

# Exploring the Embedding Methods in Genomic Language Models

*Anisa Habib*  
*University of Utah*

UUCS-24-005

School of Computing  
University of Utah  
Salt Lake City, UT 84112 USA

26 April 2024

## *Abstract*

Language Models (LMs) have revolutionized natural language processing, excelling in language translation, sentiment analysis, and text generation. Researchers have proposed using LMs to learn generalizable features from DNA, aiming to fine-tune these models for diverse prediction tasks. While several LMs trained on DNA sequences now exist, they vary in tokenization methods, the types and amounts of data used for training, and the specific tasks they are fine-tuned for. Existing benchmark reports often lack comprehensive coverage and consistency in reported metrics. To address this gap and explore the impact of different encoding schemes for DNA, this study conducts benchmarking tests on standard tasks to assess and compare existing models' performance capabilities. Additionally, we construct our own fine-tuning task to perform preliminary investigations on whether an LM can accurately identify the locations of prophage sequences integrated into the bacterial genome. Our findings suggest that model accuracy varies depending on the task, with no single model performing best across all tasks. We observed that tasks exhibit different levels of difficulty, and there is a wide distribution of variation in performance even with the same model.

EXPLORING THE EMBEDDING METHODS IN GENOMIC LANGUAGE MODELS

by

Anisa Habib

A Senior Honors Thesis Submitted to the Faculty of  
The University of Utah  
In Partial Fulfillment of the Requirements for the  
Degree of Bachelor of Science

In

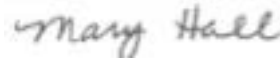
Computer Science

Approved:



---

Hari Sundar  
Thesis Faculty Supervisor



---

Mary Hall  
Director, School of Computing



---

Thomas C. Henderson  
Honors Faculty Advisor

---

Monisha Pasupathi  
Dean, Honors College

April 2024

Copyright © 2024

All Rights Reserved

## TABLE OF CONTENTS

ABSTRACT	ii
INTRODUCTION	1
BACKGROUND	2
METHODS	6
RESULTS	7
Benchmark Evaluation	7
Phage Results	9
DISCUSSION	10
REFERENCES	12
APPENDIX	14
Model Details	14
Dataset Details	15
Full Results	18

## INTRODUCTION

Language Models (LMs) have revolutionized natural language processing, excelling in language translation, sentiment analysis, and text generation. Their capacity for self-supervised learning enables their application across various domains, including biology and genetics. Early applications in biology involved training LMs on protein sequences to predict masked amino acids [1, 2]. Fine-tuning these models on a variety of protein tasks, they achieved comparable or superior results to previous methods [3, 4], even with limited data [5].

DNA, the complex molecule containing genetic instructions for all living organisms, is crucial for understanding trait inheritance and genomic processes [6]. Learning genetic concepts through the language of DNA aims to address challenges in genetic engineering and precision medicine, including disease risk and pharmacogenomics, among other applications [7]. Despite ongoing efforts, predicting genetic elements solely from DNA remains difficult due to limited annotated data. To address this challenge, researchers have proposed using LMs to learn generalizable features from DNA, aiming to fine-tune these models for diverse prediction tasks [8, 9, 10, 11].

While several LMs trained on DNA sequences now exist, they vary in tokenization methods, the types and amounts of data used for training, and the specific tasks they are fine-tuned for. The choice of tokenization significantly affects model performance and generalization to genomic tasks, making it crucial to compare these methods. However, existing benchmark reports often lack comprehensive coverage and consistency in reported metrics. There is currently no unified platform for comparing all genomic models across all existing benchmarks. To address this gap and explore the impact of different encoding schemes for DNA,

this study conducts benchmarking tests on standard tasks to assess and compare existing models' performance capabilities.

Additionally, we fine-tuned these models on the binary classification task of identifying a sequence as phage or bacteria. Bacteriophages, or phages, are viruses that infect and replicate within bacteria. Phages play a key role in bacterial population dynamics, nutrient cycling, and even human health through their impact on the microbiome. They are also of interest in biotechnology and medicine for their potential use in phage therapy to combat bacterial infections [12]. By fine-tuning existing LMs on bacterial and phage genomic sequences, we aim to perform preliminary investigations on whether a LM can accurately identify the locations of prophage sequences integrated into the bacterial genome.

## BACKGROUND

Tokenization is the process of breaking text into smaller units called tokens, which can range from individual characters to complete words. This process aids machines in analyzing and understanding text by identifying patterns and contextual cues. While tokenizing natural language text is typically straightforward, often relying on white spaces or characters, the process becomes considerably more challenging when applied to DNA sequences. This complexity arises due to their unique four-letter alphabet (A, C, G, T) and extensive lengths, which can make traditional tokenization methods computationally demanding.

