True)	5.61M
Router (Stage 1)	DifferentiableRoutingModule ($2\times$ Linear 640×640)	819.20K
Encoder (Stage 2)	2-layer BiMamba (bidirectional=True)	5.61M
Router (Stage 2)	DifferentiableRoutingModule ($2\times$ Linear 640×640)	819.20K
Main network	30-layer TransformerBlock • Attention: RoPE, 20 heads, 32 dim per head • MLP: `mlp_mult` $= 4$ (hidden dim $= 2560$)	147.50M
Decoder (Stage 1)	2-layer BiMamba (bidirectional=True)	5.61M
Decoder (Stage 2)	2-layer BiMamba (bidirectional=True)	5.61M
Dechunker 1	BidirectionalGatedSmoothing	640
Dechunker 2	BidirectionalGatedSmoothing	640
Final normalization	RMSNorm	640
Total	–	171.59M

B PRE-TRAINING DETAILS

Table 6 summarizes the pretraining setup of DNACHUNKER, including dataset specifications, optimization strategy, and masking details. The model is trained on the Enformer study splits, covering 34,021 genomic segments with a maximum sequence length of 2^{13} (8,192 bp), amounting to approximately 35 billion base pairs in total. To ensure scalability across different context lengths, we adopt a constant token budget of 2^{20} tokens per batch. This results in dynamically adjusted batch sizes depending on the sequence length: for instance, sequences of length 1,024 are processed in batches of 1,024, whereas long sequences of length 2^{17} (131k) are trained with a reduced batch size of 8.

Optimization is performed with the Adam optimizer (Kingma, 2015), using $\beta_1 = 0.95$ and $\beta_2 = 0.9$, and a learning rate of 5×10^{-4} following a cosine decay schedule. Pretraining follows a masked language modeling objective with 15% of input tokens selected for corruption: 80% of these are replaced with a `[MASK]` token, 10% with a random base, and 10% left unchanged. This bi-directional masking scheme encourages the model to leverage both local and global dependencies within DNA sequences while learning robust, context-aware representations.

Table 6: Pre-training Hyperparameters and Dataset Details

Component	**Hyperparameter**	**Value**
Dataset	Source	Enformer study splits
	Training Segments	34,021
	Max Sequence Length	8192 (2^{13})
	Total Tokens	≈ 35 billion base pairs
Training Configuration	Tokens per Batch	2^{20} (constant)
	Example Batch Sizes	Seq length 1,024 $\rightarrow$ Batch size 1,024; Seq length 131k (2^{17}) $\rightarrow$ Batch size 8
Optimizer & Learning Rate	Optimizer	ADAM (Kingma, 2015)
	ADAM β_1	0.95
	ADAM β_2	0.9
	LR	5×10^{-4}
	LR Schedule	Cosine Decay
Masking (bi-directional)	Masking Percentage	15% of input tokens
	Masking Strategy	80% replaced with `[MASK]` token, 10% replaced with a random token, and 10% unchanged

C FINETUNING DETAILS ON DOWNSTREAM TASKS

C.1 NUCLEOTIDE TRANSFORMER BENCHMARK

We fine-tune DNACHUNKER with a search space over learning rates $\{1\times10^{-5},\ 5\times10^{-5},\ 1\times10^{-4}\}$ and effective batch sizes $\{32,\ 64\,128\}$. We use attention pooling over token embeddings and *do not* apply RC augmentation or conjoining. Training runs for up to 15 epochs with shuffling each epoch; the best-validation checkpoint is used for scoring. All experiments use a single NVIDIA A100 GPU with 40GB VRAM.

Table 7: Hyperparameter settings for DNACHUNKER on Nucleotide Transformer benchmark.

Task	LR	BS
H3	5×10^{-5}	32
H3K14ac	5×10^{-5}	32
H3K36me3	5×10^{-5}	32
H3K4me1	5×10^{-5}	32
H3K4me2	5×10^{-5}	32
H3K4me3	5×10^{-5}	32
H3K79me3	5×10^{-5}	32
H3K9ac	5×10^{-5}	32
H4	1×10^{-4}	32
H4ac	5×10^{-5}	32
Enhancers	5×10^{-5}	32
Enhancers types	5×10^{-5}	32
Promoter all	5×10^{-5}	32
Promoter non-TATA	1×10^{-4}	32
Promoter TATA	5×10^{-5}	64
Splice sites acceptors	1×10^{-4}	32
Splice sites all	1×10^{-4}	32
Splice sites donors	1×10^{-4}	32

