# Model Decides How to Tokenize: Adaptive DNA Sequence Tokenization with MxDNA

**Lifeng Qiao**[1,2][*], **Peng Ye**[1,3][†], **Yuchen Ren**[1,4], **Weiqiang Bai**[1], **Chaoqi Liang**[1],
**Xinzhu Ma**[1,3], **Nanqing Dong**[1], **Wanli Ouyang**[1,3]
[1]Shanghai Artificial Intelligence Laboratory, [2]Shanghai Jiao Tong University
[3]The Chinese University of Hong Kong, [4]The University of Sydney
yepeng@pjlab.org.cn

## Abstract

Foundation models have made significant strides in understanding the genomic
language of DNA sequences. However, previous models typically adopt the tok-
enization methods designed for natural language, which are unsuitable for DNA
sequences due to their unique characteristics. In addition, the optimal approach
to tokenize DNA remains largely under-explored, and may not be intuitively
understood by humans even if discovered. To address these challenges, we in-
troduce MxDNA, a novel framework where the model autonomously learns an
effective DNA tokenization strategy through gradient decent. MxDNA employs
a sparse Mixture of Convolution Experts coupled with a deformable convolu-
tion to model the tokenization process, with the discontinuous, overlapping, and
ambiguous nature of meaningful genomic segments explicitly considered. On
Nucleotide Transformer Benchmarks and Genomic Benchmarks, MxDNA demon-
strates superior performance to existing methods with less pretraining data and
time, highlighting its effectiveness. Finally, we show that MxDNA learns unique
tokenization strategy distinct to those of previous methods and captures genomic
functionalities at a token level during self-supervised pretraining. Our MxDNA
aims to provide a new perspective on DNA tokenization, potentially offering broad
applications in various domains and yielding profound insights. Code is available
at https://github.com/qiaoqiaoLF/MxDNA.

## 1 Introduction

Foundation models in natural language processing (NLP) have achieved remarkable success, trans-
forming how machines understand and generate human language [1, 2, 3]. Inspired by this success,
researchers are now exploring the application of foundation models to decode the complex "language"
of genomic sequences, aiming to potentially revolutionize our understanding of genomics [4, 5, 6, 7].
Tokenization, a critical initial step in NLP models, leverages human knowledge of natural language
structures such as grammar and punctuation to segment text into meaningful units. However, DNA
sequences present a distinct challenge: they lack natural delimiters and their "grammar" is not readily
understood by humans. These challenges make the tokenization process of DNA sequences not
straightforward.

Various tokenization methods have been employed by existing foundation models to analyse DNA
sequences [7, 4, 6, 5]. For example, single nucleotide tokenization [7] treats each nucleotide as an
individual token, K-mer [4, 6] segments the DNA blocks of k consecutive nucleotides, and Byte-Pair

---

[*]Work done during an internship at Shanghai Artificial Intelligence Laboratory.
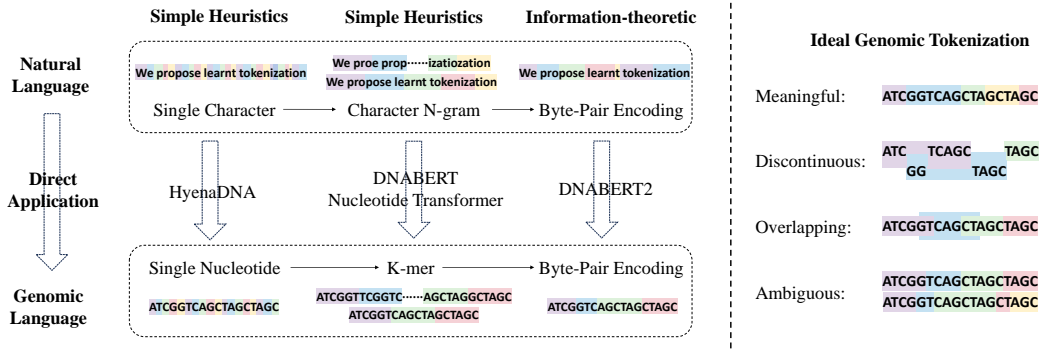[†]Corresponding Author.

Figure 1: Evolution of tokenization and Ideal Properties. **Left:** The progression from basic tokenization methods to more sophisticated techniques, with the direct but unsuitable applications from natural language to genomic language. **Right:** the ideal tokenization properties for genomics—Meaningful, Discontinuous, Overlapping, and Ambiguous—outlined in [8], which our MxDNA aims to achieve.

Encoding (BPE) [5] iteratively merges the most frequent pairs of existing tokens. All of these methods are borrowed directly from NLP as depicted in Fig. 1, each with its own inherent limitations. Single nucleotide tokenization, while offering high resolution for input, leads to an extremely large number of tokens, significantly increasing the complexity of the model. K-mer comes in two forms: overlapping and non-overlapping. Overlapping K-mer, despite its attempt to capture more contextual information, does not offer substantial benefits over single nucleotide approaches and can suffer from information leakage [5, 9]. Non-overlapping K-mer greatly reduces tokenized sequence length but can disrupt a potentially meaningful unit by splitting it into separate K-mers. BPE, adopted from NLP, attempts to optimize vocabulary size but often results in suboptimal segmentation that may not correspond to meaningful units [10, 11].

Unlike natural languages, where linguistically meaningful units such as words and sentences are almost standardized and well understood, the optimal approach to tokenize DNA remains under-explored due to the complex and varied nature of genomics. In NLP, common tokenization strategies have been validated by human knowledge, but such understanding does not extend to DNA. Consequently, rather than relying manually crafted tokenization rules, it may be better to trust a neural network to learn and determine the most effective tokenization strategy for genomic sequences. Additionally, recent research suggests that biologically meaningful protein tokens can be discontinuous, overlapping, and may require mapping to several tokenization possibilities [8, 12, 13], properties that are likely applicable to DNA sequences due to the genetic central dogma [14]. To handle these complexities, we can further equip our model with capabilities to manage discontinuities, overlaps, and the ambiguities of genomic sequences explicitly.

Building on the analysis above, we introduce MxDNA ("Mx" draws from **M**ixture of **Ex**perts [15]), a novel framework designed to autonomously learn an effective DNA tokenization strategy solely through gradient decent. The core of the framework starts with a sparse Mixture of Convolution Experts that identifies and embeds basic units within DNA sequences. Unlike conventional Mixture of Experts models, which focus on scaling up the model while maintaining computational efficiency [15, 16, 17], the experts in MxDNA are uniquely designed to capture DNA basic units of varied lengths. Following this, a deformable convolution [18, 19] assembles these basic units into final tokens. Throughout the process, MxDNA is explicitly equipped to manage the inherent discontinuities, overlaps and ambiguities in genomic sequences, enabling it to handle complex biological characteristics it encounters. Furthermore, we incorporate a cross-attention mechanism to align the output resolution with the original input during pretraining on a masked language modeling task [1].

The proposed MxDNA demonstrates strong performance on both the Nucleotide Transformer Benchmarks [6] and the Genomic Benchmarks [20]. Despite only being pretrained on human reference genome, it still outperforms or matches previous models [7, 4, 6, 5]—some pretrained on multi-species data and for much longer duration, achieving state-of-the-art average performance and the best on 15 of the 26 individual tasks. Finally, by visualizing the learnt tokenization process, we illustrate that MxDNA learns unique tokenization strategy distinct to those of previous methods

and captures genomic functionalities at a token level during self-supervised pretraining, potentially offering novel biological insights. Our contributions can be summarized as follows:

- **Learnt Tokenization**: We highlight the unsuitability of current DNA tokenization methods directly borrowed from NLP. Based on the belief that humans may not know the best tokenization approach but a model could potentially discover it, we propose a novel approach where the model autonomously learns an effective tokenization strategy.

- **Architectural Design**: We introduce a sparse Mixture of Convolution Experts coupled with a deformable convolution to dynamically learn tokenization, specifically designed to manage the inherent discontinuities overlaps and ambiguities in genomic sequences. Additionally, we leverage cross attention to align input and output sequence length to enable self-supervised pretraining.

- **Empirical Results**: MxDNA demonstrates robust quantitative performance with less pretraining data compared to some existing models, achieving state-of-the-art average performance on both Nucleotide Transformer Benchmarks and Genomic Benchmarks. Furthermore, visual analysis of the tokenization behaviour and the token embedding space highlights the unique strategy and capability to capture genomic functionalities at a token level of MxDNA, potentially offering new biological insights.

## 2 Background and Related Work

### 2.1 Tokenization Methods

Tokenization is a fundamental step in both natural language processing (NLP) and DNA sequence modelling, transforming complex texts or DNA sequences into manageable tokens. In NLP, whitespace tokenization uses spaces and punctuation as delimiters but faces out-of-vocabulary issues. Similarly, in both fields, character (or single nucleotide in DNA) tokenization provides high resolution but can lead to computational inefficiency [21, 22, 7, 23]. N-gram in NLP and K-mer in DNA analysis both use contiguous sequences of N (K) items from given inputs [24, 25, 6, 26, 4], but can disrupt meaningful units due to their fixed-length nature (non-overlapping) or lead to potential information redundancy or leakage (overlapping) [5, 9]. Byte-Pair Encoding (BPE) is employed across both domains to reduce vocabulary size by merging frequent pairs of existing tokens [27, 5]. However, it might not adequately encode more complex patterns and are unreliable for finding linguistically sound tokens [11, 10]. These rule-based methods show limitations in different aspects, and our study aim to develop a learning-based tokenization method without these limitations.

### 2.2 DNA Foundation Models

Recent advancements in DNA modeling have leveraged foundation models to decode the complex language of genomes. **DNABERT** [4] pioneers the use of a BERT-like pretrained model for genomic sequence analysis, enhancing the understanding of nucleotide relationships via attention mechanisms. **Nucleotide Transformer**[6] offers a comprehensive analysis of foundation models pretrained on DNA sequences, with model sizes reaching up to 2.5 billion parameters and pretraining data drawn from the 1000G[28] human genomes and 850 various species. **DNABERT2** [5] introduces an enhanced genome foundation model, utilizing an efficient BPE tokenizer and techniques to address input length constraints, resulting in reduced time and memory consumption while improving performance. **HyenaDNA** [7] introduces a genomic foundation model capable of handling context with 1 million tokens at single nucleotide resolution, enabling the first exploration of in-context learning in genomics. **DNAGPT** [26] extends the traditional GPT model by integrating tasks such as binary classification of DNA sequence order and numerical regression for predicting guanine-cytosine content, alongside developing a comprehensive token language. **Caduceus** [23] designs an architecture that leverages the long-range Mamba [29] block to support bi-directionality and reverse complementarity equivariance, addressing specific challenges in genomic analysis. Following VQVAE [30], **VQDNA** [31] employs a convolutional encoder alongside a vector-quantized codebook to model tokenization, sharing a similar motivation with us yet ultimately adopting distinct solutions. Each work offers unique insights and innovations to the filed. Our research specifically concentrate on the tokenization methods for DNA, hopefully providing our unique contributions to the filed.
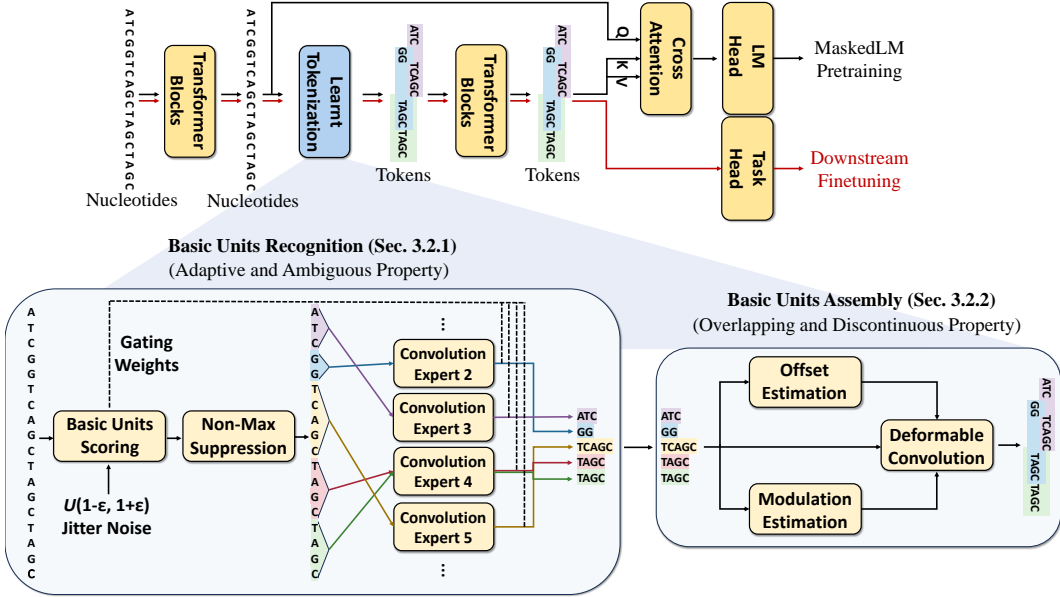
# 3 Method