Several LMs have been trained using DNA as input, including the Nucleotide Transformer [8], DNABERT [7, 9], and HyenaDNA [10]. In addition to tokenization, these models all vary in their parameters, architectures, training and evaluation methods. A summary is shown in Table 1. As a baseline, we include the English language model GPT-2 [13].

Table 1. Tested Language Models.

Pre-training time is indicated for the smallest model size where relevant.

Model	Authors	Parameters	Tokenization	Benchmark	Context Window	PreTraining Data	PreTraining Time	PreTraining Hardware
GPT-2	Radford et al	124M-1.5B	BPE		1024	WebText	4 days	256 Tesla P100
DNABERT-1	Zhou et al	117M	kmer	GUE	512	Human Genome	25 days	8 Nvidia 2080Ti
DNABERT-2	Zhou et al	117M	BPE	GUE	1000	Human Genome + 135 Other Species	14 days	8 Nvidia 2080Ti
HyenaDNA	Ngyuen et al	0.5M-6.6M	Nucleotide	Genomic Benchmark	1000-1M	Human Genome	80 min	1 Nvidia A100
Nucleotide Transformer	Dalla-Torre et al	500M-2.5B	kmer	Nucleotide Transformer	1000	3202 Human Genomes + 850 Other Species	14 days	128 Nvidia A100

Nucleotide Transformer (NT) and DNABERT-1 tokenize DNA sequences into overlapping  $k$ -mers, capturing contextual information by combining each deoxynucleotide base with its  $k$  subsequent bases. The NT models all use 6-mer tokenization, where  $k = 6$ . The DNABERT team released models experimenting with different  $k$ -mers, the 6-mer model performing the best overall [9]. While  $k$ -mer tokenization preserves local sequence patterns, it involves repeated information and demands high computational requirements.

In contrast, DNABERT-2 employs Byte Pair Encoding (BPE) tokenization. This method compresses data, replacing the most frequent pair of consecutive bytes in a data stream with a single byte value [14]. This allows DNABERT-2 to handle longer sequences within the same context window as DNABERT-1. It achieved better results than its predecessor and comparable results to NT models while having fewer parameters and less GPU time for pre-training [11]. However, its improvements cannot solely be attributed to BPE; in addition to efficiency enhancements like flash attention and Low-Rank adaptation, DNABERT-2 was trained on a much larger multi-species dataset.

HyenaDNA, based on the Hyena [15] model, uses a convolutional neural network instead of attention layers, significantly reducing computational complexity. It uses a nucleotide tokenization approach, treating each DNA base as an individual token. HyenaDNA is a long-range genomic foundation model, with context length of up to 1 million tokens at a single nucleotide resolution. It reportedly achieved state of the art results on 19 downstream tasks while using magnitudes of less parameters, training data, and pre-training time

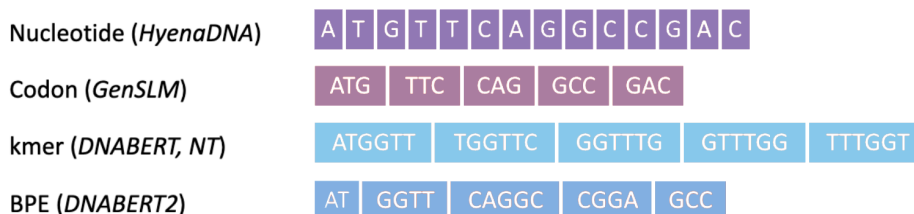


Figure 1. DNA Tokenization Methods.

Figure 1 illustrates all of the described DNA tokenization methods. Certain protein language models like GenSLM [16] employ codon tokenization, where a codon represents the three-nucleotide sequences that encode specific amino acids during protein synthesis. However, as our investigation focuses on finding regulatory sequences that occur outside of the coding region, we do not include a model for codon tokenization in our experiments.

Different tokenization methods impact the total information captured in a context window and each models’ performance, efficiency, and generalization to genomic tasks. This performance can be measured through fine-tuning. Several benchmarks for genomic tasks have been introduced: the Nucleotide Transformer [8] benchmark, the Genomic Understanding Evaluation or GUE benchmark [11], and the Genomic Benchmark [17]. In Table 2, we categorize the tasks of these respective datasets. For more details, refer to Appendix A1.

Table 2. Task Category Summary for Tested Benchmarks.

