
EMBEDDING IS (ALMOST) ALL YOU NEED: RETRIEVAL-AUGMENTED INFERENCE FOR GENERALIZABLE GENOMIC PREDICTION TASKS

Nirjhor Datta^{1,2}Swakkhar Shatabda²M Sohel Rahman¹

¹**Department of Computer Science and Engineering,**
Bangladesh University of Engineering and Technology,
West Palashi, Dhaka 1205, Bangladesh

²**Department of Computer Science and Engineering,**
BRAC University, Dhaka 1205, Bangladesh

August 8, 2025

ABSTRACT

Large pre-trained DNA language models such as DNABERT-2, Nucleotide Transformer, and HyenaDNA have demonstrated strong performance on a wide range of genomic benchmarks. However, most applications rely on expensive fine-tuning, which benefits significantly from the similar distribution between training and test data. In this work, we investigate whether task-specific fine-tuning is always necessary. We show that simple embedding-based pipelines extracting fixed representations from these models and feeding them into lightweight classifiers can achieve competitive performance. Moreover, in independent evaluation settings with different data distributions, embedding-based methods outperform fine-tuning in several tasks while reducing inference time by 10 \times –20 \times . Our results suggest that embedding extraction is not only a strong baseline but also a more generalizable and efficient alternative to fine-tuning, particularly for deployment in diverse or unseen genomic contexts. Qualitatively, our retrieval-augmented pipeline combining fixed transformer embeddings (DNABERT-2, Nucleotide Transformer, HyenaDNA) with lightweight sequence features achieved competitive result across nine genomic tasks. Also, DNABERT-2 and HyenaDNA embeddings with simple sequence features closely matches or exceeds fine-tuned transformer performance on two new independent test set, enhancer and promoter classification tasks, without any model retraining. Quantitatively, our embedding-based approaches consistently outperform fine-tuning in terms of carbon efficiency while maintaining competitive accuracy. For enhancer classification, embedding with zCurve achieves an accuracy of 0.68 using HyenaDNA (vs. 0.58 for fine-tuned HyenaDNA) with almost 88% reduction in inference time and over 8 \times lower carbon emissions (0.02 kg vs. 0.17 kg CO₂). For non-TATA promoter classification, fixed embeddings from DNABERT-2 combined with zCurve, GC content, or AT/GC ratio reach 0.85 accuracy, compared to 0.89 for fine-tuning, but with a 22 \times lower carbon footprint (0.02 kg vs. 0.44 kg CO₂). Similarly, HyenaDNA with embeddings achieves up to 0.84 accuracy while emitting just 0.01 kg CO₂ compared to 0.04 kg for fine-tuning. These results show embedding-based pipelines can offer over 10 \times higher carbon efficiency while achieving similar or better predictive performance. The code is available here: <https://github.com/NIRJHOR-DATTA/EMBEDDING-IS-ALMOST-ALL-YOU-NEED>.

1 Introduction

Genomic sequence analysis is one of the most promising areas of research in bioinformatics. The ability to understand the functional elements of the genome, such as promoters, enhancers, and splice sites, has vast implications for fields, such as gene regulation, personalized medicine, and disease prediction [1]. Traditional computational methods often rely on handcrafted features or pre-defined models, which, while useful, can struggle to capture the complexities inherent in genomic sequences [2]. Recent advances in transformer-based models, originally developed for natural language processing, have been successfully adapted to nucleotide sequences, yielding state-of-the-art performance on a variety of benchmarks [3, 4, 5].

Despite their empirical success, these transformer-based methods are computationally expensive to fine-tune and deploy, often requiring hundreds of GPU hours and substantial memory resources. Prior work has highlighted the environmental impact of large-scale model training, showing that the carbon footprint of training a single transformer can rival that of multiple automobiles over their lifetimes [6, 7]. In genomics, where sequence lengths frequently exceed 10 Kb, the quadratic complexity of self-attention exacerbates these challenges, making full fine-tuning impractical in many research and clinical settings.

To address these limitations, retrieval-augmented methods, originally proposed in the NLP domain offer a promising alternative. By indexing fixed transformer embeddings in FAISS (Facebook AI Similarity Search) and performing nearest-neighbor retrieval at inference time, one can sidestep full model fine-tuning while still leveraging rich contextual representations [8, 9]. Recent works in gene interaction prediction has shown that retrieval-augmented generation can achieve better accuracy with substantially lower computational cost [10]. In parallel, hybrid approaches that combine pretrained embeddings with handcrafted genomic features (e.g., GC content, z-curve, pseudoKNC) have demonstrated marginal gains in classification accuracy while maintaining lightweight inference. So, there is a need for an approach which can leverage the power of large pre-trained models, without the need for fine-tuning, but still achieve state of the art results on various genomic classification tasks. This motivates the exploration of alternative methods that can provide high performance without the computational cost and complexities associated therewith.

In this work, we propose a comprehensive, retrieval-augmented genome classification framework that: (1) employs pretrained transformer embeddings (DNABERT-2, Nucleotide Transformer, HyenaDNA) as fixed representations; (2) optionally augments these embeddings with biologically motivated features such as GC content, z-curve components, AT/GC ratio, cumulative skew, and pseudoKNC; and (3) uses FAISS L_2 indexing for fast k-nearest-neighbor retrieval and weighted voting. We have evaluated our approach on nine diverse genomic benchmarks. The central focus of this research is to investigate whether fixed, pretrained embeddings can serve as a competitive alternative to full model fine-tuning, particularly in scenarios that demand strong generalizability. To this end, we conduct evaluations on a new, independent test set to assess how well the learned representations transfer beyond the training distribution. Our embedding-based approach achieves comparable or even superior performance to fine-tuned models, while reducing GPU fine-tuning time by over 70% and reducing carbon emission by upto 77.5x, thereby promoting a more sustainable and efficient paradigm aligned with Green AI principles [11].

The key contributions of this work are as follows.

1. **Retrieval-Augmented Classification Pipeline:** We propose a hybrid framework that combines pretrained transformer embeddings (DNABERT-2, Nucleotide Transformer, HyenaDNA) with handcrafted genomic features (e.g., pseudoKNC, GC-content, z-curve) and a lightweight kNN classifier. By indexing precomputed embeddings in FAISS and using majority/weighted voting, we avoid full fine-tuning of large models at inference time, drastically reducing GPU memory usage and compute cost.
2. **Modular, Scalable, and Carbon-Efficient Architecture:** The embedding and retrieval stages are decoupled, enabling plug-and-play substitution of any pretrained model or handcrafted feature set. This modularity allows flexible trade-offs between accuracy and resource usage, with our pipeline achieving up to $77.5\times$ lower CO_2 emissions compared to full fine-tuning while maintaining competitive performance.
3. **Comprehensive Empirical Evaluation:** We benchmark our method on nine publicly available genomic datasets (e.g., Human Ensembl Regulatory, Drosophila Enhancers Stark, Human Non-TATA Promoters), comparing embedding-only, feature-augmented, and fine-tuned variants across three transformer models. We evaluated our approach on an independent test set across two genome classification tasks, using both a fine-tuned model trained on a genomic benchmark dataset and a retrieval-based method that leverages embeddings from the same dataset.

