Multilayer Hierarchical Tokenization

Anonymous ACL submission

Abstract

Tokenization is a critical component of large language models, yet standard Byte Pair Encoding (BPE) suffers from uncontrolled growth of low-frequency tokens as vocabulary size increases. We propose a multilayer hierarchical BPE framework that explicitly regulates vocabulary growth by treating tokens learned at one layer as atomic symbols at the next, biasing merges toward frequent patterns while suppressing rare tokens. As a proof of concept, we apply this approach to DNA sequence modeling under the DNABERT2 framework. Hierarchical tokenization across four layers reduces corpus size in characters by nearly fourfold while preserving total token count and average tokens per sequence. The method outperforms the DNABERT2 baseline on four of seven GUE tasks, maintains a near-Zipfian token frequency distribution, and shows limited gains on short-sequence tasks. These results demonstrate hierarchical BPE as a principled alternative to vocabulary scaling.

1 Introduction

With the rise of Large Language Models (LLMs), tokenization has become a fundamental component of modern language modeling, motivating extensive work on how to better exploit it. Prior studies examine the relationship between model performance and vocabulary size (Tao et al., 2024), as well as how tokenizers interact with corpus (Reddy et al., 2025). Among widely used approaches, Byte Pair Encoding (BPE), originally developed as a compression algorithm, has emerged as a practical and lossless tokenizer by progressively merging frequent character patterns, and has proven broadly effective across languages and domains, as evidenced by widely adopted frameworks such as SentencePiece (Kudo and Richardson, 2018). This effectiveness also extends to languages (e.g., Chinese) that lack explicit word boundaries (Meng et al., 2019).

Recent work has proposed a variety of modifications to BPE. For instance, Elsner et al. (2025) extend BPE to multidimensional data for visual tokenization, showing the merge process generalizes beyond text. Another effort introduces hierarchical variants: Dolga et al. (2025) propose a two-level scheme that applies a second tokenizer over character sequences derived from first-stage BPE tokens to control token granularity. A key remaining challenge is frequency imbalance under vocabulary scaling: Chung and Kim (2025) show that while larger vocabularies can simplify pattern learning, excessive growth shifts probability mass toward rare tokens and degrades performance. Lian et al. (2025) also study this imbalance and remove scaffold tokens that rarely occur on their own, to free capacity for other tokens; however, this operates as an encoding-time omission and does not regulate how the merge process continues to introduce increasingly sparse tokens as vocabulary grows.

Motivated by these observations, we introduce a multilayer hierarchical BPE tokenization scheme that operates across multiple layers to explicitly regulate vocabulary growth under scaling by treating one layer's tokens as the characters of the next layer. Rather than allowing vocabulary growth to expand uncontrollably toward infrequent tail tokens, our approach controls tail growth and encourages merges to focus on frequent patterns.

As a proof of concept, we apply this idea to DNA sequence modeling, which has increasingly adopted language-modeling paradigms. Following DNABERT (Ji et al., 2021), DNABERT2 (DB2) (Zhou et al., 2024), and Nucleotide Transformer (Dalla-Torre et al., 2025), genomic foundation models have demonstrated that DNA sequences can be effectively treated as a form of language. Similar to the Chinese language, DNA sequences lack explicit word boundaries, and due to their sensitive nature, any lossy tokenization cannot be tolerated. To avoid out-of-vocabulary tokens (Wu and Zhao,

2018), BPE has emerged as one of the most effective tokenization strategies for DNA, a finding strongly supported by DB2.

DB2 further demonstrates that careful tokenization design can be more effective than simply scaling model capacity: despite using a compact 117M-parameter model, it outperforms earlier DNABERT variants and remains competitive with much larger Nucleotide Transformers (2.5B parameters). Moreover, DB2 shows that increasing vocabulary size alone does not guarantee improved performance, with a vocabulary size of 4096 performing best within the range of 256 to 32,768 due to data sparsity effects. Building on this insight, our hierarchical encoding progressively compacts the effective corpus representation in terms of characters while preserving both the total number of tokens (T) and the average tokens per sequence (AT/S), jointly referred to as the token budget, which matches that produced by DB2's 4096-vocabulary tokenizer. Rather than confounding tokenization effects with changes in vocabulary size, model capacity, or training objectives, we adopt DB2's pretraining setup and introduce only minimal, task-specific adjustments during finetuning, allowing us to isolate the impact of hierarchical tokenization on representation efficiency and downstream performance.