Publication Introduced In	Benchmark	Species	Task Category	Number of Tasks	Number of Classes	Sequence Length
---------------------------	-----------	---------	---------------	-----------------	-------------------	-----------------

Nucleotide Transformer	Nucleotide Transformer	Human	Regulatory Elements	5	2-3	200-300
			Splice Site Detection	3	2-3	400-600
		Yeast	Epigenetic Marks Prediction	10	2	500
DNABERT2	GUE	Human	Regulatory Elements	6	2	70-300
			Splice Site Detection	1	3	400
			Transcription Factor Prediction	5	2	100
		Mouse	Transcription Factor Prediction	5	2	100
		Yeast	Epigenetic Marks Prediction	10	2	500
		Virus	Covid Variant Classification	1	9	1000
HyenaDNA	Genomic Benchmark	Human	Regulatory Elements	4	2-3	251-500
			OCR	1	2	315
			Coding vs. Intergenic	1	2	200
		Multi-Species	Human vs. Worm	1	2	200
		Mouse	Regulatory Elements	1	2	2381

While some models have been benchmarked against one another, these reports generally do not cover all models on all tasks and are inconsistent in the metrics chosen to report. The NT and DNABERT teams report their performance across different fine-tuning tasks using the Matthew correlation coefficient (MCC). The HyenaDNA team reports MCC for some tasks but uses accuracy and F1 score for others. F1 score is widely used in machine learning applications as it considers both precision and recall, being particularly useful when there is an uneven distribution between the positive and negative classes. However, research has shown that MCC provides a more robust performance measure in evaluating binary classifications than accuracy and F1 score as it balances true positives, true negatives, false positives, and false negatives [18].

Equation 1. Matthew Correlation Coefficient.

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



To maintain consistency with previous publications, our results are collected over accuracy, F1 score, and MCC, with a primary focus on reporting MCC.

## METHODS

We evaluated several models by running each on a set of tasks 10 times. We selected DNABERT-1 (6-mer), DNABERT-2, HyenaDNA (1kb context length), Nucleotide Transformer (500M 1000G), and GPT-2 small as a baseline. All models were retrieved using the HuggingFace ‘Transformers’ package in Python. Fine-tuning datasets were retrieved via HuggingFace and GitHub, preprocessed, and split into training, development, and test sets in an 8:1:1 ratio for consistency.

Additionally, we fine-tuned all models to classify sequences as phage or bacteria using a dataset of 33,238 bacterial assemblies and 22,026 phage genomes from NCBI and INPHRED, respectively. We also trained a BPE tokenizer on bacterial DNA sequences for the purposes of analyzing its output vocabulary.

All models were fine-tuned with the same technique and parameters provided by the authors, running on 1 NVIDIA Tesla V100 GPU on the Pittsburgh Supercomputing Center’s Bridges-2. Further information is provided in Table 3.

Table 3. Model Fine-Tuning Configurations.

	<b>DNABERT-1</b>	<b>DNABERT-2</b>	<b>GPT2</b>	<b>HyenaDNA</b>	<b>Nucleotide Transformer</b>
Layers	12	24	12	256	16
Width	768	768	768	1024	512
Parameters	117M	117M	500M	1000	500M
Epochs	3-5	3-5	3-5	100	3-5
Batch size	16-32	16-32	16-32	128-256	16-32
Learning Rate	3.00E-05	3.00E-05	3.00E-05	6.00E-04	1.00E-04

## RESULTS

### Benchmark Evaluation

In Figure 2, we present each model’s mean MCC across 10 replications for every task category, with error bars depicting one standard deviation. An MCC of +1 indicates perfect classification,  $-1$  a complete misclassification, and an MCC of 0 suggests the model’s predictions are no better than random.

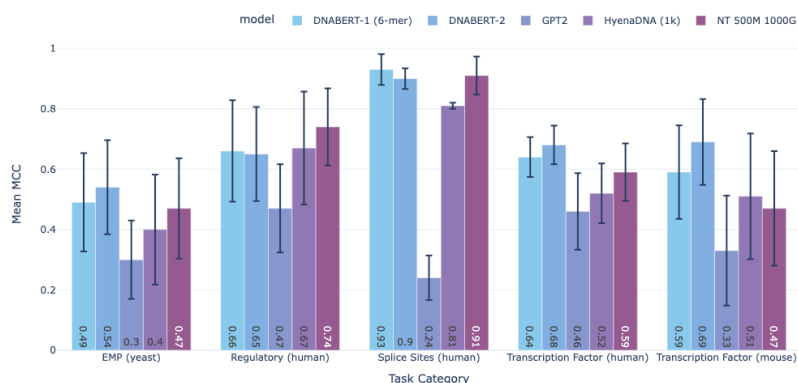


Figure 2. Model Performance Across Task Categories.

We observed a large variation across different categories, suggesting that task difficulty varies. This is particularly notable in the Epigenetic Mark Prediction tasks for yeast DNA (Figure 3). DNABERT-2 achieves an MCC of  $0.81 \pm 0.008$  predicting H4 histone marks, but  $0.31 \pm 0.03$  for H3k4me3.