C.2 GENOMICS BENCHMARK

We fine-tune DNACHUNKER and GENERATOR with a search space over learning rates $\{1 \times 10^{-5},\ 5 \times 10^{-5},\ 1 \times 10^{-4}\}$ and effective batch sizes $\{32,\ 64\,128\}$. We use average pooling over token embeddings and *do not* apply RC augmentation or conjoining. Training runs for up to 20 epochs with shuffling each epoch; the best-validation checkpoint is used for scoring. All experiments use a single NVIDIA A100 GPU with 40GB VRAM.

Table 8: Hyperparameter settings for DNACHUNKER on Genomic benchmark.

Task	LR	BS
Coding vs. Intergenomic	1×10^{-5}	128
Drosophilia Enhancers Stark	1×10^{-4}	128
Human Enhancers Cohn	1×10^{-5}	128
Human Enhancer Ensembl	1×10^{-4}	32
Human Ensembl Regulatory	5×10^{-4}	64
Human non-TATA Promoters	1×10^{-4}	128
Human OCR Ensembl	5×10^{-4}	128
Human vs. Worm	1×10^{-5}	128
Mouse Enhancers Ensembl	1×10^{-5}	128

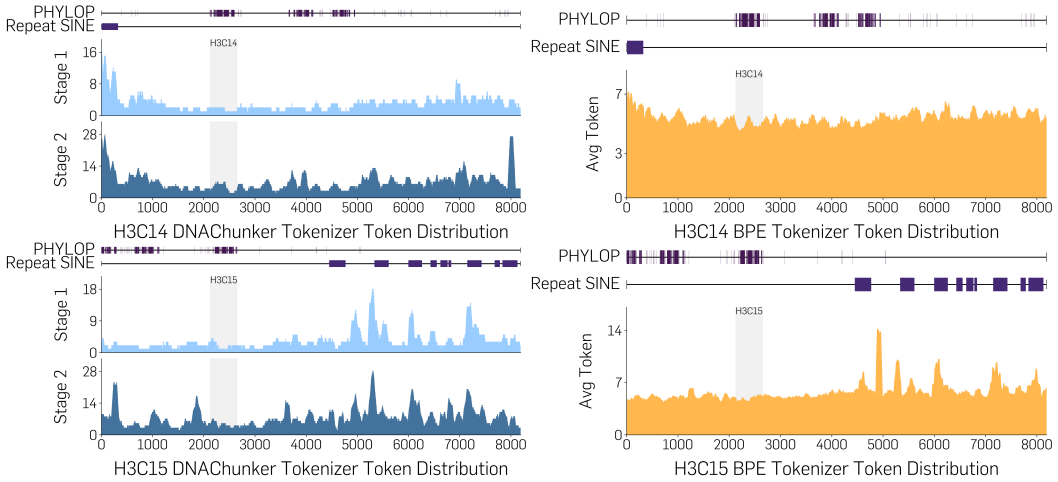


Figure 3: **Token size distributions of BPE and DNACHUNKER.** The BPE tokenizer (right) is compared against the two-stage DNACHUNKER tokenizer (left) on the H3C14, H3C15 gene. The plots visualize the average token size of BPE and the Stage 1 and Stage 2 token sizes of DNACHUNKER. Key genomic features are included as a reference, like gene bodies (shaded regions), conserved elements (PHYLOP), and SINE repeats.

Table 9: **Revised Nucleotide Transformer Benchmark.** The reported values represent the Matthews Correlation Coefficient (MCC; mean ± standard error) averaged over 3 random seeds. Best results are **bold**; second best are underlined.