To the best of our knowledge, we are the first to show that gradually compacting information into a denser corpus representation, while keeping the token budget fixed, can achieve comparable or even better model performance. We instantiate this idea using three hierarchical tokenization variants beyond the DB2 baseline, under the same token budget. Across layers, hierarchical encoding reduces the effective corpus size in characters by nearly fourfold while still outperforming DB2 on four out of seven downstream tasks on the GUE benchmark (Zhou et al., 2024). We further justify our multilayer hierarchical tokenizer by showing it preserves a near-Zipfian token frequency distribution (Piantadosi, 2014), closely matching that of the original tokenizer with the same token budget applied to the uncompressed corpus. We also examine the remaining three tasks and find that the proposed hierarchical scheme offers limited gains on short-sequence tasks ($\leq$100 base pairs).

The main contributions of this paper are: (1) We introduce a multilayer hierarchical BPE tokenization scheme that explicitly controls vocabulary growth by preventing uncontrolled expansion toward infrequent tail tokens, while encouraging merges to concentrate on frequent patterns through hierarchical re-encoding. (2) We demonstrate that applying this hierarchical tokenization across four layers reduces the effective corpus size in terms of characters by nearly fourfold and achieves comparable or better downstream performance than DB2 on GUE benchmark tasks, while preserving a near-Zipfian token frequency distribution.

V	Total Tokens	AT/S	V	Total Tokens	AT/S
128	13,069,392,325	402.29	8k	6,600,710,088	203.17
256	11,345,166,346	349.21	16k	6,086,573,061	187.35
512	9,950,317,795	306.28	32k	5,629,896,006	173.29
1k	8,870,426,460	273.04	64k	5,221,427,461	160.72
2k	7,963,786,431	245.13	128k	4,858,783,400	149.56
4k	**7,218,360,156**	**222.19**	256k	4,534,784,309	139.58

Table 1: Token budget across vocabulary sizes (V).

2 Methodology

As shown in Table 1, we apply BPE to the multi-species genome dataset (Zhou et al., 2024), which contains 32,487,832 sequences, using 12 different vocabulary sizes and record the token budget. As reported in DB2, a BPE with vocabulary size of 4096 yields the best downstream performance by mitigating data sparsity. We refer to the token budget produced by this tokenizer as the DB2 configuration. Following this, we introduce three hierarchical variants beyond the DB2 configuration, each matched to the same token budget as DB2.

Figure 1 shows the multilayer hierarchical BPE and its integration with the pretraining and finetuning pipeline. We first apply two BPE tokenizers with vocabulary sizes of 128 and 4096 to the raw dataset, referred to as Layer-1 (L-1) and Layer-1a (L-1a), respectively. Here, L-1a corresponds exactly to the DB2 configuration. From the L-1 tokenized corpus, we treat each of the 123 unique tokens (excluding 5 special tokens) as atomic characters and re-encode the dataset, which becomes the input corpus for L-2 and L-2a. On this corpus with 123 unique characters, we again apply two BPE tokenizers (256 and 2048 vocabularies) and use the 251 usable tokens from the 256-vocabulary tokenizer (L-2) as characters to construct a new corpus with 251 unique symbols. This procedure is repeated iteratively up to L-4 and L-4a. We do not extend beyond L-4, as at this stage the total number of tokens (approx. 7.9 billion) is already comparable to the token budget produced by the DB2 configuration. As summarized in Table 2, hierarchical re-encoding progressively compacts the corpus, reducing the character-level corpus size from roughly 32B to roughly 8B characters ($4\times$).