2 Related Work

Transformer architectures have revolutionized sequence modeling across domains, including genomics. DNABERT [3] introduced a BERT-style model pre-trained on k-mers from the human genome, achieving strong performance on promoter and enhancer classification. However, limitations such as redundant k-mer tokenization, short context windows (512 tokens), and single-species training motivated DNABERT-2 [12], which uses byte pair encoding, multi-species data, and longer input handling via gradient checkpointing.

Nucleotide Transformer (NT) [4] extends this trend by pre-training large-scale transformer models (up to 2.5B parameters) on thousands of genomes. NT embeddings support strong performance across diverse tasks even in low-data settings, demonstrating the utility of context-aware nucleotide representations.

HyenaDNA [5], inspired by efficient implicit convolution mechanisms, offers longer context windows with lower time complexity. It outperforms existing methods on multiple GenomicBenchmarks tasks while using fewer parameters and resources.

Despite these advances, fine-tuning large models remains energy-intensive. Studies by Strubell [6] and Patterson [7] highlight the environmental cost of deep learning, advocating for energy-efficient alternatives. In response, Green AI [11] promotes trade-offs between performance and resource consumption using metrics like FLOPs, GPU hours, and emissions.

Retrieval-Augmented Generation (RAG) [8] has emerged as a lightweight alternative, replacing full model updates with external memory lookups via FAISS [9]. GeneRAG [10] applied this idea to genomics, boosting performance without costly fine-tuning.

While transformer embeddings have been shown to cluster functionally related sequences [3, 4, 5], few works compare retrieval-based classification across multiple models and tasks. We address this by benchmarking DNABERT-2, HyenaDNA, and NT under a unified retrieval framework, augmented with biologically meaningful descriptors, offering an efficient and accurate alternative to fine-tuning.

3 Proposed Method

3.1 Overview

We propose a novel retrieval-augmented genomic classification framework that eliminates the need for fine-tuning large genomic language models. First, raw DNA sequences are embedded using a pretrained model (such as, DNABERT-2 or Nucleotide Transformer) without any parameter updates. During inference, a nearest-neighbor retrieval mechanism is employed to fetch relevant training embeddings based on similarity. The retrieved context is combined with the test sequence representation. Classification is performed directly by comparing similarity scores, avoiding the need for additional fully connected layers or retraining. This lightweight approach enables efficient zero-shot or few-shot classification, particularly suitable for imbalanced datasets. Throughout the rest of the paper, we define this pipeline as base framework. Top level diagram of our proposed method is presented at Figure 1. Our proposed framework leverages

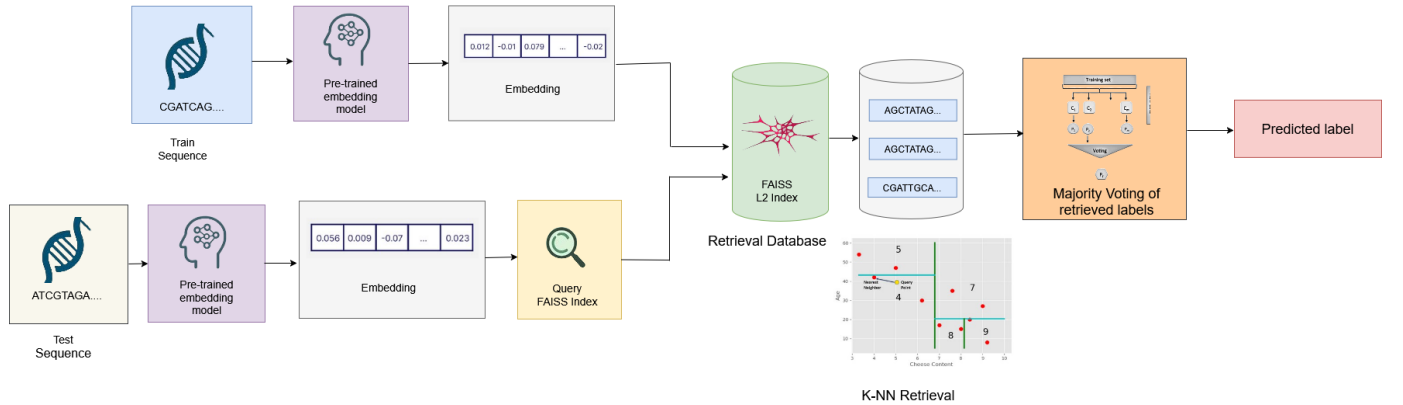


Figure 1: Top-level overview of the proposed framework

a training-free, retrieval-augmented classification paradigm for genomic sequence classification. The methodology

involves three core components: (1) hybrid feature extraction, (2) retrieval using FAISS similarity search, and (3) label prediction via weighted voting. This approach eliminates the need for training complex models while maintaining competitive classification performance.

3.2 Hybrid Feature Extraction

Each DNA sequence is first encoded using two complementary representations as follows:

Pretrained Embeddings. We utilize embeddings obtained from a pretrained transformer model trained on genomic sequences. These embeddings capture the high-level contextual semantics of the sequences.

$$\mathbf{e}_i = f_\theta(\mathbf{x}_i) = \frac{1}{L_i} \sum_{t=1}^{L_i} h_t, \quad (1)$$

where f_θ is the pretrained transformer model with frozen weights and h_t are the hidden representations of each nucleotide position t in the sequence \mathbf{x}_i .

Biological Feature Engineering. Inspired by PyFeat [13], to complement the high-dimensional representations from pretrained embeddings, we extract set of biologically-inspired handcrafted features that capture essential patterns and characteristics of DNA sequences, namely, AT/GC Ratio, Z-curve features, Cumulative GC and AT skew and Pseudo k-mer composition.