Figure 2: Our proposed MxDNA. **(Top) Overall pipeline of the MxDNA model:** Black arrows indicate pretraining data flow, and red arrows indicate finetuning data flow. The learnt tokenization module tokenizes single nucleotide input into learnt tokens. **(Bottom) Illustration of the learnt tokenization module:** Meaningful basic units are recognized with a linearly scoring layer and non-maximum suppression, embedded through convolution experts (Sec. 3.2.1), and assembled into final tokens by a deformable convolution. (Sec. 3.2.2) This process ensures meaningful, discontinuous, overlapping, and ambiguous tokenization, addressing the unique properties of genomic data.

## 3.1 Motivation

The concept of "correct" tokenization in genomic sequences analysis remains undefined due to the complex nature of genomics. Unlike natural languages, where linguistically meaningful units are well-understood, biological units in genomics are not limited to contiguous nucleotide or amino acid sequences. Instead, they often encompass discontinuous, overlapping, and ambiguous segments crucial for understanding biological functions [8]. Current DNA modeling practices directly borrow tokenization methods from natural language processing (NLP), such as single nucleotide tokenization, K-mer and Byte-Pair Encoding (BPE). These fixed, predefined approaches, though useful, often fail to capture the unique properties of DNA sequences, which lack explicit delimiters and consist of biologically meaningful units that defy simple segmentation.

Recognizing these challenges, MxDNA was developed based on the belief that although an optimal tokenization schema for genomic sequences is yet to be discovered, we can explicitly equip our model with the desired tokenization properties—such as handling discontinuities, overlaps, and ambiguities, and allow it to learn and adapt its tokenization strategy all by itself.

## 3.2 Learnt Tokenization Module

This section introduces our learned tokenization module, which is central to our approach. The module first identifies meaningful basic units within the input sequence, which are then assembled into tokens with discontinuous, overlapping, and ambiguous properties. Implementation details are in Appx. A.2.

4

### 3.2.1 Basic Units Recognition

**Basic Units Scoring** Initially, MxDNA identifies the basic units that serve as the building blocks for tokens. It estimates the probability of the existence of various sized basic units centred at each nucleotide position by a linear gating mechanism commonly used in Mixture of Experts models. Following this, one-dimensional non-maximum suppression is applied to eliminate redundant proposals and select the most significant basic units.

Specifically, given the input nucleotide sequence $X \in \mathbb{R}^{l \times d}$, where $l$ is the sequence length and $d$ is the hidden dimension, $X$ is first linearly scored to produce $S \in \mathbb{R}^{l \times n}$, where $n$ represents the number of experts. Training-time multiplicative jitter noise [16] is applied to introduce **ambiguity**, while ensuring deterministic inference. The jitter noise is applied by multiplying the scores with a random factor sampled uniformly between $[1 - 0.01, 1 + 0.01]$, resulting in slight perturbations to the probability distribution used for tokenization.

Modified non-maximum suppression is then applied to $S$, where $S_{ij}$ indicates the presence probability of a basic unit of length $L_j$ centered at position $i$, and $L \in \mathbb{N}^n$ is a predefined set of lengths. The results are tracked using an expert mask $M \in \mathbb{N}^l$, where each $M_i$ is a natural integer indicating the presence of a basic unit's center of length $M_i$ at position $i$.

**Basic Units Embedding** After identifying the basic units, the nucleotides within each unit are aggregated to form embeddings. Convolution kernels of corresponding sizes are applied to the center of each basic unit to capture local features. The initial scoring and gating into specific convolution experts is similar to the Mixture of Experts paradigm, with each expert being a convolutional unit focusing on a specific segment rather than a single nucleotide.

Specifically, a basic unit at position $i$ of length $L_j = M_i$ is processed by the convolution expert $E_j$ with kernel size $L_j$, and weighted by $\text{softmax}(S_i)_j$, aggregating the nucleotides within the unit. This transforms the original input $X \in \mathbb{R}^{l \times d}$ into an array of basic units $U \in \mathbb{R}^{l \times d}$, where $k$ is the number of basic units:

$$
U_i = \begin{cases} E_j(X_{[i - \lceil \frac{M_i}{2} \rceil + 1 : i + \lfloor \frac{M_i}{2} \rfloor]}) \cdot \text{softmax}(S_i)_j, \text{where } L_j = M_i & , M_i > 0 \\ 0 & , M_i = 0 \end{cases} \tag{1}
$$

Then, the unwanted entries $\{i | M_i = 0\}$ of $U$ are removed to keep the basic units $U \in \mathbb{R}^{k \times d}$ only, where $k$ is the number of basic units.

### 3.2.2 Basic Units Assembly

**Distal Relation Estimation** Building upon the identified basic units, the more complex genomic patterns that extend beyond simple segmentation are modelled by a one-dimensional deformable convolution. This technique uniquely accommodates the modeling of complex local geometric transformations, adaptively adjusting to the input sequence. The linkages between the distal basic units are modeled by the offsets and modulation factors of each basic unit.

Following [18, 19], offsets $\Delta P \in \mathbb{R}^{k \times f}$ and modulation factors $\Delta M \in \mathbb{R}^{k \times f}$ are computed based on the basic units $U$ to model the distal relationships among them. This strategy ensures the combination of basic units is **discontinuous**, and reuses units across tokens achieve the **overlapping** property.

**Final Tokens Embedding** Using the computed offsets and modulation factors, deformable convolution is applied to embed basic units into final tokens. The embedding process for each position incorporates deformations of the convolution kernel specified by the offsets, with the results modulated by the modulation factors.

Specifically, a one-dimensional deformable convolution with kernel size $f$ is applied to embed these basic units into the final learnt tokens $T \in \mathbb{R}^{k \times d}$:

$$
T_i = \sum_{p \in \{-\lceil \frac{f}{2} \rceil + 1, \ldots, \lfloor \frac{f}{2} \rfloor\}} w_p \cdot U_{i + p + \Delta p} \cdot \Delta m \tag{2}
$$

For a fractional location $p' = i + p + \Delta p$, bilinear interpolation is implemented as:

$$U_{p'} = \sum_{q \in \{1,\dots,k\}} \max\left(0, 1 - |p' - q|\right) \cdot U_q \tag{3}$$

### 3.3 Overall Pipeline

The framework begins with single nucleotide input represented as $X_{input} \in \mathbb{R}^{l \times d}$. This single nucleotide resolution input allows for fine-grained analysis of genomics from the beginning.

Initially, $X_{input}$ is processed through several transformer encoder blocks designed to extract global relationships within the sequence, producing $X \in \mathbb{R}^{l \times d}$. This sets the stage for effective tokenization. Following this, the learnt tokenization module transforms the nucleotide sequence $X$ into a more manageable form $T \in \mathbb{R}^{k \times d}$, improving the efficiency and focus of subsequent layers. The tokenized output $T$ is then passed through another series of transformer encoder blocks to further refine the token representation to $T_{output} \in \mathbb{R}^{k \times d}$, enhancing the model's ability to encode deeper genomic information.

For the mask language modeling pretraining stage, the enriched nucleotide level representation $X$ serves as the query, with the refined tokenized output $T_{output}$ acting as both the key and value. This setup maps the output resolution to single nucleotides, essential for reconstructing masked tokens. During the finetuning stage, the [CLS] token of $T_{output}$ is used for classification by convention.

## 4 Experiments

In this section, we first introduce the implementation and pretraining settings of MxDNA. Then, we evaluate MxDNA against other foundation models on Genomic Benchmarks [20] and Nucleotide Transformer Benchmarks [6]. Next, we present ablation studies on the effect of different tokenization methods and different components of MxDNA. Finally, we conduct a simple analysis on the tokenization behaviors of MxDNA. Experiment settings and results are detailed in Appx. A.4.

### 4.1 Model Implementation & Pretraining

Our MxDNA is built on the architecture Nucleotide Transformer v2 100M model with 512 hidden units and 22 layers, totaling approximately 100M parameters. Specifically, the model's learnt tokenization module includes 10 convolution experts with kernel sizes ranging from 1 to 10, along with a deformable convolution block with a kernel size of three. We integrate this module by replacing the fifth transformer block, aiming to avoid introducing additional computations.

MxDNA is pretrained on the whole Human Reference Genome [32] on masked language modeling task [1] with 15% of the nucleotides randomly masked. An auxiliary balancing loss with a weight of 0.01 is used to prevent degradation towards a single expert, following [16]. The model undergoes training for 500k steps for main performance comparisons and 100k steps for ablations.

### 4.2 Downstream Evaluation

We primarily follow the evaluation settings of HyenaDNA [7], performing evaluation on Nucleotide Transformer Benchmarks and Genomic Benchmarks. To ensure fair comparison, we fully finetune all the BERT-like DNA foundation models including Nucleotide Transformer v2 [6], DNABERT [4], DNABERT2 [5], MxDNA under same hyperparameter settings. For HyenaDNA, we utilize the hyperparameters recommended by [7, 23]. All experiments are repeated with three random seeds, and we report the average performance with sample standard deviations. [3]

### 4.2.1 Genomic Benchmarks

First, we begin our evaluation on the Genomic Benchmarks [20], which consists of eight regulatory element classification tasks. For this benchmark, all BERT-like models are finetuned for 10 epochs with Top-1 accuracy reported for each dataset.

---

[3]The Nucleotide Transformer and DNABERT2 are pretrained on much larger datasets than other models.