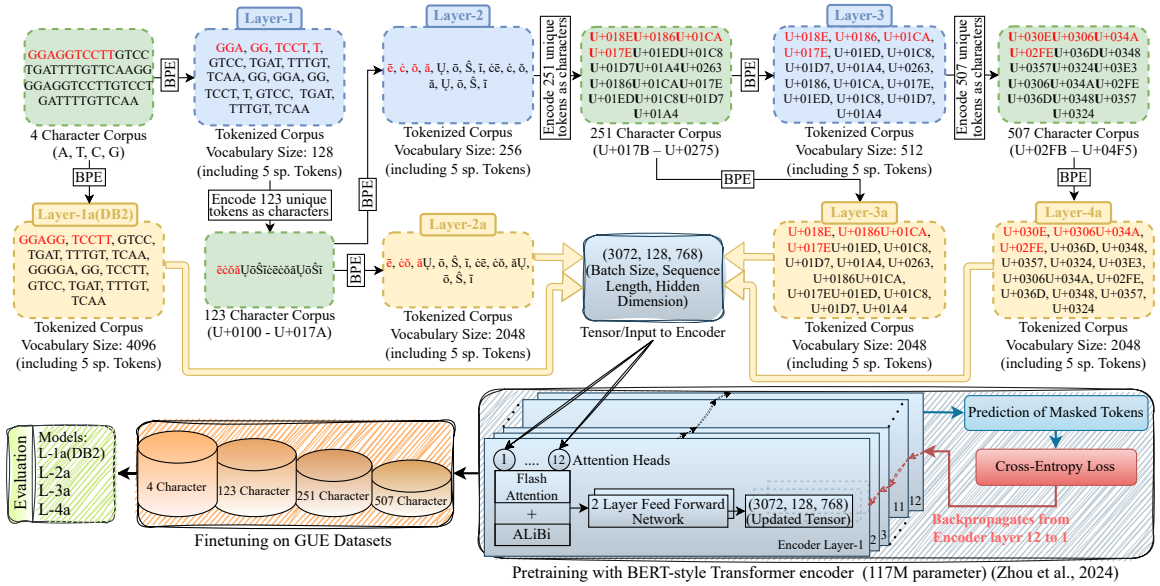


Figure 1: Overview of the proposed framework, showing multilayer hierarchical DNA tokenization and encoding followed by pretraining and task-specific fine-tuning stages.

Initial Corpus Stats.		Layer	After Tokenization Statistics		
UC	Total Char.		Vocabs	Total Tokens	AT/S
4	32,487,765,051	L-1	128	13,069,392,325	402.29
		L-1a	**4k**	**7,218,360,156**	**222.19**
123	13,069,392,325	L-2	256	9,676,398,768	297.85
		L-2a	**2k**	**7,161,711,392**	**220.44**
251	9,676,398,768	L-3	512	8,682,312,220	267.25
		L-3a	**2k**	**7,161,711,392**	**220.44**
507	8,682,312,220	L-4	1k	7,914,475,961	243.61
		L-4a	**2k**	**7,294,655,979**	**224.54**

Table 2: Token statistics across different corpora and hierarchical layers. UC = number of unique characters.

Here, L-1, L-2, L-3, and L-4 serve only as intermediate configurations used to construct progressively compacted corpora; models are pretrained exclusively using the four tokenizers L-1a (DB2), L-2a, L-3a, and L-4a. From now on, we call the pretrained models with the same name as their corresponding tokenizers.

2.1 Pretraining and Fine-tuning

We pretrain four BERT-style Transformer encoders while closely following the architectural and training setup of DB2. Starting from an initial corpus of 32 billion characters, the hierarchical encoding process represents the same information content in progressively more compact forms at higher layers. To support effective learning from these denser representations, we gradually reduce both the learning rate and masking probability at higher layers to reflect the increased abstraction and representational density of higher-layer tokens (Table 3).

After pretraining, we fine-tune the models on the GUE tasks following the DB2 fine-tuning protocol with two minor modifications: we omit the optional

	L-1a	L-2a	L-3a	L-4a
Learning Rate	0.0005	0.00045	0.000425	0.0004
Masking Probability(%)	15	14	12	10

Table 3: Training hyperparameters for different models.

Tokenizer	EMP	TF -M	CVC	TF -H	PD tata	PD o	CPD tata	CPD o	SSP
L-1a	132	29	236	29	82	77	27	21	110
L-2a	122	29	238	29	77	78	21	22	99
L-3a	122	29	238	29	77	78	21	22	99
L-4a	125	29	239	30	78	79	22	22	100

Table 4: P99 tokenized sequence lengths used to determine maximum sequence lengths for GUE tasks across tokenizers. PD-tata and PD-o denote tata-only and non-tata/all splits of PD; CPD follows same notation.