The learned embedding vector \mathbf{e}_i is ℓ_2 -normalized and linearly weighted by a coefficient α , while the handcrafted feature vector \mathbf{f}_i is min-max normalized and weighted by β . These scaled representations are concatenated to form the final hybrid feature vector:

$$\mathbf{h}_i = \alpha \cdot \frac{\mathbf{e}_i}{\|\mathbf{e}_i\|} \parallel \beta \cdot \text{MinMax}(\mathbf{f}_i),$$

where \parallel denotes vector concatenation. We found that using $\alpha = 0.8$ and $\beta = 0.2$ yielded better performance compared to other combinations of weights.

These handcrafted features are concatenated and min-max normalized before fusing with deep embeddings to form the final hybrid representation used for similarity search and classification.

3.3 Similarity-Based Retrieval

Given the hybrid representation of the training data, we build a similarity search index using the FAISS library, which supports efficient nearest neighbor search based on inner product similarity. For a test sequence with hybrid vector \mathbf{h}_q , the system retrieves its top- k most similar training samples from the index. A dynamic k retrieval strategy is adopted, where k is increased adaptively if the mean similarity of the top- k results falls below a predefined threshold τ .

$$k = \min \left\{ k' : \frac{1}{k'} \sum_{j=1}^{k'} s_j \geq \tau \right\}. \quad (2)$$

3.4 Weighted Voting for Prediction

Let $\mathcal{N}_q = \{(\mathbf{h}_{i_j}, y_{i_j}, s_j)\}_{j=1}^k$ denote the retrieved neighbors, where y_{i_j} is the label and s_j the similarity score; a sigmoid transformation is applied to convert similarity scores into soft weight, $w_j = \frac{1}{1+e^{-s_j}}$.

These weights are then accumulated per class to determine the final prediction:

$$\hat{y}_q = \arg \max_c \sum_{j=1}^k w_j \cdot \mathbb{I}[y_{i_j} = c]$$

4 Experimental Setup

First, we evaluate our proposed framework along with additional handcrafted features on genomic benchmark. Even though the GenBench paper [14] presents fine-tuning results on the Genomic Benchmark [15], it does not provide the specific hyperparameters used for those models. As a result, a direct comparison with our approach is not feasible. However, our primary objective is not merely to achieve competitive performance with fine-tuned models, but also

to evaluate the generalizability of our method. Therefore, our main experiment focuses on assessing how well our approach performs on a different distribution of data for the same task, highlighting its robustness and effectiveness in generalization and compare the results against fine-tuned based approach in a similar setting. This is why, in the next stage, we evaluate DNABERT-2 and Hyenadna using both our proposed framework and standard fine-tuning on two downstream genomics tasks, namely, enhancer classification and non-TATA promoter classification using two new independent test sets. Due to its substantially larger parameter count (2.5B) compared to DNABERT-2 (117M) and HyenaDNA (6.5M), the Nucleotide Transformer was excluded from evaluation at this stage. All the models are trained on a A6000 GPU 48 GB, RAM: 64GB DDR5 SSD.

4.1 Datasets

We evaluate our models on nine publicly available genomic benchmark datasets (Table 1) covering diverse biological classification tasks. These datasets span enhancer detection, promoter classification, and other regulatory sequence identification tasks in various organisms including humans, mice, worms, and Drosophila. All datasets are sourced from the Genomic Benchmark [15].

Table 1: Summary of Genomic Datasets Used in Experiments ($|C|$ indicates number of classes in the dataset).

#	Dataset Name	Train Samples	Test Samples	$ C $
1	Human Ensembl Regulatory	231,348	57,713	3
2	Drosophila Enhancers Stark	5,184	1,730	2
3	Demo Coding vs Intergenic Seqs	75,000	25,000	2
4	Demo Human vs Worm	75,000	25,000	2
5	Human Enhancers Cohn	20,843	6,948	2
6	Human Enhancers Ensembl	123,872	30,972	2
7	Human OCR Ensembl	139,804	34,952	2
8	Human Non-TATA Ensembl	27,097	9,034	2
9	Mouse Enhancers Ensembl	968	242	2

For our second and main task, checking the performance of embedding based mechanisms against fine-tuned based models, we have chosen two downstream tasks, namely, human enhancer classification and human non-tata promoter classification. We have used human enhancer ensemble and human non-tata promoters from genomic benchmark as train set. For human enhancer classification task, we have evaluated ienhancer [16] test set and human non-tata promoter classification task, dataset described in [17] as an independent test dataset. For the *ienhancer* test set, we have merged strong enhancers and weak enhancers as enhancer keeping non-enhancer as it is since our original *human enhancer ensemble* dataset from genomic benchmark has only 2 labels - enhancer and non-enhancer.

4.2 Preprocessing for Fine-Tuning-Based Models

We convert DNA sequences into k-mer representations ($k=6$) to capture local nucleotide patterns. These k-mers are then tokenized using the corresponding model tokenizer with a maximum input length of 512. Labels are mapped and datasets are formatted into PyTorch tensors for compatibility with the HuggingFace Trainer API.

4.3 Model Variants

We evaluate the following model configurations:

- **Embedding (emb):** Sequence is encoded using pre-trained biological language models embeddings without fine-tuning.
- **+ Feature:** Inspired by PyFeat [13], to complement the high-dimensional representations from pretrained embeddings, we extract set of biologically-inspired handcrafted features that capture essential patterns and characteristics of DNA sequences. The detailed way of calculating additional features are available at
- **DNABERT-2 (Fine-tuned):** Full model fine-tuned end-to-end on the training set. Fine-tuning parameters on new independent datasets are presented at Table 2.

All hybrid models use a simple KNN classifier on top of the combined feature vector.

4.4 Optimized Retrieval Pipeline

To accelerate the inference stage of our retrieval-based evaluation, we implemented an optimized version of our pipeline with the following key improvements:

Table 2: Hyperparameter Settings

Hyperparameter	Value
Learning Rate	2×10^{-5}
Batch Size (per device)	32
Number of Epochs	3
Weight Decay	0.01

- **Batch Processing:** Instead of computing embeddings one at a time, we use a PyTorch DataLoader to process input sequences in batches, significantly improving GPU utilization and memory efficiency.
- **Mean Pooling:** Token-level outputs from the HyenaDNA model are pooled using mean pooling to obtain fixed-length embeddings, replacing any earlier simplistic representations.
- **Efficient Retrieval:** The brute-force distance computation is replaced by a FAISS IndexFlatL2 index, enabling much faster nearest neighbor search on the CPU.
- **Inference Time Tracking:** A timing mechanism is added to measure the total runtime of the retrieval pipeline, providing a quantitative measure of efficiency gains.