Table 1: Genomic Benchmarks. Average performance across three random seeds for Nucleotide Transformer v2 100M, DNABERT, DNABERT2, HyenaDNA and MxDNA with sample standard deviations. We highlight the best values in **bold** type and underline the second best.

| Dataset | NTv2 | DNABERT | DNABERT2 | HyenaDNA | MxDNA |
|---|---|---|---|---|---|
| Average | 88.13 $\pm$ 0.03 | 87.50 $\pm$ 0.13 | <u>88.29</u> $\pm$ 0.19 | 87.17 $\pm$ 0.15 | **89.13** $\pm$ 0.13 |
| Mouse Enhancers | **83.94** $\pm$ 0.41 | <u>81.54</u> $\pm$ 0.86 | 81.34 $\pm$ 0.84 | 80.99 $\pm$ 0.72 | 80.57 $\pm$ 0.97 |
| Coding vs Intergenomic | 94.50 $\pm$ 0.06 | 93.13 $\pm$ 0.05 | <u>94.94</u> $\pm$ 0.34 | 90.74 $\pm$ 0.11 | **95.28** $\pm$ 0.08 |
| Human vs Worm | 96.88 $\pm$ 0.18 | 96.98 $\pm$ 0.07 | <u>97.57</u> $\pm$ 0.04 | 96.53 $\pm$ 0.04 | **97.64** $\pm$ 0.01 |
| Human Enhancer Cohn | 74.33 $\pm$ 0.40 | 74.54 $\pm$ 0.27 | **75.93** $\pm$ 0.20 | 73.36 $\pm$ 0.15 | <u>74.67</u> $\pm$ 0.09 |
| Human Enhancer Ensembl | 92.05 $\pm$ 0.38 | 92.18 $\pm$ 0.14 | <u>92.31</u> $\pm$ 0.03 | 88.12 $\pm$ 0.17 | **93.13** $\pm$ 0.35 |
| Human Regulatory | <u>93.79</u> $\pm$ 0.12 | 88.16 $\pm$ 0.09 | 87.94 $\pm$ 0.54 | 93.08 $\pm$ 0.22 | **94.11** $\pm$ 0.08 |
| Human OCR Ensembl | 78.51 $\pm$ 0.55 | **81.40** $\pm$ 0.11 | 80.94 $\pm$ 0.09 | 79.15 $\pm$ 0.34 | <u>81.05</u> $\pm$ 0.07 |
| Human NonTATA Promoters | 91.05 $\pm$ 0.47 | 92.06 $\pm$ 0.20 | 95.34 $\pm$ 0.17 | <u>95.39</u> $\pm$ 0.26 | **96.56** $\pm$ 0.29 |

Table 2: Nucleotide Transformer Benchmarks. Average performance across three random seeds for Nucleotide Transformer v2 100M, DNABERT, DNABERT2, HyenaDNA and MxDNA with sample standard deviations. We highlight the best values in **bold** type and underline the second best.

| Dataset | NTv2 | DNABERT | DNABERT2 | HyenaDNA | MxDNA |
|---|---|---|---|---|---|
| Average | 70.70 $\pm$ 0.12 | 68.61 $\pm$ 0.16 | <u>76.66</u> $\pm$ 0.20 | 75.96 $\pm$ 0.20 | **78.14** $\pm$ 0.12 |
| Histone Markers Avg. | 55.22 $\pm$ 0.21 | 51.67 $\pm$ 0.17 | <u>65.89</u> $\pm$ 0.46 | 65.24 $\pm$ 0.26 | **68.14** $\pm$ 0.19 |
| H3 | 78.22 $\pm$ 1.15 | 77.41 $\pm$ 1.01 | <u>82.31</u> $\pm$ 0.22 | 80.86 $\pm$ 0.52 | **82.78** $\pm$ 0.14 |
| H3K14ac | 51.76 $\pm$ 0.99 | 46.51 $\pm$ 1.83 | 65.13 $\pm$ 1.10 | <u>65.96</u> $\pm$ 0.26 | **68.27** $\pm$ 0.19 |
| H3K36me3 | 59.18 $\pm$ 0.53 | 50.98 $\pm$ 0.75 | **68.19** $\pm$ 0.82 | 64.31 $\pm$ 0.33 | <u>67.05</u> $\pm$ 1.05 |
| H3K4me1 | 51.87 $\pm$ 1.09 | 43.83 $\pm$ 0.34 | <u>56.09</u> $\pm$ 0.48 | 55.04 $\pm$ 1.07 | 56.15 $\pm$ 0.63 |
| H3K4me2 | 29.63 $\pm$ 1.68 | 32.38 $\pm$ 0.98 | 49.25 $\pm$ 2.06 | <u>49.96</u> $\pm$ 0.90 | **55.59** $\pm$ 1.08 |
| H3K4me3 | 38.76 $\pm$ 1.30 | 31.49 $\pm$ 3.40 | 57.90 $\pm$ 0.92 | <u>60.92</u> $\pm$ 0.72 | **63.68** $\pm$ 0.34 |
| H3K79me3 | 60.98 $\pm$ 1.06 | 60.48 $\pm$ 0.50 | <u>71.94</u> $\pm$ 0.44 | 70.97 $\pm$ 0.77 | **74.29** $\pm$ 0.08 |
| H3K9ac | 54.57 $\pm$ 0.71 | 52.55 $\pm$ 0.85 | <u>64.35</u> $\pm$ 1.03 | 62.57 $\pm$ 0.46 | **64.78** $\pm$ 0.50 |
| H4 | 79.19 $\pm$ 0.49 | 79.60 $\pm$ 0.55 | <u>80.71</u> $\pm$ 0.43 | 78.73 $\pm$ 0.66 | **81.18** $\pm$ 0.25 |
| H4ac | 48.02 $\pm$ 1.40 | 41.53 $\pm$ 0.24 | 63.03 $\pm$ 0.81 | <u>63.06</u> $\pm$ 0.62 | **67.65** $\pm$ 0.39 |
| Regulatory Annotation Avg. | 84.86 $\pm$ 0.01 | 84.83 $\pm$ 0.29 | **86.16** $\pm$ 0.17 | 84.87 $\pm$ 0.20 | **86.16** $\pm$ 0.07 |
| Enhancer | 78.16 $\pm$ 0.29 | 79.13 $\pm$ 0.56 | <u>79.40</u> $\pm$ 0.29 | 78.81 $\pm$ 0.54 | **79.90** $\pm$ 0.39 |
| Enhancer Types | 58.14 $\pm$ 1.02 | 54.73 $\pm$ 1.24 | <u>59.84</u> $\pm$ 0.45 | 58.36 $\pm$ 0.58 | **60.50** $\pm$ 0.59 |
| Promoter All | 96.23 $\pm$ 0.61 | 97.05 $\pm$ 0.05 | **97.37** $\pm$ 0.04 | 96.19 $\pm$ 0.14 | <u>97.16</u> $\pm$ 0.12 |
| Promoter Non-TATA | 96.69 $\pm$ 0.16 | 97.02 $\pm$ 0.16 | **97.65** $\pm$ 0.12 | 96.43 $\pm$ 0.09 | <u>97.24</u> $\pm$ 0.14 |
| Promoter TATA | 95.10 $\pm$ 0.24 | <u>96.22</u> $\pm$ 0.33 | **96.55** $\pm$ 0.24 | 94.55 $\pm$ 0.38 | 96.01 $\pm$ 0.09 |
| Splice Site Annotation Avg. | **98.71** $\pm$ 0.05 | 98.02 $\pm$ 0.09 | 96.69 $\pm$ 0.09 | 96.82 $\pm$ 0.11 | <u>98.09</u> $\pm$ 0.09 |
| All | **98.36** $\pm$ 0.04 | 97.83 $\pm$ 0.06 | 95.68 $\pm$ 0.11 | 96.78 $\pm$ 0.18 | <u>98.14</u> $\pm$ 0.08 |
| Accpetor | **98.69** $\pm$ 0.14 | 97.81 $\pm$ 0.28 | 97.71 $\pm$ 0.11 | 96.52 $\pm$ 0.19 | <u>98.01</u> $\pm$ 0.13 |
| Donor | **99.09** $\pm$ 0.05 | <u>98.43</u> $\pm$ 0.05 | 96.67 $\pm$ 0.17 | 97.16 $\pm$ 0.16 | 98.10 $\pm$ 0.13 |

As shown in Table 1, MxDNA achieves the best performance on 5 out of 8 datasets and ranks in the top-2 on 7 out o f 8 datasets. On average, MxDNA shows an improvement of 0.84 points compared to the second-best model, DNABERT2. These results demonstrate MxDNA's robustness and effectiveness in regulatory element classification.

### 4.2.2 Nucleotide Transformer Benchmarks

Next, we evaluate MxDNA on the Nucleotide Transformer Benchmarks [6], which includes 18 datasets across three task types: histone marker prediction, regulatory annotation prediction, and splice site annotation prediction. For this benchmark, the BERT-like models are finetuned for 20 epochs. Following [23], we use the Matthews Correlation Coefficient (MCC) for histone markers tasks, F1 score for regulatory and splice site annotation tasks, except accuracy for splice site all task.

Table 3: Average results on Nucleotide Transformer Benchmarks and Genomic Benchmarks with different tokenization methods. We highlight the best values in **bold** type, <u>underline</u> the second best.

| Method | NT Benchmarks | Genomic Benchmarks |
|---|---|---|
| Single Nucleotide (1-mer) | <u>75.07</u> $\pm$ 0.26 | <u>88.56</u> $\pm$ 0.02 |
| Overlapping k-mer (6-mer) | 74.35 $\pm$ 0.35 | 88.55 $\pm$ 0.07 |
| Non-overlapping k-mer (6-mer) | 67.65 $\pm$ 0.17 | 86.83 $\pm$ 0.06 |
| Byte-pair Encoding (4096 tokens) | 74.96 $\pm$ 0.16 | 87.30 $\pm$ 0.16 |
| MxDNA (Learnt Tokenization) | **77.52** $\pm$ 0.18 | **88.89** $\pm$ 0.05 |

Table 4: Average results on Nucleotide Transformer Benchmarks and Genomic Benchmarks with components added successively. We highlight the best values in **bold** type, <u>underline</u> the second best.

| Method | NT Benchmarks | Genomic Benchmarks |
|---|---|---|
| Single Nucleotide Baseline | 75.07 $\pm$ 0.26 | 88.56 $\pm$ 0.02 |
| + Mixture of Convolution Experts | 77.00 $\pm$ 0.05 | 88.72 $\pm$ 0.07 |
| + Deformable Convolution | <u>77.35</u> $\pm$ 0.12 | <u>88.86</u> $\pm$ 0.18 |
| + Jitter Noise (MxDNA) | **77.52** $\pm$ 0.18 | **88.89** $\pm$ 0.05 |

As shown in Table 2, MxDNA achieves the best performance on 10 out of 18 datasets and ranks in the top-2 on 16 out of 18 datasets. On average, MxDNA shows an improvement of 1.48 points compared to the second-best model, DNABERT2. Notably, MxDNA significantly outperforms all other models in the histone markers tasks while maintaining competitive performance in regulatory annotation and splice site annotation tasks.

### 4.3 Ablation Studies

**Different Tokenization Methods:** We compare various tokenization methods by pretraining models for 100k steps using the same backbone but different tokenization methods. The results in Table 3 show that our learnt tokenization significantly outperforms traditional methods such as non-overlapping K-mer, BPE, overlapping K-mer, and single nucleotide tokenization. Among the rule-based methods, single nucleotide tokenization performs best, possibly because it doesn't incorporate human biases and focuses solely on the raw data, though it may make it difficult to capture higher-level semantics. Conversely, non-overlapping K-mer might disrupt meaningful units, BPE might fail to segment DNA sequences meaningfully, and overlapping K-mer could suffer from information leakage.