Low-Rank Adaptation (LoRA) and set the maximum sequence length to the 99th percentile (P99) of the token length distribution for each dataset to retain most sequences while reducing padding and memory overhead (Table 4). For each model, the GUE dataset is encoded in the same manner as its corresponding pretraining corpus.

3 Experimental Results

Average performance across GUE tasks is reported in Table 5, and task-specific results are presented in Table 8. Following standard practice, we evaluate model performance using Matthews Correlation Coefficient (MCC) for six tasks and F1-score for Covid Variants Classification (CVC). Overall, our three hierarchical tokenizer variants outperform DB2 on four out of seven tasks. We further examine the remaining three tasks and observe that their raw input sequences in GUE are short ($\leq$100 base pairs), whereas the other tasks use longer se-

	Yeast	Mouse	Virus	Human			
Model	EMP	TF-M	CVC	TF-H	PD	CPD	SSP
L-1a	55.38	71.00	68.17	66.61	81.79	70.15	85.16
L-2a	**55.40**	66.42	**69.31**	66.07	**82.89**	62.95	85.10
L-3a	**55.58**	66.10	**69.20**	65.82	**82.20**	70.13	**86.10**
L-4a	**56.09**	66.77	**68.36**	64.81	**82.53**	68.26	**86.13**

Table 5: Average performance on GUE tasks: Epigenetic Marks Prediction (EMP), Transcription Factor Prediction on Mouse (TF-M) and Human (TF-H), Covid Variants Classification (CVC), Promoter Detection (PD), Core PD (CPD), and Splice Site Prediction (SSP).

Tokenizer	EMP	TF-M	CVC	TF-H	PD	CPD	SSP
L-1a	22.50	25.87	21.92	24.09	22.76	26.02	22.27
L-2a	57.53	58.17	58.66	67.71	62.12	67.54	58.74
L-3a	75.99	80.67	75.70	80.65	76.32	84.31	75.56
L-4a	85.81	91.46	84.69	90.59	86.07	94.64	85.52

Table 6: Character to token compression ratio (%).

quences, including Promoter Detection (300 bp), Splice Site Prediction (400 bp), Epigenetic Marks Prediction (500 bp), and CVC (1000 bp).

To further analyze this behavior, we compute the token compression ratio for each GUE task, defined as the ratio of total tokens to total characters. As shown in Table 6, for a given tokenizer, compression ratios are nearly identical across most subtasks, with a few notable deviations. These deviations align with the tasks where no improvement over DB2 is observed, indicating that the proposed hierarchical compaction provides limited benefit for tasks dominated by short input sequences.

3.1 Controlling the Vocabulary Growth

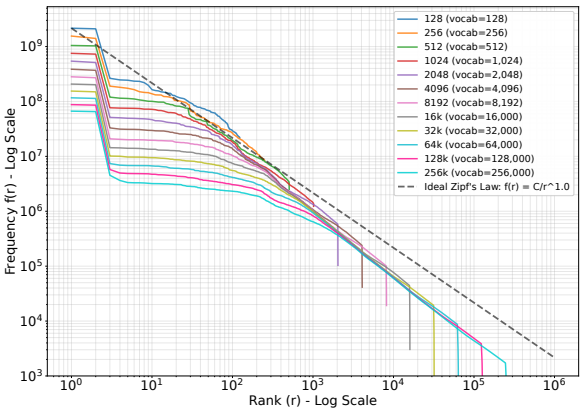
Table 7 summarizes the distribution of probability mass across head and tail regions of the vocabulary for different hierarchical tokenization layers. We compute the probability mass of a region as the sum of token frequencies within that region, normalized by the total token count. As hierarchical layers increase, probability mass increasingly concentrates in high-frequency regions, while the share held by low-frequency tail regions steadily decreases. This behavior reflects a controlled reshaping of the token frequency distribution that favors frequent patterns over rare ones without increasing the token budget.