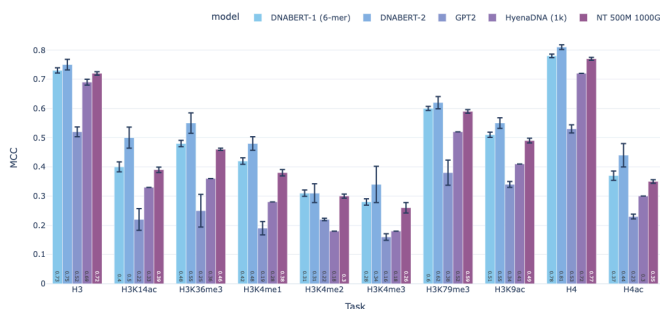


Figure 3. Model Performance Across Epigenetic Mark Prediction Tasks.

We also observed a wide distribution across replications, which varied by task. Using the H3k4me3 task as an example, DNABERT-2 observed the largest variation and HyenaDNA saw none (Figure 4). Across all tasks, HyenaDNA saw the least variation overall (Figure 5).

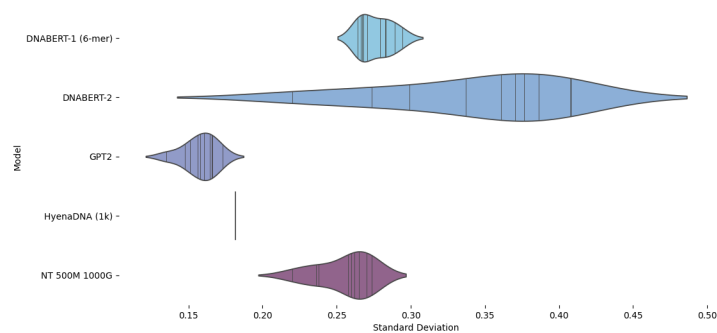


Figure 4. Model Variability in MCC for the H3k4me3 Task

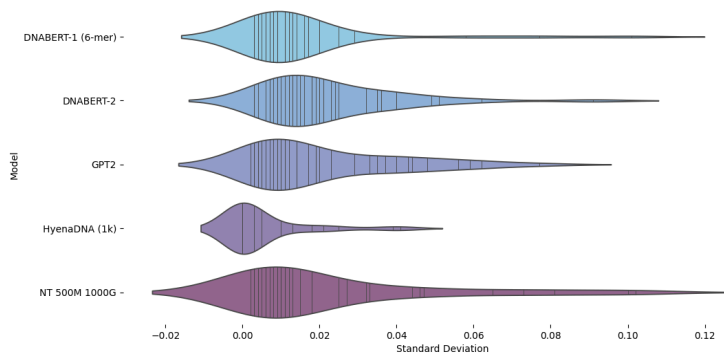


Figure 5. Model Variability in MCC for All Tasks.

No single model outperformed all others across all benchmarks. However, when considering task sequence length, certain models showed better performance on tasks with shorter or longer sequences (Figure 6).

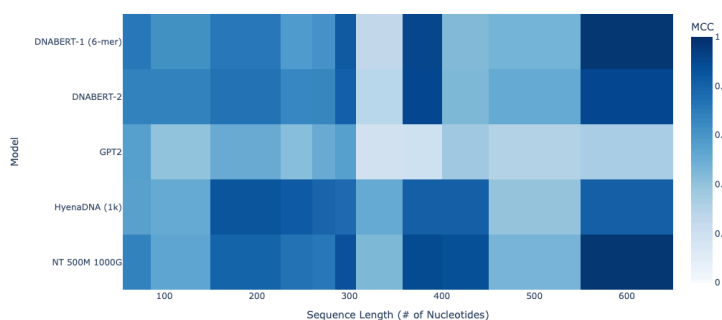


Figure 6. Model Performance by Task Sequence Length.

### Phage Results

In Figure 7, we report the results of the selected genomic models on our phage identification task. DNABERT-2 reported the best performance, with a mean MCC of 0.96.

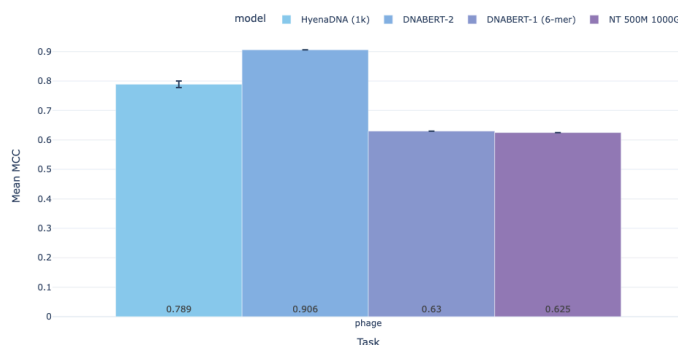


Figure 7. Model Performance on Phage Identification Task.