All other steps, including the use of majority voting over the top- k retrieved labels, remain unchanged from the original implementation. This optimized pipeline was used in the second and main set of experiments, which evaluated the performance of embedding-based methods against fine-tuned models. These optimizations significantly reduced retrieval latency while preserving prediction quality.

5 Results

5.1 Performance on Genomic Benchmark

We have presented the algorithms for our base framework (Figure 1) and the framework along with handcrafted features. Top 1% accuracy of base and fine-tuned models of DNABERT-2 (117 M), Nucleotide Transformer (2.5 B) and HyenaDNA (6.5 M) across all the datasets of genomic benchmark is presented at Table 3.

- **DNABERT-2 Embeddings** Accuracy ranges from 0.58 (Human Ensembl Regulatory) to 0.91 (Demo Human vs Worm). Notably, DNABERT-2 embeddings achieve 0.85 on Demo Coding vs Intergenic and 0.88 on Human Non-TATA Ensembl.
- **Nucleotide Transformer Embeddings** Accuracy ranges from 0.57 (Human Ensembl Regulatory) to 0.90 (Demo Human vs Worm). In many cases (e.g., Demo Coding vs Intergenic at 0.85, Demo Human vs Worm at 0.90), its performance closely matches DNABERT-2.
- **HyenaDNA Embeddings** Accuracy spans 0.61 (Drosophila Enhancers Stark) to 0.83 (Demo Coding vs Intergenic and Demo Human vs Worm). Across datasets, HyenaDNA embeddings lag behind DNABERT-2 and Nucleotide Transformer by 0.1 - 0.2 AUROC points on average.

These results indicate that pretrained transformer embeddings particularly those specialized for genomic sequences encode discriminative features even without task-specific training. HyenaDNA’s relatively lower embedding performance suggests that its architectural modifications (e.g., convolutional attention mechanisms for long contexts) may not align as directly with the classification tasks in GenBench. The results presented in Table 3 are comparable to those reported in the literature with heavy fine-tuning; however, due to the lack of detailed fine-tuning configurations in the GenBench [14], a direct comparison could not be performed.

5.2 Comparative performance on downstream tasks

We have chosen two down stream tasks from genomic benchmark, namely, human enhancer classification and human non-tata promoter classification. We evaluated both the fine-tuned model and our proposed framework, trained on the same training set (human enhancer ensemble and human non-tata promoters from Genomic Benchmark), using a new independent test dataset to assess their generalization performance. We have used DNABERT-2 and HyenaDNA with both fine-tuning and our RAG based approach (Suppl. Section Algorithm in the Appendix).

For the embedding-based approach, we first extracted sequence embeddings using a frozen DNABERT-2 model and combined them with various handcrafted features, including GC content, zCurve components, AT/GC ratios, cumulative

Table 3: Top 1% Accuracy Performance of Embedding Variants and Fine-Tuned Models Across Nine Genomic Datasets. **Emb** means the embedding of that particular model. Best scores are in **bold**

DNABERT-2						
Dataset	Emb	+gcCont	+zCurve	+AT/GC	+cumSkew	+pseudoKNC
Human Ensembl Regulatory	0.58	0.59	0.57	0.59	0.59	0.60
Drosophila Enhancers Stark	0.70	0.71	0.68	0.70	0.68	0.68
Demo Coding vs Intergenomic	0.85	0.84	0.86	0.86	0.84	0.87
Demo Human vs Worm	0.91	0.92	0.92	0.92	0.91	0.92
Human Enhancers Cohn	0.70	0.71	0.72	0.72	0.72	0.72
Human Enhancers Ensembl	0.71	0.72	0.73	0.73	0.73	0.73
Human OCR Ensembl	0.63	0.64	0.65	0.64	0.64	0.65
Human Non-TATA Ensembl	0.88	0.88	0.86	0.87	0.88	0.83
Mouse Enhancers Ensembl	0.71	0.71	0.79	0.74	0.73	0.75
Nucleotide Transformer						
Dataset	Emb	+gcCont	+zCurve	+AT/GC	+cumSkew	+pseudoKNC
Human Ensembl Regulatory	0.57	0.57	0.55	0.57	0.57	0.58
Drosophila Enhancers Stark	0.66	0.67	0.67	0.66	0.66	0.65
Demo Coding vs Intergenomic	0.85	0.86	0.85	0.85	0.85	0.86
Demo Human vs Worm	0.90	0.88	0.87	0.88	0.87	0.89
Human Enhancers Cohn	0.66	0.68	0.68	0.68	0.68	0.68
Human Enhancers Ensembl	0.68	0.69	0.69	0.69	0.69	0.69
Human OCR Ensembl	0.60	0.61	0.61	0.61	0.61	0.62
Human Non-TATA Ensembl	0.80	0.81	0.80	0.80	0.81	0.81
Mouse Enhancers Ensembl	0.78	0.75	0.79	0.76	0.79	0.76
HyenaDNA						
Dataset	Emb	+gcCont	+zCurve	+AT/GC	+cumSkew	+pseudoKNC
Human Ensembl Regulatory	0.78	0.74	0.69	0.78	0.66	0.69
Drosophila Enhancers Stark	0.61	0.59	0.53	0.63	0.52	0.51
Demo Coding vs Intergenomic	0.83	0.78	0.78	0.83	0.78	0.83
Demo Human vs Worm	0.83	0.70	0.64	0.83	0.71	0.77
Human Enhancers Cohn	0.67	0.65	0.68	0.68	0.68	0.67
Human Enhancers Ensembl	0.70	0.67	0.64	0.72	0.63	0.73
Human OCR Ensembl	0.62	0.59	0.60	0.63	0.57	0.63
Human Non-TATA Ensembl	0.82	0.71	0.73	0.79	0.73	0.74
Mouse Enhancers Ensembl	0.73	0.76	0.73	0.73	0.77	0.73

nucleotide skews, and pseudo k-tuple nucleotide compositions (PseudoKNC). Each variant represents a hybrid model that integrates DNABERT-2 embeddings with one specific feature type. In contrast, the fine-tuned baseline directly updates the DNABERT-2 model weights on the downstream task through supervised fine-tuning.

We report both accuracy and inference time (including feature extraction) on a held-out independent test set to assess generalization and computational efficiency. Results on additional independent datasets are provided in Table 4, where accuracy and carbon emissions are denoted as Acc and CE, respectively. For fine-tuned models, the reported time includes preprocessing, fine-tuning, and inference.