**Different Components:** We assess the impact of individual components by incrementally integrating each into the baseline model and pretraining them for 100k steps. Starting with a baseline of single nucleotide tokenization, we sequentially add the Mixture of Convolution Experts, the deformable convolution and jitter noise, resulting in the proposed MxDNA. The results in Table 4 show substantial performance gains from the Mixture of Convolution Experts alone, demonstrating the effectiveness of the idea which allows the model to learn tokenization autonomously rather than depending on predefined tokenization. There are also noticeable performance improvements contributed by the deformable convolution and jitter noise, showing the effectiveness of explicitly equipping the model with capabilities to handle discontinuities, overlaps and ambiguities.

### 4.4 Analysis

We conduct an analysis of the tokenization behaviors of MxDNA against previous methods on both a sample and dataset level, and present a output embedding analysis at a token level. Notably, MxDNA exhibits unique tokenization strategy distinct from prior methods and is able to inherently capture and differentiate genomic functionalities at a token level during self-supervised pretraining, potentially providing new biological insights. Visualization details are in Appx A.6.

**Sample Level** We first visualize the tokenization of a DNA sequence. For MxDNA, two individual forward passes with identical input and model yield slightly different results during training. It is
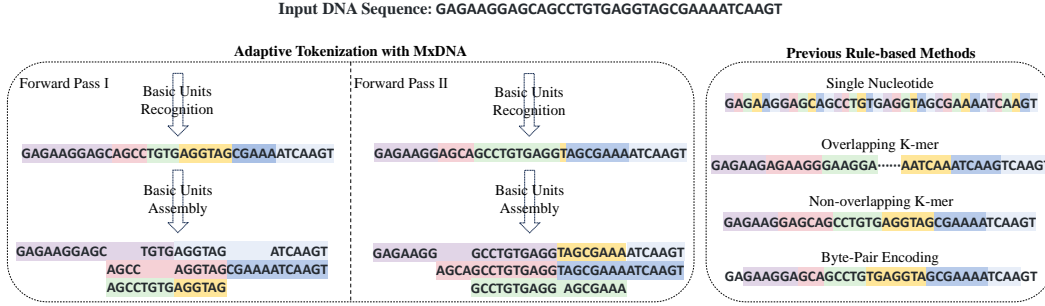
Figure 3: Tokenization results of MxDNA over two individual forward passes (left) compared to those of traditional rule-based methods (right). A block of the same colour refers to a single token.
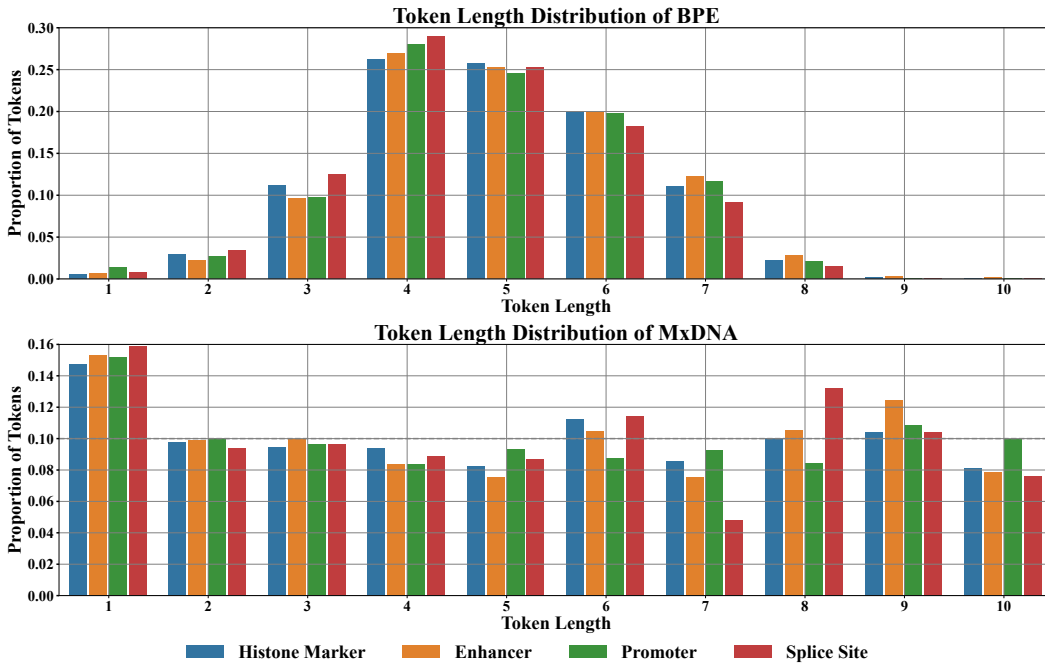


Figure 4: Distribution of token lengths for BPE (top) and MxDNA (bottom) across different downstream datasets, illustrating the distinct strategy of MxDNA for handling DNA tokenization. For the sake of simplicity, we regard the basic units as tokens for MxDNA.

worth mentioning that there are usually a small number of differences between the two results, and we display the region where the tokenization outcomes are different to show the ambiguous property for illustrative purposes. For previous rule-based methods, tokenization is static and performed only once. As depicted in Fig. 3, our learnt tokenization method tokenize the DNA sequence in a way distinctly different from previous rule-based method. Moreover, the discontinuous, overlapping and ambiguous tokenization results validate our design choices to manage these properties.

**Dataset Level**    To gain more insights, we measure the distribution of token lengths across different downstream datasets for both MxDNA and BPE. For simplicity, we regard the basic units as tokens for MxDNA. BPE and MxDNA shows very distinct distribution of token lengths. As shown in Fig. 4, BPE tends to produce a bell-shaped distribution, inherently biased by its frequency-based merging rule. Conversely, MxDNA's distribution is closer to a uniform distribution with preferences for specific lengths, reflecting its adaptive, task-oriented segmentation capabilities. Moreover, the
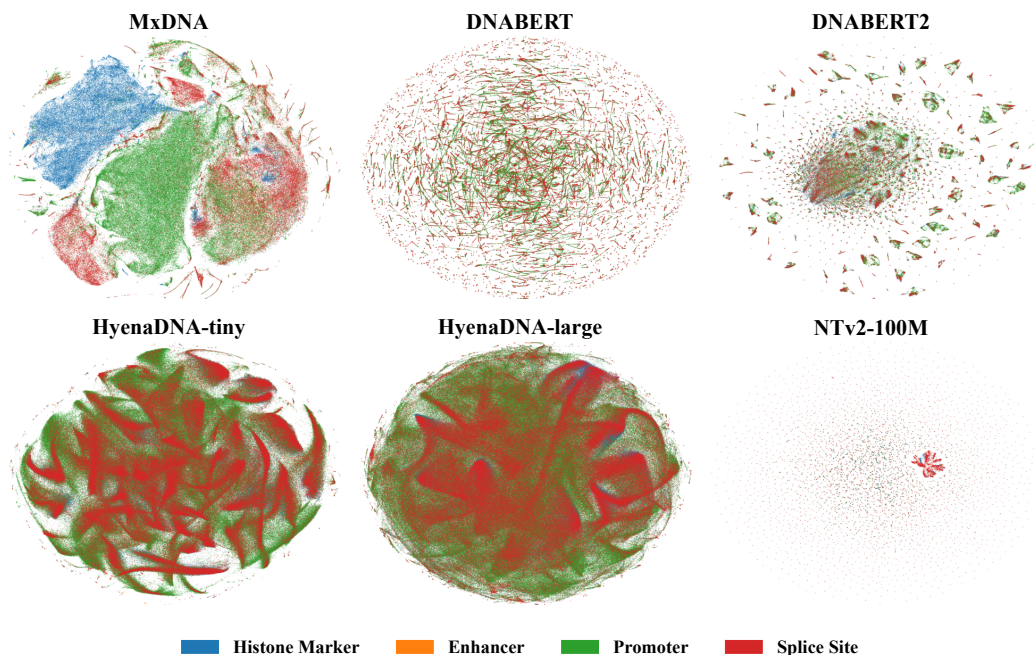
Figure 5: t-SNE visualization of the output embeddings at a token level across different functional sequences of different models, demonstrating MxDNA's unique capability to inherently capture and differentiate genomic functionalities at a token level.

variability in token distribution across datasets might suggest that DNA sequences of different functions might possess distinct patterns and meaningful units.

**Token Embedding Analysis**   Next, we use t-SNE to visualize the pretrained output tokens in sequences with different genomic functions of different foundation models. As is shown in Fig. 4.4, without any finetuning, the token embedding distributions of MxDNA are different across sequences with different functions: the tokens of Histone Marker, Promoter and Splice Site form unique clusters. While for all other foundation models, their tokens do not form clear clusters as MxDNA does. This shows MxDNA's superior capability to inherently capture and differentiate genomic functionalities at a token level, suggesting its robustness and specificity in representing biological sequences even before any supervised finetuning is applied.

## 5   Conclusion

**Summary**   We present MxDNA, a framework developed to autonomously learn effective DNA tokenization strategies solely through gradient descent. MxDNA demonstrates strong performance against existing sota models and tokenization methods across 26 diverse genomic tasks in Nucleotide Transformer Benchmarks and Genomic Benchmarks with no additional cost. We also perform an analysis of the tokenization mechanism and the token embedding space of MxDNA, showing its distinct tokenization strategy against previous methods and unique capability to capture genomic functionalities at a token level.

**Limitations & Future Works**   While MxDNA demonstrates strong quantitative performance on various downstream tasks, direct biological validation of the model's tokenization decision remains limited. Furthermore, the evaluation on long range tasks is lacking due to quadratic cost of self-attention, although the learnt tokenization is expected to help reduce sequence length effectively and can be combined with sub-quadratic architectures [29, 33]. Future research will focus on refining MxDNA's design to learn a better and more interpretable tokenization strategy, and testing its applicability to broader genomic analyses especially on more long range tasks.

10

## Acknowledgments and Disclosure of Funding

## References

[1] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[2] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[3] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[4] 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.

[5] Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana Davuluri, and Han Liu. Dnabert-2: Efficient foundation model and benchmark for multi-species genome. *arXiv preprint arXiv:2306.15006*, 2023.

[6] 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, Guillaume Richard, Marcin Skwark, Karim Beguir, Marie Lopez, and Thomas Pierrot. The nucleotide transformer: Building and evaluating robust foundation models for human genomics. *bioRxiv*, pages 2023–01, 2023.

[7] 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, 2024.

[8] Mai Ha Vu, Rahmad Akbar, Philippe A Robert, Bartlomiej Swiatczak, Geir Kjetil Sandve, Victor Greiff, and Dag Trygve Truslew Haug. Linguistically inspired roadmap for building biologically reliable protein language models. *Nature Machine Intelligence*, 5(5):485–496, 2023.

[9] Chaoqi Liang, Weiqiang Bai, Lifeng Qiao, Yuchen Ren, Jianle Sun, Peng Ye, Hongliang Yan, Xinzhu Ma, Wangmeng Zuo, and Wanli Ouyang. Rethinking the bert-like pretraining for dna sequences. *arXiv preprint arXiv:2310.07644*, 2023.