None of the pre-training data for the selected models included sequences from bacteria or phages. Trained on large corpora of text to select tokens according to the defined method, tokenizers compile these selected tokens in their *vocabulary*. To explore the differences in vocabulary between bacteria and mammalian DNA, we trained a BPE tokenizer on 33,238 bacterial genomes and compared its vocabulary with DNABERT-2’s multi-species vocabulary. Only 38.4% of the tokens in our bacterial BPE tokenizer overlapped with DNABERT-2 (Figure 8), indicating significant vocabulary differences.

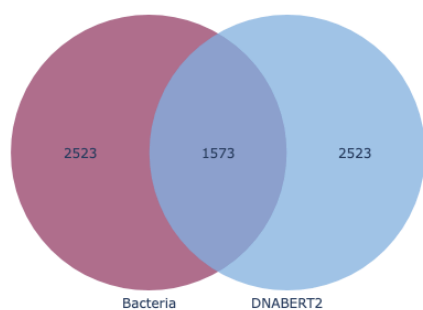


Figure 8. Vocabulary Comparison of Bacterial and DNABERT-2 BPE Tokenizers.

The vocabularies do not share many “words,” suggesting that bacterial and mammalian DNA might have unique language patterns.

## DISCUSSION

In this study, we investigated the impact of different encoding schemes for DNA sequences on the performance of genomic models across various tasks. We reviewed models with k-mer (DNABERT-1, Nucleotide Transformer), Byte Pair Encoding (DNABERT-2), and nucleotide (HyenaDNA) tokenization methods. Our findings suggest that model accuracy varies depending on the task, with no single model performing best across all tasks. We observed that tasks exhibit different levels of difficulty, and there is a wide distribution of variation in performance even with the same model. Specifically, DNABERT-2 and HyenaDNA demonstrated strong performance on tasks with longer input sequences, while DNABERT-1 and Nucleotide Transformer (NT) performed well on tasks with shorter input sequences. These results highlight the importance of considering task-specific characteristics when selecting a genomic model.

The variation in model performance across tasks suggests that more research is needed to understand why certain models excel in specific tasks. Future studies could explore training a HyenaDNA model using BPE tokenization to determine if this improves accuracy. Additionally,

training DNABERT-2 solely on the human genome could inform the extent to which its performance on different tasks is influenced by tokenization versus multi-species data.

DNABERT-2 performed best overall for the phage identification task, possibly due to its multi-species training dataset and BPE tokenization. Despite challenges in reproducing HyenaDNA results in our benchmark evaluation, it also showed promising performance on the phage identification task. Potential improvements in methodology for running the HyenaDNA model and reframing the phage identification task should be explored.

Language models trained on DNA assume a shared linguistic structure across different species, implying that fundamental aspects of DNA can be captured and utilized for various genomic tasks. Achieving a foundational model that generalizes to any genomic task and species is theoretically possible with sufficient data and appropriate encoding schemes. Current models demonstrate impressive results, however, the variability observed across replication for specific models and tasks raise concerns about reliability. Consistency in LMs is essential for ensuring predictable and reliable behavior across diverse contexts. Without consistent models, trust in their outputs diminishes, particularly in critical applications such as genetic research. Recent research indicates that smaller, task-specific models may outperform large, general-purpose models in specialized tasks [19], emphasizing the need to balance model size, training data, and task specificity in AI models for genomics. Smaller language models also offer faster results crucial for real-time applications, a reduced carbon footprint, and lower privacy risks associated with large-scale sensitive data processing. As we advance the development of genomic language models, addressing these considerations will be crucial for their successful and ethical integration into genomic research and applications.