5.3 Accuracy - Efficiency Trade-offs in Enhancer and Promoter Classification

Table 4 presents accuracy and inference-time results for two independent tasks, enhancer classification and non-TATA promoter classification comparing DNABERT-2 and Hyenadna embeddings (with and without feature augmentations) against a fully fine-tuned DNABERT-2 and Hyenadna model. Several key insights emerge as follows:

Table 4: Performance and Inference Time for Enhancer and Non-TATA Promoter Classification Using DNABERT-2 and HyenaDNA Based Methods

Task	Model Type	Model	Acc	Inference Time (s)	CE (kg)
Enhancer	DNA BERT-2	Embedding	0.65	525.74	0.02
		+ GC Content	0.65	+ 2.61	0.02
		+ zCurve	0.65	+ 2.49	0.02
		+ AT/GC Ratio	0.67	+ 2.24	0.02
		+ Cumulative Skew	0.67	+ 2.29	0.02
		+ PseudoKNC	0.67	+ 10.97	0.02
		Fine-tuned	0.61	31497.08	1.55
	Hyena DNA	Embedding	0.65	410.96	0.02
		+ GC Content	0.61	+ 2.05	0.02
		+ zCurve	0.68	+ 1.94	0.02
		+ AT/GC Ratio	0.65	+ 1.78	0.02
		+ Cumulative Skew	0.64	+ 1.82	0.02
		+ PseudoKNC	0.65	+10.11	0.02
		Fine-tuned	0.58	3394.15	0.17
Non-TATA Promoter	DNA BERT-2	Embedding	0.72	284.62	0.01
		+ GC Content	0.85	+ 31.25	0.02
		+ zCurve	0.85	+ 31.22	0.02
		+ AT/GC Ratio	0.85	+ 31.17	0.02
		+ Cumulative Skew	0.84	+ 31.22	0.02
		+ PseudoKNC	0.85	+ 84.30	0.02
		Fine-tuned	0.89	8886.68	0.44
	Hyena DNA	Embedding	0.84	125.41	0.01
		+ GC Content	0.66	+ 30.88	0.01
		+ zCurve	0.75	+ 30.68	0.01
		+ AT/GC Ratio	0.83	+ 31.63	0.01
		+ Cumulative Skew	0.72	+ 29.98	0.01
		+ PseudoKNC	0.79	+ 26.12	0.01
		Fine-tuned	0.77	940.51	0.04

5.3.1 Enhancer Classification

Embedding-only DNABERT-2 achieves 0.65 accuracy in 525.74 seconds, with carbon emissions of only 0.02 kg. Simple feature augmentations such as GC content, z-curve, AT/GC ratio, and cumulative skew yield modest improvements (up to 0.67 accuracy) for minimal computational overhead (2.24 - 2.61 seconds). For example, adding the AT/GC ratio boosts accuracy to 0.67 with just 2.24 additional seconds. Even the higher-dimensional PseudoKNC feature raises accuracy to 0.67 at the cost of 10.97 seconds. In contrast, full fine-tuning of DNABERT-2 leads to lower accuracy (0.61) and significantly greater compute time (74.90 seconds for pre-processing, 31411.48 seconds for fine-tuning and 10.70 seconds for inference), with 1.55 kg of carbon emission.

HyenaDNA embeddings perform comparably, achieving 0.65 accuracy in 410.96 seconds with 0.02 kg carbon emission. zCurve augmentation brings the highest accuracy (0.68) with only 1.94 seconds overhead. However, fine-tuning HyenaDNA yields lower performance (0.58 accuracy) despite 3394.15 (179.98 for pre-processing, 3204.06 for fine-tuning and 10.11 for inference) seconds of compute and 0.17 kg CO₂. This highlights the superiority of fixed embedding approaches with simple features in both efficiency and accuracy.

Non-TATA Promoter Classification. In the non-TATA promoter classification task, DNABERT-2 embeddings combined with handcrafted features offer significant reductions in inference time and carbon emission compared to the fine-tuned model. For instance, DNABERT-2 + AT/GC ratio achieves competitive performance with an inference time of only 315.79 seconds and 0.02 kg CO₂. In contrast, fine-tuned DNABERT-2 takes 8886.68 seconds and emits 0.44 kg CO₂.

On the other hand, the HyenaDNA embedding with AT/GC ratio achieved strong accuracy (0.83) while requiring only 157.04 seconds and emitting 0.01 kg CO₂. In contrast, the fine-tuned HyenaDNA model required 940.51 seconds and emitted 0.04 kg CO₂.

To further analyze the embedding space learned by the DNABERT-2 model, we applied Principal Component Analysis (PCA) to reduce the high-dimensional embeddings to two dimensions for visualization. We randomly sampled up to 500 positive and 500 negative sequences from both the training and test sets and projected their embeddings into a 2D space using PCA. The resulting plots Figure 2 and Figure 3 provide insight into how well the embeddings separate classes and generalize across datasets. Clear separation between positive and negative samples in both sets suggests that the model captures class-discriminative information, while the structural similarity between the training and test embeddings indicates good generalization performance.

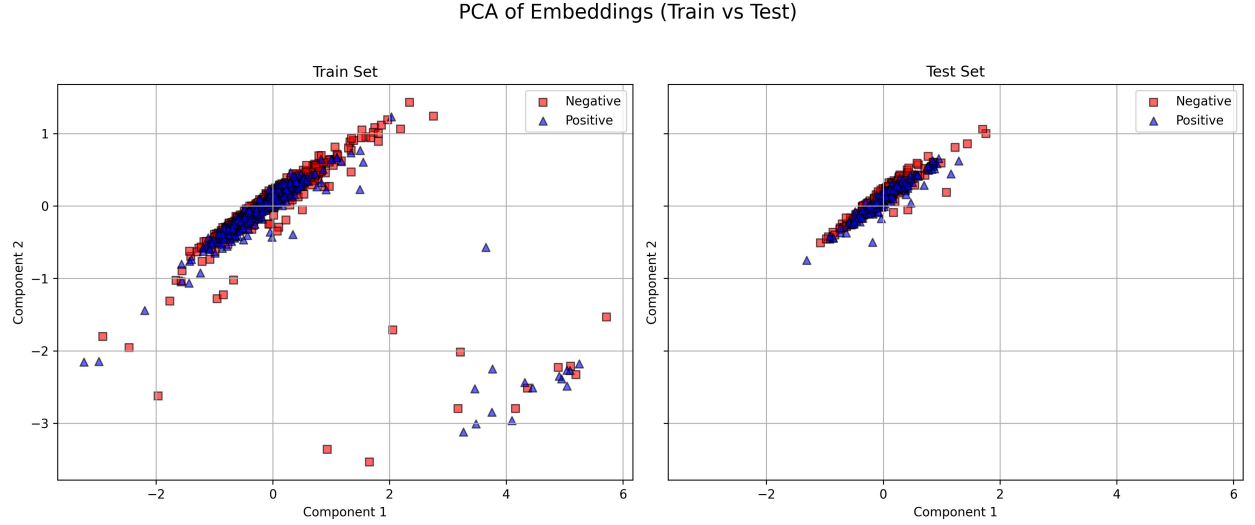


Figure 2: Comparison of dimensionality reduction methods applied to DNABERT-2 sequence embeddings for the enhancer classification task. Both methods show class separation and generalization between training and test sets.