[10] Kaj Bostrom and Greg Durrett. Byte pair encoding is suboptimal for language model pretraining. *arXiv preprint arXiv:2004.03720*, 2020.

[11] Valentin Hofmann, Janet B Pierrehumbert, and Hinrich Schütze. Superbizarre is not superb: Derivational morphology improves bert's interpretation of complex words. *arXiv preprint arXiv:2101.00403*, 2021.

[12] Philippe A Robert, Rahmad Akbar, Robert Frank, Milena Pavlović, Michael Widrich, Igor Snapkov, Andrei Slabodkin, Maria Chernigovskaya, Lonneke Scheffer, Eva Smorodina, et al. Unconstrained generation of synthetic antibody–antigen structures to guide machine learning methodology for antibody specificity prediction. *Nature Computational Science*, 2(12):845–865, 2022.

[13] Rahmad Akbar, Philippe A Robert, Milena Pavlović, Jeliazko R Jeliazkov, Igor Snapkov, Andrei Slabodkin, Cédric R Weber, Lonneke Scheffer, Enkelejda Miho, Ingrid Hobæk Haff, et al. A compact vocabulary of paratope-epitope interactions enables predictability of antibody-antigen binding. *Cell Reports*, 34(11), 2021.

[14] Francis Crick. Central dogma of molecular biology. *Nature*, 227(5258):561–563, 1970.

[15] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

[16] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[17] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[18] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 764–773, 2017.

[19] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9308–9316, 2019.

[20] 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.

[21] Yoon Kim, Yacine Jernite, David Sontag, and Alexander Rush. Character-aware neural language models. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[22] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones. Character-level language modeling with deeper self-attention. In *Proceedings of the AAAI conference on artificial intelligence*, volume 33, pages 3159–3166, 2019.

[23] Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range dna sequence modeling. *arXiv preprint arXiv:2403.03234*, 2024.

[24] Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer. Class-based n-gram models of natural language. *Computational linguistics*, 18(4):467–480, 1992.

[25] Paul McNamee and James Mayfield. Character n-gram tokenization for european language text retrieval. *Information retrieval*, 7:73–97, 2004.

[26] Daoan Zhang, Weitong Zhang, Bing He, Jianguo Zhang, Chenchen Qin, and Jianhua Yao. Dnagpt: A generalized pretrained tool for multiple dna sequence analysis tasks. *bioRxiv*, pages 2023–07, 2023.

[27] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[28] Marta Byrska-Bishop, Uday S Evani, Xuefang Zhao, Anna O Basile, Haley J Abel, Allison A Regier, André Corvelo, Wayne E Clarke, Rajeeva Musunuri, Kshithija Nagulapalli, et al. High-coverage whole-genome sequencing of the expanded 1000 genomes project cohort including 602 trios. *Cell*, 185(18):3426–3440, 2022.

[29] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[30] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[31] Siyuan Li, Zedong Wang, Zicheng Liu, Di Wu, Cheng Tan, Jiangbin Zheng, Yufei Huang, and Stan Z Li. Vqdna: Unleashing the power of vector quantization for multi-species genomic sequence modeling. *arXiv preprint arXiv:2405.10812*, 2024.

[32] Valerie A Schneider, Tina Graves-Lindsay, Kerstin Howe, Nathan Bouk, Hsiu-Chuan Chen, Paul A Kitts, Terence D Murphy, Kim D Pruitt, Françoise Thibaud-Nissen, Derek Albracht, et al. Evaluation of grch38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly. *Genome research*, 27(5):849–864, 2017.

[33] 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*, pages 28043–28078. PMLR, 2023.

[34] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR, 2020.

[35] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[36] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International conference on machine learning*, pages 8821–8831. Pmlr, 2021.

[37] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[38] Peter Anderson, Xiaodong He, Chris Buehler, Damien Teney, Mark Johnson, Stephen Gould, and Lei Zhang. Bottom-up and top-down attention for image captioning and visual question answering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6077–6086, 2018.

[39] Emanuele Bugliarello, Ryan Cotterell, Naoaki Okazaki, and Desmond Elliott. Multimodal pretraining unmasked: A meta-analysis and a unified framework of vision-and-language berts. *Transactions of the Association for Computational Linguistics*, 9:978–994, 2021.

[40] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015.

[41] Delong Chen, Samuel Cahyawijaya, Jianfeng Liu, Baoyuan Wang, and Pascale Fung. Subobject-level image tokenization. *arXiv preprint arXiv:2402.14327*, 2024.

[42] Young Kyung Kim, J Matías Di Martino, and Guillermo Sapiro. Vision transformers with natural language semantics. *arXiv preprint arXiv:2402.17863*, 2024.

[43] Alexander Kirillov, Eric Mintun, Nikhila Ravi, Hanzi Mao, Chloe Rolland, Laura Gustafson, Tete Xiao, Spencer Whitehead, Alexander C Berg, Wan-Yen Lo, et al. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 4015–4026, 2023.

[44] Nithin Chalapathi, Yiheng Du, and Aditi Krishnapriyan. Scaling physics-informed hard constraints with mixture-of-experts. *arXiv preprint arXiv:2402.13412*, 2024.

[45] Anmol Gulati, James Qin, Chung-Cheng Chiu, Niki Parmar, Yu Zhang, Jiahui Yu, Wei Han, Shibo Wang, Zhengdong Zhang, Yonghui Wu, et al. Conformer: Convolution-augmented transformer for speech recognition. *arXiv preprint arXiv:2005.08100*, 2020.

[46] Haiping Wu, Bin Xiao, Noel Codella, Mengchen Liu, Xiyang Dai, Lu Yuan, and Lei Zhang. Cvt: Introducing convolutions to vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 22–31, 2021.

[47] Tete Xiao, Mannat Singh, Eric Mintun, Trevor Darrell, Piotr Dollár, and Ross Girshick. Early convolutions help transformers see better. *Advances in neural information processing systems*, 34:30392–30400, 2021.

[48] 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.

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

[50] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[51] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[52] Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359, 2022.

[53] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*, 2023.

[54] Jason Ansel, Edward Yang, Horace He, Natalia Gimelshein, Animesh Jain, Michael Voznesensky, Bin Bao, Peter Bell, David Berard, Evgeni Burovski, Geeta Chauhan, Anjali Chourdia, Will Constable, Alban Desmaison, Zachary DeVito, Elias Ellison, Will Feng, Jiong Gong, Michael Gschwind, Brian Hirsh, Sherlock Huang, Kshiteej Kalambarkar, Laurent Kirsch, Michael Lazos, Mario Lezcano, Yanbo Liang, Jason Liang, Yinghai Lu, CK Luk, Bert Maher, Yunjie Pan, Christian Puhrsch, Matthias Reso, Mark Saroufim, Marcos Yukio Siraichi, Helen Suk, Michael Suo, Phil Tillet, Eikan Wang, Xiaodong Wang,

William Wen, Shunting Zhang, Xu Zhao, Keren Zhou, Richard Zou, Ajit Mathews, Gregory Chanan, Peng Wu, and Soumith Chintala. PyTorch 2: Faster Machine Learning Through Dynamic Python Bytecode Transformation and Graph Compilation. In *29th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2 (ASPLOS '24)*. ACM, April 2024.

[55] William Falcon and The PyTorch Lightning team. PyTorch Lightning, March 2019.

[56] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[57] Wenzel Jakob, Jason Rhinelander, and Dean Moldovan. pybind11 – seamless operability between c++11 and python, 2017. https://github.com/pybind/pybind11.

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[59] Charles R. Harris, K. Jarrod Millman, Stéfan J. van der Walt, Ralf Gommers, Pauli Virtanen, David Cournapeau, Eric Wieser, Julian Taylor, Sebastian Berg, Nathaniel J. Smith, Robert Kern, Matti Picus, Stephan Hoyer, Marten H. van Kerkwijk, Matthew Brett, Allan Haldane, Jaime Fernández del Río, Mark Wiebe, Pearu Peterson, Pierre Gérard-Marchant, Kevin Sheppard, Tyler Reddy, Warren Weckesser, Hameer Abbasi, Christoph Gohlke, and Travis E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020.

[60] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[61] Michael L. Waskom. seaborn: statistical data visualization. *Journal of Open Source Software*, 6(60):3021, 2021.

# A    Appendix / supplemental material

## A.1    Extended Related Works

### A.1.1    Image Tokenization

Tokenization in computer vision (CV) attempts to transform images into formats that can be efficiently processed by machine learning models especially transformers. In line with Character-level tokenization, [34] directly using raw pixels as input units. The Vision Transformer (ViT) [35] splits an image into patches of identical sizes and treats these patches as tokens in NLP, demonstrating remarkable performance on standard image recognition tasks. Inspired by [30], [36, 37] utilized an image tokenizer learnt by discrete variational autoencoder (dVAE) to map pixels into discrete tokens according to a visual codebook. Meanwhile, researchers also utilize detection or segmentation features for visual representations. For instance, [38, 39] used a pretrained Faster R-CNN model [40] to extract region features. Recently, [41, 42] exploit Segment Anything Model (SAM) [43] to construct a sub-word tokenization and semantic tokenization respectively.

### A.1.2    Mixture of Experts

(Sparse) MoE is first designed to improve the capacity of neural networks while maintaining total computations. [15] uses MoE as a general purpose neural network component and realizes sparse gating, demonstrating its use as a practical way to massively increase model capacity. By replacing FNN with Mixture of Experts, [16, 17] successfully combine sparse MoE and Transformers, achieving superior capabilities with less computational cost. Previous methods generally replace a layer of the neural networks with multiple, sparsely activated identical alternatives, governed by a gating mechanism. Recently, [44] explicitly adds interpretability to each expert by letting each expert solve the constraint over smaller decomposed domains through differentiable optimization.

### A.1.3    Deformable Convolution

Deformable convolution explicitly equips the model with ability to adapt to the geometric variations of different objects [18, 19]. Unlike the attention mechanism, which focuses on capturing long-range relationships, deformable convolution locally samples feature maps using learnt offsets and modulation factors. By modeling complex geometric transformations effectively, deformable convolution networks achieve significant performance improvements in various tasks, including image classification, object detection and semantic segmentation.

### A.1.4    Combination of Convolution and Transformer

The integration of convolutional layers with transformer architectures has emerged as a powerful approach across various domains, effectively combining the strengths of both techniques. Conformer [45] applies this hybrid design to audio processing, enhancing the capture of local and global dependencies in audio signals. In computer vision, the CvT [46] introduces convolutions into transformers to improve efficiency and representational power, while Early Convolution [47] in Vision Transformer incorporates convolutional layers early in the architecture to enhance input feature representations. Extending to genomic data, Enformer [48] applies a similar approach as in [47] to model complex dependencies in DNA sequences and reduce the computational cost, showcasing the potential of hybrid architectures to handle highly specialized data types like genomic sequences.