## REFERENCES

- [1] J. Jumper *et al.*, “Highly accurate protein structure prediction with AlphaFold,” *Nature*, vol. 596, no. 7873, pp. 583–589, Aug. 2021, doi: 10.1038/s41586-021-03819-2.
- [2] A. Rives *et al.*, “Biological structure and function emerge from scaling unsupervised learning to 250 million protein sequences,” *Proc. Natl. Acad. Sci. U. S. A.*, vol. 118, no. 15, p. e2016239118, Apr. 2021, doi: 10.1073/pnas.2016239118.
- [3] M. Littmann, M. Heinzinger, C. Dallago, T. Olenyi, and B. Rost, “Embeddings from deep learning transfer GO annotations beyond homology,” *Sci. Rep.*, vol. 11, no. 1, p. 1160, Jan. 2021, doi: 10.1038/s41598-020-80786-0.
- [4] C. Marquet *et al.*, “Embeddings from protein language models predict conservation and variant effects,” *Hum. Genet.*, vol. 141, no. 10, pp. 1629–1647, Oct. 2022, doi: 10.1007/s00439-021-02411-y.
- [5] M. Littmann, M. Heinzinger, C. Dallago, K. Weissenow, and B. Rost, “Protein embeddings and deep learning predict binding residues for various ligand classes,” *Sci. Rep.*, vol. 11, no. 1, p. 23916, Dec. 2021, doi: 10.1038/s41598-021-03431-4.
- [6] “Definition of DNA - NCI Dictionary of Genetics Terms - NCI.” Accessed: Apr. 22, 2024. [Online]. Available: <https://www.cancer.gov/publications/dictionaries/genetics-dictionary/def/dna>
- [7] H. Collins, S. Calvo, K. Greenberg, L. Forman Neall, and S. Morrison, “Information Needs in the Precision Medicine Era: How Genetics Home Reference Can Help,” *Interact. J. Med. Res.*, vol. 5, no. 2, p. e13, Apr. 2016, doi: 10.2196/ijmr.5199.
- [8] Hugo Dalla-Torre *et al.*, “The Nucleotide Transformer: Building and Evaluating Robust Foundation Models for Human Genomics,” *bioRxiv*, p. 2023.01.11.523679, Jan. 2023, doi: 10.1101/2023.01.11.523679.
- [9] Y. Ji, Z. Zhou, H. Liu, and R. V. Davuluri, “DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome,” *Bioinformatics*, vol. 37, no. 15, pp. 2112–2120, Aug. 2021, doi: 10.1093/bioinformatics/btab083.
- [10] E. Nguyen *et al.*, “HyenaDNA: Long-Range Genomic Sequence Modeling at Single Nucleotide Resolution.” arXiv, Nov. 14, 2023. doi: 10.48550/arXiv.2306.15794.
- [11] Z. Zhou, Y. Ji, W. Li, P. Dutta, R. Davuluri, and H. Liu, “DNABERT-2: Efficient Foundation Model and Benchmark For Multi-Species Genome.” arXiv, Jun. 26, 2023. doi: 10.48550/arXiv.2306.15006.
- [12] S. T. Abedon, P. García, P. Mullany, and R. Aminov, “Editorial: Phage Therapy: Past, Present and Future,” *Front. Microbiol.*, vol. 8, p. 981, 2017, doi: 10.3389/fmicb.2017.00981.
- [13] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language Models are Unsupervised Multitask Learners”.
- [14] R. Sennrich, B. Haddow, and A. Birch, “Neural Machine Translation of Rare Words with Subword Units,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, K. Erk and N. A. Smith, Eds., Berlin, Germany: Association for Computational Linguistics, Aug. 2016, pp. 1715–1725. doi: 10.18653/v1/P16-1162.
- [15] M. Poli *et al.*, “Hyena Hierarchy: Towards Larger Convolutional Language Models.” arXiv, Apr. 19, 2023. doi: 10.48550/arXiv.2302.10866.
- [16] M. Zvyagin *et al.*, “GenSLMs: Genome-scale language models reveal SARS-CoV-2

- evolutionary dynamics.” bioRxiv, p. 2022.10.10.511571, Oct. 11, 2022. doi: 10.1101/2022.10.10.511571.
- [17] K. Grešová, V. Martinek, D. Čechák, P. Šimeček, and P. Alexiou, “Genomic benchmarks: a collection of datasets for genomic sequence classification,” *BMC Genomic Data*, vol. 24, no. 1, p. 25, May 2023, doi: 10.1186/s12863-023-01123-8.
- [18] D. Chicco and G. Jurman, “The advantages of the Matthews correlation coefficient (MCC) over F1 score and accuracy in binary classification evaluation,” *BMC Genomics*, vol. 21, no. 1, p. 6, Jan. 2020, doi: 10.1186/s12864-019-6413-7.
- [19] N. Mireshghallah, J. Mattern, S. Gao, R. Shokri, and T. Berg-Kirkpatrick, “Smaller Language Models are Better Zero-shot Machine-Generated Text Detectors,” in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, Y. Graham and M. Purver, Eds., St. Julian’s, Malta: Association for Computational Linguistics, Mar. 2024, pp. 278–293. Accessed: Apr. 18, 2024. [Online]. Available: <https://aclanthology.org/2024.eacl-short.25>
- [20] J. Portes *et al.*, “MosaicBERT: A Bidirectional Encoder Optimized for Fast Pretraining,” presented at the Thirty-seventh Conference on Neural Information Processing Systems, Nov. 2023. Accessed: Apr. 17, 2024. [Online]. Available: <https://openreview.net/forum?id=5zipcfLC2Z>



## APPENDIX

All code and data relevant to this project can be found at [github.com/anihab/dnaTokenization](https://github.com/anihab/dnaTokenization)

### Model Details

We selected the following models: nucleotide-transformer-500m-1000g, nucleotide-transformer-2.5b-multi-species, DNA\_bert\_6, DNABERT-2-117M, and hyendna-1k-tiny.