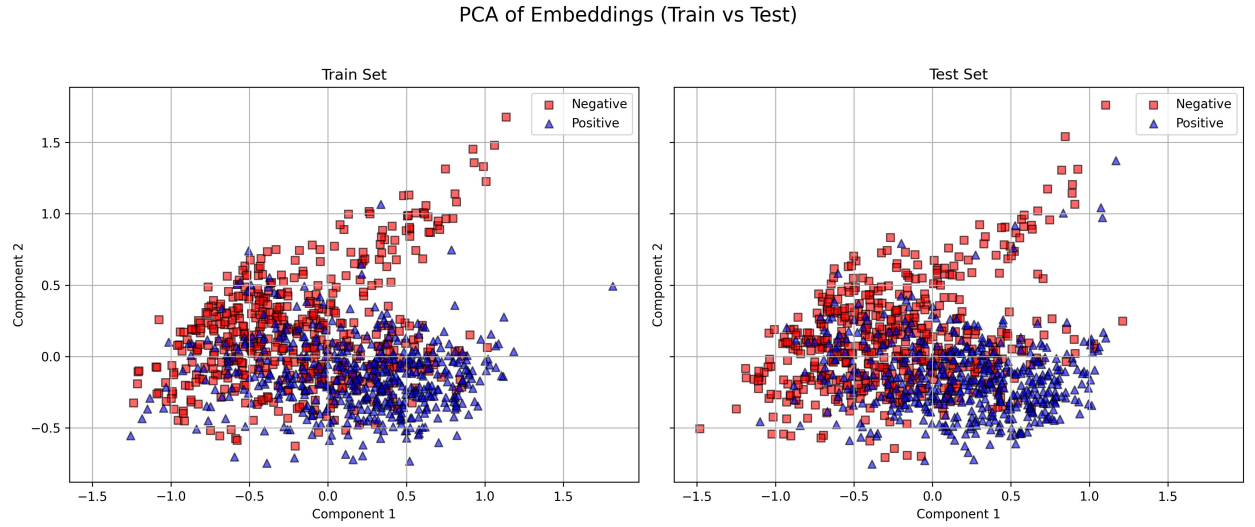


Figure 3: Comparison of dimensionality reduction methods applied to DNABERT-2 sequence embeddings for the promoter classification task. Both methods show class separation and generalization between training and test sets.

5.4 Carbon Footprint and Efficiency Analysis

To assess the sustainability and efficiency of the models used in our experiments, we evaluated not only the classification accuracy and the inference time but also estimated carbon emissions. Table 4 includes carbon emission results for both the Enhancer Classification and Non-TATA Promoter Classification tasks. Experiments were conducted using a private infrastructure, which has a carbon efficiency of 0.593 kgCO₂eq/kWh. Estimations were conducted using the MachineLearning Impact calculator presented in [18].

Notably, although fine-tuning DNABERT-2 leads to higher resource consumption and CO₂ emissions, the classification accuracy does not consistently improve over the embedding-based approaches with auxiliary features. This insight emphasizes the trade-off between accuracy, computational cost, and environmental impact, advocating for green AI practices in bioinformatics model development.

6 Discussion

6.1 Performance vs. Carbon Efficiency

We conducted a comprehensive evaluation of DNABERT-2 and its variants by augmenting the base embeddings with biologically inspired handcrafted features, and compared these against fine-tuned DNABERT-2 models across two downstream tasks: Enhancer Classification and Non-TATA Promoter Classification.

Embedding vs. Fine-Tuning. As shown in Table 4, embedding-based approaches closely match or outperform fine-tuned models with significantly lower inference time and carbon emissions. For enhancer classification, DNABERT-2 with zCurve achieved an accuracy of 0.67 (vs. 0.61 for fine-tuning), while reducing emissions from 1.55 kg to 0.02 kg CO₂ and inference time from over 31,000 s to 528 s. HyenaDNA with zCurve slightly improved accuracy to 0.68 and brought inference time to just 412.9 s and emissions to 0.02 kg.

In the non-TATA promoter classification task, DNABERT-2 with zCurve achieved 0.85 accuracy close to the fine-tuned model’s 0.89 while cutting emissions from 0.44 kg to 0.02 kg CO₂ and runtime from 8,886s to 316s. HyenaDNA’s zCurve variant reached 0.75 accuracy with just 156s of total inference time and 0.01kg CO₂, offering high efficiency at modest accuracy trade-offs.

6.2 Impact of Handcrafted Features

Augmenting transformer embeddings with handcrafted features improved classification in most cases. For enhancer prediction, adding zCurve to DNABERT-2 increased accuracy from 0.65 to 0.67, and from 0.65 to 0.68 with HyenaDNA. Features like AT/GC ratio and cumulative skew also yielded moderate gains. However, PseudoKNC, despite being computationally intensive, offered no significant accuracy advantage, suggesting limited benefits from high-dimensional features in this context.

6.3 Generalization Across Datasets

Table 3 demonstrates the generalization capabilities of embedding-based models over nine genomic benchmarks. DNABERT-2 embeddings combined with statistical features reached top-1% accuracies up to 0.91 (Human vs. Worm) and showed consistent improvement on challenging datasets like Mouse Enhancers and Human OCR Ensembl. Importantly, these results were obtained without fine-tuning, confirming the utility of pre-trained DNA embeddings for downstream tasks.

6.4 Environmental Implications

The observed up to 77.5× reduction in CO₂ emissions makes embedding-based workflows far more sustainable and aligns with Green AI principles. Fine-tuning large genomic models requires substantial compute and energy, limiting practical deployment. In contrast, our embedding-based pipeline offers a scalable, low-emission alternative for real-world applications where environmental cost is a concern. These observations align with *Green AI* principles, advocating for environmentally sustainable methods that balance accuracy with compute cost [11, 7].

7 Conclusion

In this study, we investigated the efficacy and efficiency of leveraging embeddings combined with biologically inspired handcrafted features for genomic sequence classification. Our results demonstrate that embedding-based approaches can achieve comparable or superior performance to fine-tuned transformer models, while drastically reducing inference time and carbon emissions achieving up to 77.5 \times lower CO₂ emissions in certain scenarios.