## A.2    Method Details

For the convolution expert, we adapt design principles from [45]. Our 1-D convolution expert starts with a pointwise convolution $W_{pj}^{in}$ paired with a Gated Linear Unit (GLU), followed by a 1-D grouped convolution $W_{gj}$. Subsequent to the grouped convolution, a Layer Normalization (LayerNorm) and a Swish activation layer $W_{pj}^{out}(\text{Swish}())$ are applied. The grouped convolution here has number of groups equal to the factor of hidden size closest to kernel size, and number of output channel equal to number of input channels. This ensures that each convolution expert has similar parameter counts in spite of different kernel sizes.

$$E_j\left(X\right) = W_{pj}^{out}\left(\text{Swish}\left(\text{LayerNorm}(W_{gj} * \text{GLU}\left(W_{pj}^{in}X\right))\right)\right) \tag{4}$$

15

| Term | Description |
|------|-------------|
| $l$ | Number of nucleotides |
| $d$ | Dimension of hidden states |
| $n$ | Number of experts |
| $k$ | Number of basic units |
| $f$ | Kernel size of the deformable convolution |
| $i$ | Indices of nucleotides or tokens |
| $j$ | Indices of experts |
| $\mathbf{X} \in \mathbb{R}^{l \times d}$ | Input nucleotide sequence |
| $\mathbf{S} \in \mathbb{R}^{l \times n}$ | Confidence scores of basic units existence |
| $\mathbf{L} \in \mathbb{N}^{n}$ | Kernel sizes of convolution experts |
| $\mathbf{M} \in \mathbb{N}^{l}$ | Mask indicating the existence of basic units |
| $\mathbf{E_j} \in \mathbb{R}^{L_j \times d} \to \mathbb{R}^{d}$ | Convolution experts |
| $\mathbf{U} \in \mathbb{R}^{k \times d}$ | Basic units |
| $\Delta \mathbf{P} \in \mathbb{R}^{k \times f}$ | Offsets ofthe deformable convolution |
| $\Delta \mathbf{M} \in \mathbb{R}^{k \times f}$ | Modulation factors of the deformable convolution |
| $\mathbf{T} \in \mathbb{R}^{k \times d}$ | Final tokens |

Table 5: Glossary of terms used in describing the method.

### A.2.1 Non-Maximum Suppression

The pseudocode in 1 describes the selection process for optimal basic units based on scores, ensuring no overlaps, and using kernel sizes to guide the selection.

The input consists of: positions (all nucleotide positions), kernel sizes (all kernel sizes), scores (scores for each (position, kernel size) pair) for the possibility of a basic unit of a given size existing at a given position. The output is a mask indicating selected basic units with their corresponding kernel sizes.

---

**Algorithm 1** Detailed Non-Maximum Suppression for Basic Unit Placement

---

1: **procedure** NMS(positions $P = [1, 2, \ldots, l]$, kernel sizes $L \in \mathbb{N}^{n}$, scores $S \in \mathbb{R}^{l \times n}$)
2:     Sort all $(P_i, L_j)$ pairs by $S_{ij}$ in descending order, where $i \in [1, 2, \ldots, l]$ and $j \in [1, 2, \ldots, n]$.
3:     Initialize an output array with zeros $M \in \mathbb{N}^{l}$.
4:     **for** each $(P_i, L_j)$ pair in the sorted pairs **do**
5:         Calculate the start and end of the region at $P_i$ with width $L_j$.
6:         **if** the region is not overlapped with any region in $M$ **then**
7:             $M_{P_i} \in \mathbb{N} \leftarrow L_j$
8:         **end if**
9:     **end for**
10:     **return** $M$.
11: **end procedure**

---

### A.2.2 Sparse Mixture of Convolution Experts

The pseudocode in 2 outlines the selective activation of convolutions at positions determined by Non-Maximum Suppression, using corresponding kernel sizes.

The input consists of: input (embeddings of nucleotides), positions (at a nucleotide level) of selected basic units with their corresponding kernel sizes. The output is the embeddings of the selected basic units.

### A.2.3 Deformable Convolution

The pseudocode in 3 details how deformable convolution dynamically adjusts based on input features by modifying its parameters for each input segment.

The input consists of: input (embeddings of selected basic units). The output is the embeddings of the final tokens.

---

**Algorithm 2** Detailed Sparse Convolution

---

1: **procedure** SPARSE CONVOLUTION(input $X \in \mathbb{R}^{l \times d}$, selected positions with kernel sizes $M \in \mathbb{N}^l$)
2:     $k \leftarrow$ number of non-zero elements in $M$.
3:     Initialize an output array with zeros $U \in \mathbb{R}^{k \times d}$.
4:     Initialize counter $cnt = 0$.
5:     **for** each $i$ in $[1, 2, \ldots, l]$ **do**
6:         **if** $M_i \neq 0$ **then**
7:             cnt $\leftarrow$ cnt + 1.
8:             Extract the segment of $X$ centered at $i$ with width $M_i$.
9:             $U_{cnt} \in \mathbb{R}^d \leftarrow$ Apply the convolution expert of kernel size $M_i$ to the segment.
10:         **end if**
11:     **end for**
12:     **return** $U$.
13: **end procedure**

---

---

**Algorithm 3** Detailed Deformable Convolution

---

1: **procedure** DEFORMABLE CONVOLUTION(input $U \in \mathbb{R}^{k \times d}$)
2:     Initialize an output array with zeros $T \in \mathbb{R}^{k \times d}$.
3:     **for** each $i$ in $[1, 2, \ldots, k]$ **do**
4:         Calculate offsets $\Delta P_i \in \mathbb{R}^f$ based on $U_i$.
5:         Calculate modulation factors $\Delta M_i \in \mathbb{R}^f$ based on $U_i$.
6:         Extract the deformed segment of $U$ centred at $i$ according to $\Delta P_i$.
7:         Weight the segment by $\Delta M_i$.
8:         $T_i \in \mathbb{R}^d \leftarrow$ Compute the segment's dot product with the convolution kernel of size $f$.
9:     **end for**
10:     **return** $T$.
11: **end procedure**

---

### A.3 Back Propagation

The tokenization module is learnt solely through gradient descent. In this section, we will focus on the gradients with respect to the Basic Units Scoring Block and Distal Relation Estimation Block.

#### A.3.1 Sparse Mixture of Convolution Experts

Recall the forward process in Eq. 1:

$$
\begin{aligned}
U_i &= E_j \left( X_{\left[ i - \left\lceil \frac{M_i}{2} \right\rceil + 1 : i + \left\lfloor \frac{M_i}{2} \right\rfloor \right]} \right) \cdot \text{softmax} \left( S_i \right)_j \cdot \mathbb{1} \left( M_i > 0 \right), \text{where } L_j = M_i \\
\mathbb{1}(M_i > 0) &= \begin{cases} 1, M_i > 0 \\ 0, M_i \leq 0 \end{cases}, \text{softmax}(S_i)_j = \frac{e^{S_{ij}}}{\sum_k e^{S_{ik}}},
\end{aligned}
\tag{5}
$$

The gradient w.r.t the score $S_{ik}$ is as follows:

$$
\begin{aligned}
\frac{\partial U_i}{\partial S_{ik}} &= E_j \left( X_{\left[ i - \left\lceil \frac{M_i}{2} \right\rceil + 1 : i + \left\lfloor \frac{M_i}{2} \right\rfloor \right]} \right) \cdot \mathbb{1}(M_i > 0) \cdot \frac{\partial \text{softmax}(S_i)_j}{\partial S_{ik}}, \text{where } L_j = M_i \\
\frac{\partial \text{softmax}(S_i)_j}{\partial S_{ik}} &= \begin{cases} \text{softmax}(S_i)_j \cdot (1 - \text{softmax}(S_i)_j) &, k = j \\ -\text{softmax}(S_i)_j \cdot \text{softmax}(S_i)_k &, k \neq j \end{cases}
\end{aligned}
\tag{6}
$$

#### A.3.2 Deformable Convolution

Recall the forward process in Eq. 2:

$$T_i = \sum_{p \in \left\{ -\left\lceil \frac{f}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{f}{2} \right\rfloor \right\}} w_p \cdot U_{i+p+\Delta p} \cdot \Delta m \tag{7}$$

$$U_{p'} = \sum_{q \in \{1, \ldots, k\}} \max\left(0, 1 - |p' - q|\right) \cdot U_q \tag{8}$$

The gradients w.r.t the offset $\Delta p$ and the modulation $\Delta m$ are as follows:

$$\frac{\partial T_i}{\partial \Delta p} = \sum_{p \in \left\{ -\left\lceil \frac{f}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{f}{2} \right\rfloor \right\}} w_p \cdot \Delta m \cdot \sum_{q \in \{1, \ldots, k\}} \frac{\partial \max\left(0, 1 - |i + p + \Delta p - q|\right)}{\Delta p} \cdot U_q \tag{9}$$

$$\frac{\partial T_i}{\partial \Delta m} = \sum_{p \in \left\{ -\left\lceil \frac{f}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{f}{2} \right\rfloor \right\}} w_p \cdot U_{i+p+\Delta p} \cdot \frac{\partial \Delta m}{\partial \Delta m} = \sum_{p \in \left\{ -\left\lceil \frac{f}{2} \right\rceil + 1, \ldots, \left\lfloor \frac{f}{2} \right\rfloor \right\}} w_p \cdot U_{i+p+\Delta p} \tag{10}$$

## A.4   Experiment Setting Details

### A.4.1   Settings

**Model Implementation**   MxDNA is built on the Nucleotide Transformer V2 architecture which incorporates several architectural improvements recognized in the NLP community, such as rotary positional encodings [49], SwishGLU MLP [50], and the exclusion of linear bias terms [3, 51]. Consistent with Nucleotide Transformer V2 100M, MxDNA has 512 hidden units, an expansion factor of 4, 16 attention heads, and 22 layers, totaling approximately 100M parameters. Specifically, the model's learnt tokenization module includes 10 convolution experts with kernel sizes ranging from 1 to 10, along with a deformable convolution block with a kernel size of three. We integrate this module by replacing the fifth transformer block, aiming to avoid introducing additional computations. We utilize FlashAttention [52, 53] for efficient attention calculations.

**Pretraining**   Following [4], MxDNA is pretrained on the whole Human Reference Genome [32] using Masked Language Modeling. We removed all sequences gaps and unannotated regions and extracted 70 to 510-nt-long sequences as training data. We mask 15% of the tokens, with 80% replaced by a special [MASK] token, 10% replaced with a random vocabulary token, and 10% left unchanged. All masking happens at the initial input stage(single nucleotide, 6mer tokens, bpe tokens). For model using single nucleotide tokenization, non-overlapping 6mer and BPE, the masking is performed randomly and mask out 15% of total tokens except of special tokens. For model using overlapping 6mer, we follow the strategy used in [4], with contiguous k-length spans of certain k-mers are masked, totalling around 15% of the tokens. An auxiliary balancing loss with a weight of 0.01 is used to prevent degradation towards a single expert, following [16]. The model is trained with a learning rate of 1e-4 and a batch size of 512. We employ the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.98$, $\epsilon = 1e-6$, a weight decay of 0.01, and a cosine annealing learning rate scheduler with a linear warm-up over the first 10% of steps. The model undergoes training for 500k steps for main performance comparisons and 100k steps for ablations.

**Downstream**   We download the data from `https://huggingface.co/spaces/InstaDeepAI/` `nucleotide_transformer_benchmark` for Nucleotide Transformer Benchmarks and `https://` `github.com/ML-Bioinfo-CEITEC/genomic_benchmarks` for Genomic Benchmarks. Moreover, in Nucleotide Transformer Benchmarks, the BERT-like models are finetuned using PEFT (parameter efficient finetuning) without providing the exact hyperparameters. Believing that fully fine-tuning these models will better leverage their capabilities and provide a fairer comparison , we decide to proceed with full finetuning for all models. We keep the original data splits in [6, 20]. We do not perform cross validation as [23] does since it will be too computationally expensive for BERT-like models and we decide to follow the practice of HyenaDNA [7] instead. Additionally, we repeat all experiments under three random seeds, report the average results with sample standard deviations.