The Nucleotide Transformer models are pretrained on 3202 genetically diverse human genomes originating from 27 geographically structured populations of African, American, East Asian, and European ancestry taken from the 1000G project. They use 6-mer tokenization for DNA sequences, a vocabulary size of 4105, and standard BERT-style 15% Masked Language Modeling (MLM) for training. The team’s approach to tokenization involves generating all possible  $k$ -mers for nucleotides in  $O(4^k)$  time and then splitting sequences into these  $k$ -mers, along with special tokens, which is done in linear time.

DNA\_bert\_6 is another encoder-only model based on the standard BERT architecture with 12 layers and 117 million parameters. It was pretrained on the human reference genome Hg38 taken from NCBI, comprising 2.75 billion nucleotides. It also uses 6-mer tokenization and has a vocabulary size of 4101. However, the DNABERT generates  $k$ -mers by iterating over the entire length  $n$  input sequence and extracting substrings of length  $k$ , which gives a computational complexity of  $O(n \times k)$ .

DNABERT-2-117M extends their implementation from MosaicBERT [20], another BERT-style encoder model that includes FlashAttention, Attention with Linear Biases (ALiBi), Gated Linear Units (GLU), a module to dynamically remove padded tokens, low precision

LayerNorm into the classic transformer encoder block, and a 30% masking ratio for the Masked Language Modeling (MLM) objective. DNABERT-2-117M uses BPE tokenization for DNA sequences with a vocabulary size of 4096. BPE tokenization has a complexity of  $O(k \times v \times \log V)$ , where  $k$  is the number of merge operations and  $V$  is the size of the vocabulary. In practice, BPE tokenization is efficient and scales well to large vocabularies. Additionally, DNABERT-2 was pretrained on the same human genome used in DNABERT, as well as a multi-species dataset encompassing genomes from 135 species. In total, this dataset includes 32.49 billion nucleotide bases, nearly 12 times the volume of the human genome dataset.

HyenaDNA uses a stack of Hyena operators, a subquadratic drop-in replacement for attention in Transformers. Where attention layers have  $O(N^2)$  complexity, hyena layers have  $O(N \log N)$  complexity. The Hyena operator matches quality in language modeling by using modified input projections, implicit convolutions and gating, all subquadratic operations. Thus, HyenaDNA “can reach context lengths of up to 500× longer than previous genomic Transformer models using dense attention, and train 160× faster at sequence length 1M (compared to Flash Attention)” [10]. It was pretrained on the human reference genome Hg38 using next token (nucleotide) prediction, with a vocabulary of 12— 4 nucleotides plus 8 special tokens. The complexity of nucleotide tokenization is linear,  $O(n)$ , with respect to the length of the input sequence.

### **Dataset Details**

The GUE benchmark included train, test, and development sets for every task. While some NT tasks shared the exact same data with GUE tasks, the splits were different. Additionally, the NT benchmark and Genomic Benchmark only provided train and test sets. To

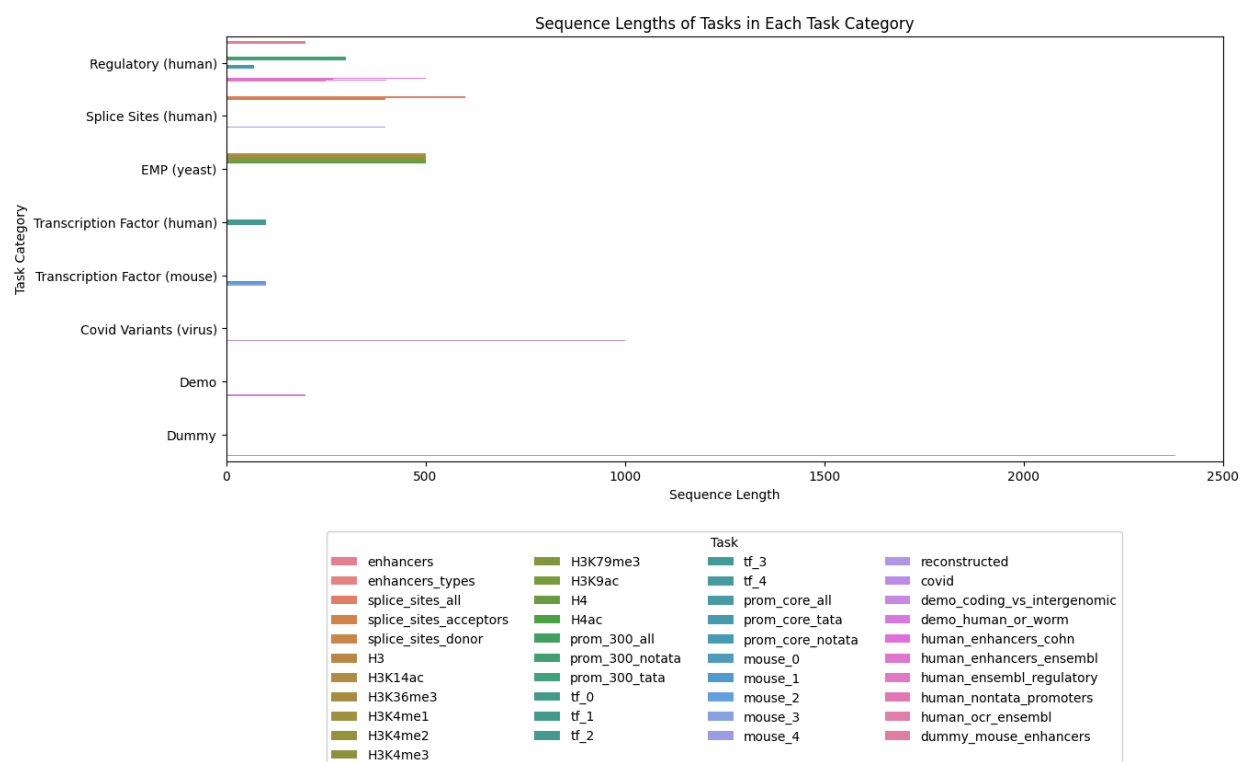
ensure consistency for comparison and ease of use with the HuggingFace library, all data was preprocessed with an 8:1:1 training, development, test split.