Despite these benefits, our approach has some limitations. Handcrafted features are fixed and may not capture task-specific contextual nuances, and static pretrained embeddings lack adaptability during training. However, this deliberate design enables significant reductions in computational cost and carbon footprint, without compromising accuracy aligning with the goals of sustainable and efficient AI research in genomics.

For future work, we plan to investigate hybrid models that integrate task-specific fine-tuning with fixed embeddings to balance adaptability and efficiency. We also aim to extend this framework to more complex regulatory genomics tasks, including enhancer-promoter interaction prediction and multi-label regulatory element classification.

References

- [1] Daria Shlyueva, Gerald Stampfel, and Alexander Stark. Transcriptional enhancers: from properties to genome-wide predictions. *Nature Reviews Genetics*, 15(4):272–286, 2014.
- [2] Yang Zhang, Lorenzo Boninsegna, Muyu Yang, Tom Misteli, Frank Alber, and Jian Ma. Computational methods for analysing multiscale 3d genome organization. *Nature Reviews Genetics*, 25(2):123–141, 2024.
- [3] 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.
- [4] Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P de Almeida, Hassan Sirelkhatim, et al. Nucleotide transformer: building and evaluating robust foundation models for human genomics. *Nature Methods*, 22(2):287–297, 2025.
- [5] Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in neural information processing systems*, 36:43177–43201, 2023.
- [6] Emma Strubell, Ananya Ganesh, and Andrew McCallum. Energy and policy considerations for modern deep learning research. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 13693–13696, 2020.
- [7] David Patterson, Joseph Gonzalez, Quoc Le, Chen Liang, Lluís-Miquel Munguia, Daniel Rothchild, David So, Maud Texier, and Jeff Dean. Carbon emissions and large neural network training. *arXiv preprint arXiv:2104.10350*, 2021.
- [8] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in neural information processing systems*, 33:9459–9474, 2020.
- [9] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019.
- [10] Xinyi Lin, Gelei Deng, Yuekang Li, Jingquan Ge, Joshua Wing Kei Ho, and Yi Liu. Generag: Enhancing large language models with gene-related task by retrieval-augmented generation. *bioRxiv*, pages 2024–06, 2024.
- [11] Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.
- [12] 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.
- [13] Rafsanjani Muhammod, Sajid Ahmed, Dewan Md Farid, Swakkhar Shatabda, Alok Sharma, and Abdollah Dehzangi. Pyfeat: a python-based effective feature generation tool for dna, rna and protein sequences. *Bioinformatics*, 35(19):3831–3833, 2019.
- [14] Zicheng Liu, Jiahui Li, Siyuan Li, Zelin Zang, Cheng Tan, Yufei Huang, Yajing Bai, and Stan Z Li. Genbench: A benchmarking suite for systematic evaluation of genomic foundation models. *arXiv preprint arXiv:2406.01627*, 2024.

- [15] 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.
- [16] Quang H Nguyen, Thanh-Hoang Nguyen-Vo, Nguyen Quoc Khanh Le, Trang TT Do, Susanto Rahardja, and Binh P Nguyen. ienhancer-ecnn: identifying enhancers and their strength using ensembles of convolutional neural networks. *BMC genomics*, 20:1–10, 2019.
- [17] Ramzan Kh Umarov and Victor V Solovyev. Recognition of prokaryotic and eukaryotic promoters using convolutional deep learning neural networks. *PloS one*, 12(2):e0171410, 2017.
- [18] Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*, 2019.

8 Appendix

8.0.1 Handcrafted Feature Details

1. **z-Curve Representation.** The z-Curve is a three-dimensional curve that uniquely maps a nucleotide sequence based on nucleotide disparity in three orthogonal directions-

$$x = (A + G) - (C + T), \quad (3)$$

$$y = (A + C) - (G + T) \text{ and}, \quad (4)$$

$$z = (A + T) - (G + C), \quad (5)$$

For rest of the sections, A , T , G , and C are the total counts of each base in the sequence. These components capture purine-pyrimidine imbalance (x), amino-keto group content (y), and weak-strong hydrogen bonding asymmetry (z).

2. **GC Content.** GC content reflects the proportion of guanine and cytosine in the sequence and is defined as:

$$\text{GC content} = \frac{G + C}{A + T + G + C} \times 100\%. \quad (6)$$

3. **AT/GC Ratio.** The AT/GC ratio indicates compositional bias and is computed as:

$$\text{AT/GC ratio} = \frac{A + T}{G + C}. \quad (7)$$

4. **Cumulative Skews.** Nucleotide skews are widely used to detect strand asymmetries. We compute:

$$\text{GC skew} = \frac{G - C}{G + C}, \quad \text{AT skew} = \frac{A - T}{A + T}$$

5. **Pseudo k-tuple Nucleotide Composition (PseKNC).** To capture both the local composition and sequence-order information, we use PseKNC. For a fixed k (typically $k = 3$), all possible k -mers from the nucleotide alphabet $\{A, C, G, T\}$ are enumerated. The number of possible k -mers $\# \text{ k-mers} = 4^k$.

8.1 Algorithm

Algorithm 1 Retrieval-Augmented Genome Classification (Transformer-Agnostic)

Require:Training set $\mathcal{D}_{\text{train}} = \{(\mathbf{x}_i, y_i)\}_{i=1}^N$,1: Test set $\mathcal{D}_{\text{test}} = \{\mathbf{x}_j\}_{j=1}^M$,2: Pretrained transformer encoder f_θ ,3: Number of neighbors k .**Ensure:** Predicted labels $\{\hat{y}_j\}_{j=1}^M$ and evaluation metrics.4: **Embedding Phase (offline):**5: **for all** $(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{train}}$ **do**

▷ Compute embedding for each training sequence

6: $\mathbf{e}_i \leftarrow \text{EMBEDSEQUENCE}(f_\theta, \mathbf{x}_i)$

▷ e.g., mean-pool transformer outputs

7: **end for**8: Form matrix $E_{\text{train}} \in \mathbb{R}^{N \times D}$ with rows \mathbf{e}_i .9: **Index Construction:**10: Build FAISS L_2 index $\mathcal{I} \leftarrow \text{IndexFlatL2}(D)$.11: $\mathcal{I}.\text{add}(E_{\text{train}})$

▷ Add all training embeddings

12: **Inference Phase (online):**13: Start timer $t_{\text{start}} \leftarrow \text{now}()$.14: **for all** $\mathbf{x}_j \in \mathcal{D}_{\text{test}}$ **do**