3.2 Zipfian Behavior in our Tokenization

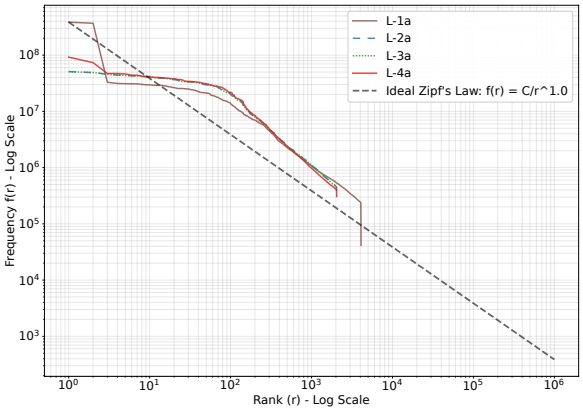
Token frequencies in both natural languages and genomic sequences are known to follow a Zipfian distribution (few tokens occur frequently while most are rare). Figure 2(a) shows that rank-frequency curves remain near-Zipfian across different vocabulary sizes on the multi-species genome corpus. Figure 2(b) shows that hierarchical layers preserve this behavior as well, indicating that the hierarchical procedure maintains the underlying token-

	Head			Tail		
Tokenizer	500 ↑	20% ↑	50% ↑	5% ↓	10% ↓	20% ↓
L-1a	68.72	76.38	90.00	0.64	1.39	3.04
L-2a	**78.69**	75.13	**90.28**	0.65	**1.36**	**2.93**
L-3a	**78.69**	75.13	**90.28**	0.65	**1.36**	**2.93**
L-4a	**80.92**	**77.44**	**91.59**	**0.57**	**1.19**	**2.55**

Table 7: Probability mass coverage of head and tail regions across hierarchical tokenization layers.



(a) Zipf plots for vocabularies in Table 1.



(b) Zipf plots across hierarchical layers.

Figure 2: Zipf-style rank-frequency distributions (Dong and Su, 2025). r = token rank, C = normalization constant that sets the frequency scale, and α controls the slope (typically close to $\alpha \approx 1$ for Zipf-like behavior).

frequency structure without distorting it.

4 Conclusion

We introduce a multilayer hierarchical BPE tokenization scheme that progressively compacts the corpus under a fixed token budget, controlling vocabulary growth and suppressing low-frequency tail tokens. Under the DB2 pretraining setup, it reduces the character-level corpus size by about $4\times$, improves performance on four of seven GUE tasks without changing model architecture or capacity, and preserves near-Zipfian rank-frequency behavior. Gains are limited on short-sequence tasks but clearer on moderate and longer sequences, highlighting that principled control over how tokenization allocates probability mass can be as important as vocabulary size for representation efficiency.

Limitations

This work evaluates hierarchical BPE exclusively in the context of DNA sequence modeling. While the proposed framework is motivated by general properties of delimiter-free sequences and heavy-tailed token distributions, its effectiveness on other domains, particularly natural languages, has not been empirically validated and remains an important direction for future work. In addition, all experiments are conducted using a relatively small 117M-parameter BERT-style model following the DB2 setup; it is unclear whether the same trends will hold for substantially larger models, where representation capacity and optimization dynamics may differ. Finally, our hierarchical design intentionally suppresses low-frequency tail tokens, which improves efficiency but can slightly degrade performance on tasks dominated by short input sequences, indicating a potential trade-off between tail coverage and compactness that warrants further investigation.

References

Woojin Chung and Jeonghoon Kim. 2025. Exploiting vocabulary frequency imbalance in language model pre-training. In *Proceedings of the Thirty-ninth Annual Conference on Neural Information Processing Systems*.

Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, and 1 others. 2025. Nucleotide transformer: Building and evaluating robust foundation models for human genomics. *Nature Methods*, 22:287–297.

Rares Dolga, Lucas Maystre, Tudor Berariu, and David Barber. 2025. From characters to tokens: Dynamic grouping with hierarchical bpe. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 11154–11162, Taipei, Taiwan. Association for Computational Linguistics.

Dong Dong and Weijie Su. 2025. Length-max tokenizer for language models. *Preprint*, arXiv:2511.20849.

Tim Elsner, Paula Usinger, Julius Nehring-Wirxel, Gregor Kobsik, Victor Czech, Yanjiang He, Isaak Lim, and Leif Kobbelt. 2025. Multidimensional byte pair encoding: Shortened sequences for improved visual data generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 21331–21341.

Yanrong Ji, Zhen Zhou, Haotian Liu, and Ramana V. Davuluri. 2021. Dnabert: Pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics*, 37:2112–2120.