**Table A.1:** Task Summary for Tested Benchmarks

Publication Introduced In	Benchmark	Species	Task	Number of Classes	Sequence Length			
Nucleotide Transformer	Nucleotide Transformer Benchmark	Human	Enhancer	2	200			
			Enhancer Types	3	200			
			Promoter All	2	300			
			Promoter non-TATA	2	300			
			Promoter TATA	2	300			
			Splice All	3	400			
			Splice Acceptor	2	600			
			Splice Donor	2	600			
			H3	2	500			
			H3K4me1	2	500			
		Yeast	H3K4me2	2	500			
			H3K4me3	2	500			
			H3K9ac	2	500			
			H3K14ac	2	500			
			H3K36me3	2	500			
			H3K79me3	2	500			
			H4	2	500			
			H4ac	2	500			
			DNABERT2	GUE Benchmark	Human	Promoter All	2	300
						Promoter non-TATA	2	300
Promoter TATA	2	300						
Core Promoter All	2	70						
Core Promoter non-TATA	2	70						
Core Promoter TATA	2	70						
Splice Reconstruct	3	400						
Mouse	Transcription Factor Prediction 0-4 (5 total)	2			100			
	Transcription Factor Prediction 0-4 (5 total)	2			100			
Virus	Covid Variant Classification	9			1000			
	H3	2			500			
Yeast	H3K14ac	2			500			

		H3K36me3	2	500
		H3K4me1	2	500
		H3K4me2	2	500
		H3K4me3	2	500
		H3K79me3	2	500
		H3K9ac	2	500
		H4	2	500
		H4ac	2	500
		Enhancers Cohn	2	500
		Enhancers Ensembl	2	269
		Ensembl Regulatory	3	401
	Human	Promoter non-TATA	2	251
HyenaDNA	Genomic Benchmark	OCR Ensembl	2	315
		Coding vs. Intergenic	2	200
	Multi-Species	Human vs. Worm	2	200
	Mouse	Enhancers	2	2381

Figure A.1. Sequence Lengths of Tasks in Each Task Category



## Full Results

Figure A.2. Mean MCC Across Each Benchmark.

Depicts the mean MCC across each benchmark for 10 replications. From left to right: Nucleotide Transformer, GUE, Genomic Benchmark.

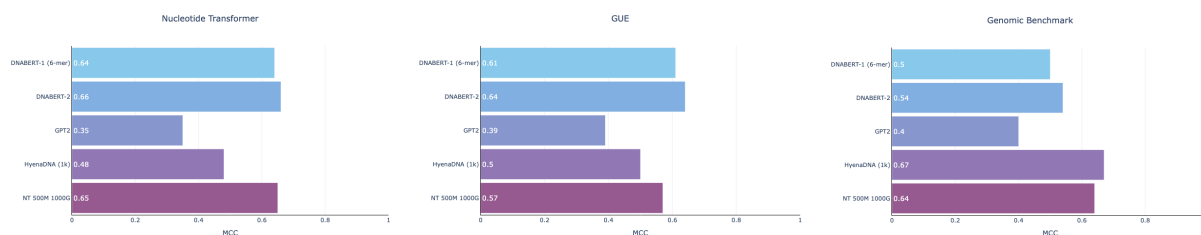


Figure A.3. Mean MCC Across Every Task

Presents the mean MCC across each task for 10 replications, with error bars depicting 1 standard deviation.