All the BERT-like models are fully finetuned with a batch size of 32 and a learning rate of 3e-5. We employ the AdamW optimizer with $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 1e-8$, and a weight decay of 0.01.

Models are trained for 10 epochs on Genomic Benchmarks and 20 epochs on Nucleotide Transformer Benchmarks, with the learning rate linearly warmed up over the first epoch and then decaying to zero during the remaining epochs. For the Mouse Enhancers dataset (sequence lengths with mean = 2381, std = 984.4, max = 4707), we truncate the sequence to a maximum length of 4096, which is considered acceptable. For DNABERT, which can not handle sequences of length over 512, we truncate the sequence to a maximum length of 512.

For HyenaDNA, we fully finetune the pretrained model from `https://huggingface.co/LongSafari/hyenadna-tiny-1k-seqlen-d256` using the hyperparameters provided by [7] in docker image `hyenadna/hyena-dna-nt6:latest` for Nucleotide Transformer Benchmarks, and `https://huggingface.co/LongSafari/hyenadna-tiny-1k-seqlen` with modified hyperparameters recommended by [23] for Genomic Benchmarks. Their research suggests that training with sequence lengths 2 to 4 times the length of sequences used in downstream tasks typically yields the best performance. Thus, the tiny models are the best choice for most of the downstream tasks in Nucleotide Transformer Benchmarks and Genomic Benchmarks since most of tasks have sequence length of around a few hundreds and the tiny model are pretrained with 1000 length sequence. Notice that although our reproduced results is a bit lower than the results reported by the authors of HyenaDNA, the performance of MxDNA is still better than originally reported results on most of the tasks.

### A.4.2 Metrics

This section defines the metrics used to evaluate the performance of models on various genomic tasks. On Nucleotide Transformer Benchmarks, We used the Matthews Correlation Coefficient (MCC) for histone marker tasks, F1 scores for regulatory and splice site annotation tasks, except accuracy for splice site all task. Top-1 Accuracy is used for all tasks in Genomics Benchmarks.

**Matthews Correlation Coefficient (MCC)** The Matthews Correlation Coefficient is a robust statistical rate which takes into account true and false positives and negatives and is generally regarded as a balanced measure that can be used even if the classes are of very different sizes.

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

where $TP, TN, FP$, and $FN$ are the numbers of true positives, true negatives, false positives, and false negatives, respectively.

**F1 Score** We use the macro-averaged F1 score, which is computed using the arithmetic mean of all the per-class F1 scores. The (per-class) F1 score is the harmonic mean of precision and recall and is particularly useful when the costs of false positives and false negatives are high.

$$\text{(per-class) F1} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}}$$

where precision $= \frac{TP}{TP+FP}$ and recall $= \frac{TP}{TP+FN}$.

**Accuracy** Accuracy is the proportion of true results (both true positives and true negatives) among the total number of cases examined.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

### A.5 Ablation Results Details

### A.5.1 Different Tokenization Methods

Detailed results on each datasets with different tokenization methods are presented in Table 6 and Table 7. "1-mer" stands for single nucleotide. "Ovlp 6-mer" stands for overlapping 6-mer. "Non-Ovlp" stands for non-overlapping 6-mer. "BPE" stands for Byte-pair Encoding with a vocabulary

Table 6: Genomic Benchmarks. Different tokenization methods.

| Dataset | 1-mer | Ovlp 6-mer | Non-Ovlp 6-mer | BPE | MxDNA |
|---|---|---|---|---|---|
| Average | $88.56 \pm 0.02$ | $88.55 \pm 0.07$ | $86.83 \pm 0.06$ | $87.30 \pm 0.16$ | $88.89 \pm 0.05$ |
| Mouse Enhancers | $77.56 \pm 0.94$ | $78.81 \pm 0.85$ | $82.43 \pm 0.90$ | $80.44 \pm 1.44$ | $80.44 \pm 0.63$ |
| Coding vs Intergenomic | $95.05 \pm 0.13$ | $94.97 \pm 0.06$ | $92.73 \pm 0.08$ | $92.25 \pm 0.11$ | $94.78 \pm 0.04$ |
| Human vs Worm | $97.52 \pm 0.04$ | $97.14 \pm 0.11$ | $96.35 \pm 0.04$ | $96.59 \pm 0.01$ | $97.27 \pm 0.07$ |
| Human Enhancer Cohn | $73.70 \pm 0.53$ | $73.07 \pm 0.56$ | $72.72 \pm 0.19$ | $72.92 \pm 0.23$ | $73.98 \pm 0.40$ |
| Human Enhancer Ensembl | $92.79 \pm 0.09$ | $92.98 \pm 0.07$ | $91.79 \pm 0.23$ | $92.38 \pm 0.08$ | $92.73 \pm 0.08$ |
| Human Regulatory | $94.03 \pm 0.02$ | $94.21 \pm 0.08$ | $93.73 \pm 0.08$ | $89.70 \pm 0.10$ | $94.10 \pm 0.12$ |
| Human OCR Ensembl | $80.84 \pm 0.50$ | $81.36 \pm 1.05$ | $75.64 \pm 0.64$ | $77.87 \pm 0.45$ | $80.62 \pm 0.42$ |
| Human NonTATA Promoters | $97.00 \pm 0.05$ | $95.84 \pm 0.69$ | $89.24 \pm 0.24$ | $96.22 \pm 0.17$ | $97.22 \pm 0.23$ |

Table 7: Nucleotide Transformer Benchmarks. Different tokenization methods.

| Dataset | 1-mer | Ovlp 6-mer | Non-Ovlp 6-mer | BPE | MxDNA |
|---|---|---|---|---|---|
| Average | $75.07 \pm 0.26$ | $74.35 \pm 0.35$ | $67.65 \pm 0.17$ | $74.96 \pm 0.16$ | $77.52 \pm 0.18$ |
| Histone Markers Avg. | $63.13 \pm 0.34$ | $61.88 \pm 0.66$ | $50.36 \pm 0.28$ | $64.58 \pm 0.13$ | $67.29 \pm 0.23$ |
| H3 | $80.92 \pm 0.85$ | $80.94 \pm 0.64$ | $74.77 \pm 0.32$ | $80.26 \pm 0.15$ | $82.14 \pm 0.76$ |
| H3K14ac | $62.00 \pm 1.79$ | $62.17 \pm 0.80$ | $46.33 \pm 0.72$ | $65.91 \pm 0.72$ | $68.29 \pm 0.65$ |
| H3K36me3 | $62.59 \pm 1.50$ | $63.26 \pm 1.60$ | $50.49 \pm 1.72$ | $63.83 \pm 0.73$ | $65.46 \pm 1.74$ |
| H3K4me1 | $51.66 \pm 0.57$ | $50.11 \pm 3.63$ | $41.26 \pm 1.05$ | $54.33 \pm 0.53$ | $54.97 \pm 1.50$ |
| H3K4me2 | $49.51 \pm 0.95$ | $35.59 \pm 5.72$ | $29.99 \pm 0.93$ | $49.48 \pm 0.74$ | $55.30 \pm 0.49$ |
| H3K4me3 | $54.14 \pm 0.95$ | $54.15 \pm 0.56$ | $30.83 \pm 0.59$ | $58.10 \pm 0.55$ | $63.82 \pm 0.92$ |
| H3K79me3 | $70.36 \pm 1.58$ | $71.30 \pm 0.04$ | $59.99 \pm 0.45$ | $70.89 \pm 0.74$ | $73.74 \pm 0.78$ |
| H3K9ac | $60.63 \pm 2.73$ | $64.70 \pm 0.32$ | $50.24 \pm 1.31$ | $62.22 \pm 0.54$ | $63.15 \pm 0.26$ |
| H4 | $80.23 \pm 0.79$ | $80.09 \pm 0.59$ | $78.27 \pm 0.60$ | $79.29 \pm 0.41$ | $80.89 \pm 0.23$ |
| H4ac | $59.25 \pm 1.36$ | $56.49 \pm 1.02$ | $41.42 \pm 0.89$ | $61.45 \pm 0.80$ | $65.14 \pm 0.23$ |
| Regulatory Annotation Avg. | $85.38 \pm 0.15$ | $84.95 \pm 0.12$ | $84.13 \pm 0.09$ | $84.62 \pm 0.32$ | $85.70 \pm 0.21$ |
| Enhancer | $78.93 \pm 0.70$ | $76.65 \pm 0.80$ | $78.48 \pm 0.88$ | $77.17 \pm 0.80$ | $79.73 \pm 0.42$ |
| Enhancer Types | $58.90 \pm 0.72$ | $59.12 \pm 0.13$ | $56.50 \pm 0.61$ | $59.57 \pm 0.39$ | $59.79 \pm 0.52$ |
| Promoter All | $96.92 \pm 0.08$ | $96.94 \pm 0.12$ | $95.76 \pm 0.10$ | $95.77 \pm 0.09$ | $96.87 \pm 0.10$ |
| Promoter Non-TATA | $97.04 \pm 0.05$ | $96.84 \pm 0.07$ | $95.87 \pm 0.08$ | $95.83 \pm 0.16$ | $96.81 \pm 0.13$ |
| Promoter TATA | $95.09 \pm 0.34$ | $95.21 \pm 0.18$ | $94.02 \pm 0.01$ | $94.77 \pm 0.33$ | $95.32 \pm 0.17$ |
| Splice Site Annotation Avg. | $97.70 \pm 0.19$ | $98.25 \pm 0.06$ | $97.85 \pm 0.08$ | $93.46 \pm 0.02$ | $98.00 \pm 0.13$ |
| All | $98.07 \pm 0.17$ | $98.18 \pm 0.11$ | $97.91 \pm 0.15$ | $93.25 \pm 0.41$ | $98.05 \pm 0.13$ |
| Accpetor | $97.61 \pm 0.44$ | $98.18 \pm 0.15$ | $97.65 \pm 0.15$ | $94.40 \pm 0.34$ | $97.68 \pm 0.20$ |
| Donor | $97.42 \pm 0.58$ | $98.39 \pm 0.16$ | $98.00 \pm 0.16$ | $92.72 \pm 0.42$ | $98.28 \pm 0.07$ |

Table 8: Genomic Benchmarks. Different components.

| Dataset | 1-mer | + MoCE | + Def Conv | + Noise (MxDNA) |
|---|---|---|---|---|
| Average | $88.56 \pm 0.02$ | $88.72 \pm 0.07$ | $88.86 \pm 0.18$ | $88.89 \pm 0.05$ |
| Mouse Enhancers | $77.56 \pm 0.94$ | $79.66 \pm 0.79$ | $80.36 \pm 0.39$ | $80.44 \pm 0.63$ |
| Coding vs Intergenomic | $95.05 \pm 0.13$ | $94.64 \pm 0.10$ | $94.57 \pm 0.43$ | $94.78 \pm 0.04$ |
| Human vs Worm | $97.52 \pm 0.04$ | $97.20 \pm 0.03$ | $97.31 \pm 0.05$ | $97.27 \pm 0.07$ |
| Human Enhancer Cohn | $73.70 \pm 0.53$ | $74.78 \pm 0.52$ | $73.75 \pm 1.10$ | $73.98 \pm 0.40$ |
| Human Enhancer Ensembl | $92.79 \pm 0.09$ | $92.59 \pm 0.19$ | $92.78 \pm 0.19$ | $92.73 \pm 0.08$ |
| Human Regulatory | $94.03 \pm 0.02$ | $93.93 \pm 0.04$ | $94.15 \pm 0.11$ | $94.10 \pm 0.12$ |
| Human OCR Ensembl | $80.84 \pm 0.50$ | $80.27 \pm 0.34$ | $81.14 \pm 0.08$ | $80.62 \pm 0.42$ |
| Human NonTATA Promoters | $97.00 \pm 0.05$ | $96.92 \pm 0.32$ | $96.81 \pm 0.26$ | $97.22 \pm 0.23$ |