Taku Kudo and John Richardson. 2018. Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium. Association for Computational Linguistics.

Haoran Lian, Yizhe Xiong, Jianwei Niu, Shasha Mo, Zhenpeng Su, Zijia Lin, Hui Chen, Jungong Han, and Guiguang Ding. 2025. Scaffold-bpe: Enhancing byte pair encoding for large language models with simple and effective scaffold token removal. In *Proceedings of the AAAI Conference on Artificial Intelligence*. Association for the Advancement of Artificial Intelligence.

Yuxian Meng, Xiaoya Li, Xiaofei Sun, Qinghong Han, Arianna Yuan, and Jiwei Li. 2019. Is word segmentation necessary for deep learning of chinese representations? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3242–3252, Florence, Italy. Association for Computational Linguistics.

Steven T. Piantadosi. 2014. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic Bulletin & Review*, 21:1112–1130.

Varshini Reddy, Craig W. Schmidt, Yuval Pinter, and Chris Tanner. 2025. How much is enough? the diminishing returns of tokenization training data. *arXiv preprint arXiv:2502.20273*.

Chaofan Tao, Qian Liu, Longxu Dou, Niklas Muennighoff, Zhongwei Wan, Ping Luo, Min Lin, and Ngai Wong. 2024. Scaling laws with vocabulary: Larger models deserve larger vocabularies. In *Proceedings of the Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Yingting Wu and Hai Zhao. 2018. Finding better subword segmentation for neural machine translation. In *Proceedings of the 21st International Conference on Text, Speech, and Dialogue (TSD 2018)*, Cham. Springer.

Zhen Zhou and 1 others. 2024. Dnabert-2: Efficient foundation model and benchmark for multi-species genome. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

A Additional Experimental Results

A.1 Setup

All experiments were conducted on Intel Ice Lake compute nodes with dual 32-core Intel Xeon Platinum 8358 processors. Tokenization was performed on a single big-memory node with 2 TB RAM. Model pretraining used 12 nodes, each with 512 GB memory and two NVIDIA A100 GPUs connected via NVLink, for a total of 24 GPUs with

Task Category	Sub Task	L-1a	L-2a	L-3a	L-4a
Epigenetic Marks Prediction (EMP)	H3	75.83	**76.35**	**76.42**	75.33
	H3K14ac	52.32	50.55	**52.72**	51.97
	H3K36me3	58.04	57.43	**59.25**	**59.38**
	H3K4me1	49.86	**50.51**	49.83	**52.42**
	H3K4me2	34.76	**36.58**	**35.04**	32.74
	H3K4me3	36.14	**38.07**	33.97	**38.90**
	H3K79me3	63.03	62.81	61.68	**63.72**
	H3K9ac	54.89	54.17	**57.51**	**57.09**
	H4	81.43	79.92	80.76	81.08
	H4ac	46.47	**47.65**	**48.62**	**48.22**
Promoter Detection (PD)	all	87.76	87.12	86.93	87.33
	notata	94.12	**94.16**	**94.85**	93.71
	tata	63.48	**67.38**	**64.82**	**66.55**
TF Prediction (Human) (TF-H)	TF-0	66.72	**69.99**	66.61	**71.39**
	TF-1	72.79	72.02	70.40	71.41
	TF-2	61.00	**63.57**	**61.81**	60.21
	TF-3	57.04	51.36	56.05	49.82
	TF-4	75.51	73.51	74.24	71.22
Core Promoter Detection (CPD)	all	67.36	64.57	67.21	65.01
	notata	69.84	67.67	69.05	67.55
	tata	73.26	56.61	**74.14**	72.22
TF Prediction (Mouse) (TF-M)	TF-0	63.48	54.78	54.86	55.13
	TF-1	84.83	**85.55**	84.72	83.87
	TF-2	79.89	**87.27**	**80.64**	71.57
	TF-3	78.25	58.89	63.25	76.05
	TF-4	48.53	45.61	47.03	47.21
Covid Variants Classification (CVC)	Covid	68.17	**69.31**	**69.20**	**68.36**
Splice Site Prediction (SSP)	Reconstruct	85.16	85.10	**86.10**	**86.13**

Table 8: Task-level performance of all models on individual GUE benchmark subtasks. Boldface denotes results better than L-1a (DB2).