	HyenaDNA (6.6M)	Caduceus-PS (7.7M)	GENA-LM-Base (110M)	NT-MS-100M (100M)	MxDNA (100M)	PatchDNA (19.2M)	DNACHUNKER (172M)
Histone Markers							
H2AFZ	0.481 ± 0.005	0.507 ± 0.007	0.466 ± 0.035	0.501 ± 0.009	0.512 ± 0.003	**0.523** ± 0.010	0.515 ± 0.002
H3K27ac	0.440 ± 0.003	0.475 ± 0.021	0.495 ± 0.010	0.496 ± 0.009	0.489 ± 0.031	0.486 ± 0.015	**0.504** ± 0.001
H3K27me3	0.554 ± 0.014	0.591 ± 0.009	0.588 ± 0.004	0.599 ± 0.009	0.599 ± 0.015	**0.607** ± 0.008	0.590 ± 0.007
H3K36me3	0.549 ± 0.002	0.607 ± 0.008	0.602 ± 0.021	0.617 ± 0.004	0.618 ± 0.002	0.621 ± 0.007	**0.623** ± 0.005
H3K4me1	0.438 ± 0.007	0.471 ± 0.014	0.465 ± 0.014	0.487 ± 0.010	**0.497** ± 0.001	0.480 ± 0.003	0.487 ± 0.011
H3K4me2	0.523 ± 0.025	0.565 ± 0.008	0.538 ± 0.027	0.551 ± 0.005	0.563 ± 0.012	**0.573** ± 0.004	0.572 ± 0.003
H3K4me3	0.618 ± 0.007	0.617 ± 0.009	0.610 ± 0.055	0.624 ± 0.003	0.627 ± 0.017	0.634 ± 0.005	**0.639** ± 0.005
H3K9ac	0.497 ± 0.014	0.526 ± 0.009	0.525 ± 0.007	0.531 ± 0.002	0.534 ± 0.015	**0.569** ± 0.010	0.565 ± 0.007
H3K9me3	0.371 ± 0.026	0.435 ± 0.015	0.440 ± 0.009	0.469 ± 0.006	0.467 ± 0.023	0.470 ± 0.017	**0.485** ± 0.009
H4K20me1	0.617 ± 0.008	0.639 ± 0.009	0.644 ± 0.011	**0.646** ± 0.010	**0.646** ± 0.007	0.641 ± 0.007	0.640 ± 0.011
Regulatory Annotation							
Enhancer	0.479 ± 0.005	0.510 ± 0.017	0.483 ± 0.023	0.513 ± 0.001	0.519 ± 0.014	0.528 ± 0.009	**0.539** ± 0.012
Enhancer type	0.450 ± 0.003	0.471 ± 0.006	0.467 ± 0.012	0.478 ± 0.002	0.480 ± 0.010	**0.496** ± 0.008	0.488 ± 0.011
Promoter all	0.693 ± 0.007	0.742 ± 0.010	0.738 ± 0.007	0.737 ± 0.019	0.734 ± 0.013	**0.791** ± 0.009	0.748 ± 0.007
Promoter non-TATA	0.724 ± 0.004	0.764 ± 0.013	0.736 ± 0.025	0.756 ± 0.003	0.755 ± 0.010	**0.788** ± 0.005	0.760 ± 0.003
Promoter TATA	0.831 ± 0.057	0.761 ± 0.028	0.689 ± 0.038	0.818 ± 0.052	0.831 ± 0.038	0.840 ± 0.019	**0.916** ± 0.005
Splice Site Annotation							
Splice acceptor	0.820 ± 0.015	0.765 ± 0.006	0.760 ± 0.005	**0.952** ± 0.002	0.812 ± 0.032	0.754 ± 0.040	0.947 ± 0.012
Splice site all	0.849 ± 0.006	0.796 ± 0.021	0.764 ± 0.013	**0.966** ± 0.000	0.860 ± 0.007	0.760 ± 0.019	0.937 ± 0.011
Splice donor	0.840 ± 0.029	0.771 ± 0.013	0.781 ± 0.004	**0.962** ± 0.003	0.931 ± 0.021	0.706 ± 0.026	0.923 ± 0.003
Avg. MCC (↑)	0.599	0.612	0.600	0.650	0.637	0.626	**0.660**
Avg. Rank (↓)	6.06	4.89	5.67	3.06	3.06	3.00	**2.05**

D ADDITIONAL RESULTS

Revised NT Benchmark. Table 9 outlines the full results upon the revised NT Benchmark. We take the scores from the (Del Vecchio et al., 2025). Note that DNACHUNKER achieves best average performance and best rank.

Token Size Distribution. Figure 3 shows the token distribution of DNACHUNKER against BPEe tokenizer on the H3C14, H3C15 gene. Note that, BPE assigns tokens regardless of their biological relevance, while DNACHUNKER assigns fine chunks to biologically meaningful (phyloP regions) while assigning larger chunks to repetitive regions (SINE regions).