Table 9: Nucleotide Transformer Benchmarks. Different components.

| Dataset | 1-mer | + MoCE | + Def Conv | + Noise (MxDNA) |
|---|---|---|---|---|
| Average | $75.07_{\pm 0.26}$ | $77.00_{\pm 0.05}$ | $77.35_{\pm 0.12}$ | $77.52_{\pm 0.18}$ |
| Histone Markers Avg. | $63.13_{\pm 0.34}$ | $66.58_{\pm 0.11}$ | $67.02_{\pm 0.10}$ | $67.29_{\pm 0.23}$ |
| H3 | $80.92_{\pm 0.85}$ | $81.18_{\pm 0.52}$ | $81.65_{\pm 0.82}$ | $82.14_{\pm 0.76}$ |
| H3K14ac | $62.00_{\pm 1.79}$ | $67.68_{\pm 0.44}$ | $66.12_{\pm 0.46}$ | $68.29_{\pm 0.65}$ |
| H3K36me3 | $62.59_{\pm 1.50}$ | $66.51_{\pm 0.46}$ | $65.26_{\pm 1.33}$ | $65.46_{\pm 1.74}$ |
| H3K4me1 | $51.66_{\pm 0.57}$ | $53.18_{\pm 2.31}$ | $56.14_{\pm 1.04}$ | $54.97_{\pm 1.50}$ |
| H3K4me2 | $49.51_{\pm 0.95}$ | $53.86_{\pm 4.01}$ | $54.13_{\pm 1.22}$ | $55.30_{\pm 0.49}$ |
| H3K4me3 | $54.14_{\pm 0.95}$ | $63.82_{\pm 1.72}$ | $61.42_{\pm 1.58}$ | $63.82_{\pm 0.92}$ |
| H3K79me3 | $70.36_{\pm 1.58}$ | $72.72_{\pm 0.71}$ | $72.88_{\pm 0.75}$ | $73.74_{\pm 0.78}$ |
| H3K9ac | $60.63_{\pm 2.73}$ | $63.78_{\pm 0.43}$ | $64.95_{\pm 1.57}$ | $63.15_{\pm 0.26}$ |
| H4 | $80.23_{\pm 0.79}$ | $79.96_{\pm 0.42}$ | $81.47_{\pm 0.97}$ | $80.89_{\pm 0.23}$ |
| H4ac | $59.25_{\pm 1.36}$ | $64.09_{\pm 0.73}$ | $66.21_{\pm 0.41}$ | $65.14_{\pm 0.23}$ |
| Regulatory Annotation Avg. | $85.38_{\pm 0.15}$ | $85.60_{\pm 0.32}$ | $85.71_{\pm 0.29}$ | $85.70_{\pm 0.21}$ |
| Enhancer | $78.93_{\pm 0.70}$ | $79.07_{\pm 0.29}$ | $78.91_{\pm 0.29}$ | $79.73_{\pm 0.42}$ |
| Enhancer Types | $58.90_{\pm 0.72}$ | $60.07_{\pm 0.82}$ | $60.28_{\pm 1.20}$ | $59.79_{\pm 0.52}$ |
| Promoter All | $96.92_{\pm 0.08}$ | $96.92_{\pm 0.17}$ | $96.80_{\pm 0.03}$ | $96.87_{\pm 0.10}$ |
| Promoter Non-TATA | $97.04_{\pm 0.05}$ | $96.71_{\pm 0.13}$ | $96.96_{\pm 0.11}$ | $96.81_{\pm 0.13}$ |
| Promoter TATA | $95.09_{\pm 0.34}$ | $95.25_{\pm 0.34}$ | $95.69_{\pm 0.25}$ | $95.32_{\pm 0.17}$ |
| Splice Site Annotation Avg. | $97.70_{\pm 0.19}$ | $97.39_{\pm 0.14}$ | $97.86_{\pm 0.11}$ | $98.00_{\pm 0.13}$ |
| All | $98.07_{\pm 0.17}$ | $98.20_{\pm 0.03}$ | $98.13_{\pm 0.09}$ | $98.05_{\pm 0.13}$ |
| Accpetor | $97.61_{\pm 0.44}$ | $97.08_{\pm 0.13}$ | $97.76_{\pm 0.21}$ | $97.68_{\pm 0.20}$ |
| Donor | $97.42_{\pm 0.58}$ | $96.90_{\pm 0.52}$ | $97.69_{\pm 0.50}$ | $98.28_{\pm 0.07}$ |

size of 4096 borrowed from DNABERT2. All models are trained for 100k steps with same backbone but different tokenization methods.

### A.5.2 Different Components

Detailed results on each datasets with components added from the single nucleotide tokenization baseline are presented in Table 8 and Table 9. "1-mer" stands for the baseline. "+ MoCE" stands for adding the sparse Mixture of Convolution Experts. "+ Def Conv" stands for adding the deformable convolution block. "+ Noise" stands for adding the multiplicative jitter noise. These components are added successively, finally equivalent to MxDNA. All models are trained for 100k steps with same backbone and components added sequentially.

### A.6 Visualization Details

The BPE tokenizer used here is directly borrowed from DNABERT2 with a vocabulary of size 4096. The visualization methods for traditional tokenization methods is straightforward. Below are the visualization details for MxDNA.

**Sample Level:** For MxDNA, we perform a forward process of the model using the sequence as input, extracting the Mask of basic units existence $M$, Offsets $\Delta P$ and Modulation factors $\Delta M$ of the deformable convolution. First, we colour the recognized basic units based on $M$. Then, we determine the distal relations using $\Delta P$ and $\Delta M$. Specifically, a distal relation is considered to be visualized only if the the product of the corresponding modulation weight and the bilinearly interpolated offset weight exceeds one. Eventually, a final learnt token is made up of a group of related basic units and coloured by the colour of the central basic unit.

**Dataset Level:** For dataset level, we first finetune the MxDNA model on downstream datasets and use the refined models to generate the Mask of basic units existence $M$ for all samples in the dataset. We then calculate the proportion of the lengths of each recognized basic units as indicated by $M$. Specifically, we finetune MxDNA on H3, Enhancer, Promoter All and Splice Site All in Nucleotide Transformer Benchmarks for Histone Marker, Enhancer, Promoter and Splice Site respectively.

Table 10: Comparison of various models based on their computational complexity.

| Model | Flops (G) | Macs (G) | Parameters (M) | Number of Tokens |
|---|---|---|---|---|
| DNABERT2 | 24.80 | 12.39 | 117.07 | 104.2 |
| NTv2 100M | 16.63 | 8.31 | 97.89 | 86 |
| DNABERT | 99.48 | 49.70 | 89.20 | 507 |
| HyenaDNA tiny d256 | 1.67 | 0.832 | 1.64 | 511 |
| HyenaDNA tiny | 0.441 | 0.219 | 0.436 | 511 |
| MxDNA | 35.94 | 17.93 | 100.09 | $512 \rightarrow 101.6$ |
| Learnt Tokenization Module | 0.914 | 0.446 | 11.69 | $512 \rightarrow 101.6$ |
| Single Nucleotide Baseline | 94.85 | 47.38 | 92.95 | 512 |

Table 11: Assets used in this work

| Asset | License |
|---|---|
| GRCh38 [32] | CC BY 4.0 |
| Genomic Benchmarks [20] | Apache-2.0 |
| Nucleotide Transformer [6] | CC BY-NC-SA 4.0 |
| DNABERT [4] | Apache-2.0 |
| DNABERT2 [5] | Apache-2.0 |
| HyenaDNA [7] | Apache-2.0 |
| FlashAttention [52, 53] | BSD-3-Clause |
| Pytorch [54] | BSD-3-Clause |
| Pytorch Lightning [55] | Apache-2.0 |
| Huggingface [56] | Apache-2.0 |
| Pybind11 [57] | BSD-3-Clause |
| Scikit-Learn [58] | BSD-3-Clause |
| Numpy [59] | BSD-3-Clause |
| Matplotlib [60] | Matplotlib License |
| Seaborn [61] | Apache-2.0 |

**Token Embedding Analysis:** For the token embedding analysis, we utilize various pretrained models to embed sequences with different functions and analyse the output embeddings at the token level. Initially, we perform principal component analysis to reduce the dimensionality to one hundred, which facilitates the visualization process. Subsequently, we employ t-SNE to visualize these token embeddings in a two-dimensional space. Specifically, we use the H3, Enhancer, Promoter All, and Splice Site All sequences from the test set of the Nucleotide Transformer Benchmarks to represent Histone Marker, Enhancer, Promoter, and Splice Site, respectively.

## A.7 Computational Resources

We train and evaluate the models on NVIDIA RTX 3090 and NVIDIA A100 GPUs. The pretraining of MxDNA takes around 3 days for 500k steps using 4 A100 GPUs. Finetuing MxDNA on all the downstream tasks takes approximately 1.5 Days using 1 A100 GPU. This is true for other BERT-like foundation models with around 100M parameters.

The detailed computational costs of the models (averaged across 5 samples of sequence length of 510) are outlined in Table 10. The integration of a mixture of convolution experts and the deformable convolution introduces an increased computational overhead initially due to the $O(l \log(l))$ time complexity of the learned tokenization mechanism (where $l$ represents the number of nucleotides). This complexity is mitigated by the substantial reduction in sequence length after tokenization, which decreases the number of tokens processed by subsequent transformer layers.

## A.8 Assets

The assets used in this work along with their licenses including data, pretrained weights, benchmarks, libraries, and software are presented in Table 11.

# NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: As pointed out in Sec. 1, We propose a method where the model autonomously learns to tokenize DNA and it achieves great performance.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Limitations are discussed in Sec. 5.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: The paper is generally empirical, not theoretical.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: The detailed experiment settings are provided in Appx. A.4. The results can be reproduced following these settings.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

24

Answer: [Yes]

Justification: Our code will be make public in the github repository once ready. We use publicly available data for our experiments as illustrated in Appx. A.4.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source Benchmarks).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [Yes]

Justification: The detailed experiment settings are provided in Appx. A.4.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: We finetune all the models under three random seeds and report the average results with sample standard deviations.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).

- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: The experiments compute resources are given in Appx. A.7.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

Answer: [Yes]

Justification: We review and follow the NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The paper focuses on tokenization techniques of DNA foundation models. While there maybe misuse of DNA foundation models, our learnt tokenization method will neither increase nor decrease the risk of the potential misuse.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: Our paper poses no such risk.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: The assets are summarized in Appx. A.8 and cited throughout the paper.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: We do not release new assets for now since it is not ready.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: We do not involve crowdsourcing nor research with human subjects.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.