NCCL-based distributed training. Fine-tuning was carried out on a single node with four NVIDIA A100 GPUs and 512 GB memory using bfloat16 mixed precision.

A.2 Task-Level Results on the GUE Benchmark

Table 8 reports a fine-grained breakdown of model performance across individual subtasks of the GUE benchmark. Results are shown for the DB2 baseline (L-1a) and hierarchical tokenization variants at higher layers. All tokenizers were trained using Hugging Face's BPE tokenizer implementation, closely matching the DB2 tokenizer design. All models were pretrained using a maximum sequence length of 128 with line-by-line input processing enabled. Pretraining employed an effective batch size of 3072, following the DB2 setup. Fine-tuning was performed with a learning rate of 3×10^{-5} and the same number of training epochs as used in DB2 for each downstream task. Fine-tuning used a per-device batch size of 16 with no gradient accumulation across four GPUs, resulting in an effective training batch size of 64, and a per-device evaluation batch size of 32. As in DB2, we selected the checkpoint with the lowest validation loss and used it for final test evaluation.

n	Raw DNA space				Converted-symbol space			
	L-1a	L-2a	L-3a	L-4a	L-1a	L-2a	L-3a	L-4a
1	4	4	4	4	4	123	251	507
2	7	13	13	13	7	1317	1760	1526
3	34	41	41	41	34	347	23	6
4	142	122	122	124	142	151	9	4
5	539	332	332	321	539	22	–	–
6	1415	820	820	720	1415	28	–	–
7	1450	612	612	703	1450	35	–	–
8	378	75	75	83	378	16	–	–
9	58	7	7	11	58	1	–	–
10	15	2	2	5	15	–	–	–
11	10	1	1	1	10	–	–	–
12	7	1	1	2	7	–	–	–
13	7	2	2	2	7	–	–	–
14	5	2	2	2	5	–	–	–
15	3	–	–	–	3	–	–	–
16	8	7	7	8	8	3	–	–
17	2	–	–	–	2	–	–	–
18	1	–	–	–	1	–	–	–
24	1	–	–	–	1	–	–	–
25	1	–	–	–	1	–	–	–
32	4	2	2	3	4	–	–	–

Table 9: Exact n-gram distributions of vocabulary tokens across hierarchical tokenization layers (multi-species genome dataset).

A.3 Token n-gram Statistics

Table 9 reports exact n-gram statistics of vocabulary tokens under two representations. In the left block, token lengths are measured after fully back-mapping each token to the raw DNA alphabet, showing how many nucleotide bases (A, T, C, G) each token spans. In the right block, token lengths are measured directly in the converted-symbol space, where tokens are treated as standard n-grams over the symbols produced by the corresponding tokenizer layer.

This comparison highlights how the same vocabulary can be viewed either as composite patterns over raw DNA characters or as ordinary n-grams in the converted corpus. As layers increase, tokens correspond to similar DNA-level patterns while forming progressively shorter n-grams in the converted representation, reflecting progressive abstraction introduced by hierarchical tokenization. Moreover, comparing L-2a and L-3a in Table 9 shows that although their n-gram distributions differ in the converted-symbol space, back-mapping to raw DNA yields the same n-gram distribution. This helps explain why they exhibit similar behavior despite being trained on different corpora, sharing the same token budget (Table 2), and producing identical rank-frequency curves (Figure 2).

For each layer and representation space, we compute the expected token length as a weighted mean

	L-1a	L-2a	L-3a	L-4a
Avg. Tok. Len. (DNA space)	6.47	6.09	6.09	6.18
Avg. Tok. Len. (Converted-symbol space)	6.47	2.50	1.90	1.76

Table 10: Average token lengths computed from the n-gram distributions in Table 9.

over the token-length distribution:

$$\langle n \rangle \;=\; \frac{\sum_n n\, N_n}{\sum_n N_n}, \tag{1}$$

where N_n denotes the number of vocabulary tokens whose length is exactly n, as reported in Table 9. As shown in Table 10, hierarchical tokenization substantially reduces the average token length in the converted-symbol space, while maintaining a similar average span in raw DNA space compared to the benchmark. This indicates that higher-layer tokenizers produce more compact symbolic representations without significantly altering the amount of underlying DNA context captured by each token.