▷ Embed each test sequence

15: $\mathbf{e}_j \leftarrow \text{EMBEDSEQUENCE}(f_\theta, \mathbf{x}_j)$ ▷ Obtain D -dimensional vector16: $[-, i_{j,1}, \dots, i_{j,k}] \leftarrow \mathcal{I}.\text{search}(\mathbf{e}_j, k)$ ▷ Retrieve indices of k nearest neighbors17: $\{y_{i_{j,1}}, \dots, y_{i_{j,k}}\} \leftarrow$ labels of retrieved neighbors

▷ Lookup ground-truth labels in training set

18: $\hat{y}_j \leftarrow \arg \max_c \sum_{m=1}^k \mathbf{1}(y_{i_{j,m}} = c)$

▷ Majority-voting across neighbor labels

19: **end for**20: Stop timer $t_{\text{end}} \leftarrow \text{now}()$.21: **Evaluation:**22: Compute accuracy, F1, and other metrics comparing $\{\hat{y}_j\}$ to true labels.23: Report wall-clock time: $\Delta t = t_{\text{end}} - t_{\text{start}}$.24: **function** $\text{EMBEDSEQUENCE}(f_\theta, \mathbf{x})$ ▷ Return D -dimensional mean-pooled embedding25: $\text{tokens} \leftarrow \text{Tokenize}(\mathbf{x})$ 26: $\text{hidden} \leftarrow f_\theta(\text{tokens})$ ▷ Output shape: (L, D) for L tokens27: **return** $\frac{1}{L} \sum_{t=1}^L \text{hidden}_t$ 28: **end function**

Algorithm 1 presents a transformer-agnostic retrieval-augmented classification framework for genomic sequences.

In the **embedding phase** (performed offline), each training sequence \mathbf{x}_i is processed by a pretrained transformer encoder f_θ to extract a fixed-length embedding vector \mathbf{e}_i , obtained by mean pooling over the transformer’s token-level outputs. These embeddings are collected into a matrix E_{train} .

Next, an efficient nearest neighbor index is built over the training embeddings using FAISS with an L_2 distance metric to enable fast similarity search.

During the **inference phase** (online), each test sequence \mathbf{x}_j is similarly embedded into vector \mathbf{e}_j . The FAISS index is queried to retrieve the indices of the k nearest training neighbors to \mathbf{e}_j . The predicted label \hat{y}_j for the test sequence is obtained by majority voting over the labels of the retrieved neighbors.

Finally, standard classification metrics such as accuracy and F1-score are computed by comparing the predicted labels $\{\hat{y}_j\}$ to the true test labels. The total inference runtime is also recorded to evaluate computational efficiency.

This approach enables flexible integration with any pretrained transformer encoder and leverages similarity search to perform non-parametric classification based on learned embeddings.

Algorithm 2 Part 1: Feature Extraction and Hybrid Embedding Construction**Require:** Training set $\mathcal{D}_{\text{train}}$, Test set $\mathcal{D}_{\text{test}}$, embedding vectors $\{\mathbf{e}_i\}$, $\{\mathbf{e}_j\}$, feature functions \mathcal{F} **Ensure:** Hybrid embeddings for train/test: $\{\mathbf{h}_{i,\text{train}}\}$, $\{\mathbf{h}_{j,\text{test}}\}$

```

1: for all  $(\mathbf{x}_i, y_i) \in \mathcal{D}_{\text{train}}$  do
2:   Initialize  $\mathbf{h}_{i,\text{feat}} \leftarrow []$ 
3:   for all  $f \in \mathcal{F}$  do
4:      $v \leftarrow f(\mathbf{x}_i)$ 
5:     if  $v$  scalar then
6:        $v \leftarrow [v]$ 
7:     end if
8:     Append  $v$  to  $\mathbf{h}_{i,\text{feat}}$ 
9:   end for
10:   $\mathbf{h}_{i,\text{feat}} \leftarrow \text{MinMaxScale}(\mathbf{h}_{i,\text{feat}})$ 
11:   $\tilde{\mathbf{e}}_i \leftarrow \mathbf{e}_i / \|\mathbf{e}_i\|$ 
12:   $\mathbf{h}_{i,\text{train}} \leftarrow [\alpha \tilde{\mathbf{e}}_i \parallel \beta \mathbf{h}_{i,\text{feat}}]$ 
13: end for
14: for all  $\mathbf{x}_j \in \mathcal{D}_{\text{test}}$  do
15:   Compute  $\mathbf{h}_{j,\text{feat}}$  similarly
16:    $\tilde{\mathbf{e}}_j \leftarrow \mathbf{e}_j / \|\mathbf{e}_j\|$ 
17:    $\mathbf{h}_{j,\text{test}} \leftarrow [\alpha \tilde{\mathbf{e}}_j \parallel \beta \mathbf{h}_{j,\text{feat}}]$ 
18: end for

```

Algorithm 3 Part 2: FAISS Indexing and Dynamic k-NN Retrieval**Require:** Hybrid embeddings $\{\mathbf{h}_{i,\text{train}}\}$, $\{\mathbf{h}_{j,\text{test}}\}$, base k_{base} , max factor α_{max} , threshold τ **Ensure:** Candidate neighbors and similarities for each test point

```

1: Build FAISS index  $\mathcal{I} \leftarrow \text{IndexFlatIP}(\cdot)$ 
2:  $\mathcal{I}.\text{add}(\{\mathbf{h}_{i,\text{train}}\})$ 
3:  $k_{\text{max}} \leftarrow \min(\alpha_{\text{max}} \cdot k_{\text{base}}, N)$ 
4:  $(D, I) \leftarrow \mathcal{I}.\text{search}(\{\mathbf{h}_{j,\text{test}}\}, k_{\text{max}})$ 
5: for  $j = 1$  to  $M$  do
6:    $\mu_j \leftarrow \frac{1}{k_{\text{base}}} \sum_{m=1}^{k_{\text{base}}} D_{j,m}$ 
7:   if  $\mu_j < \tau$  then
8:     Use  $k_{\text{max}}$  neighbors:  $D_j \leftarrow D_{j,1:k_{\text{max}}}$ ,  $I_j \leftarrow I_{j,1:k_{\text{max}}}$ 
9:   else
10:    Use  $k_{\text{base}}$  neighbors:  $D_j \leftarrow D_{j,1:k_{\text{base}}}$ ,  $I_j \leftarrow I_{j,1:k_{\text{base}}}$ 
11:   end if
12: end for

```

Algorithm 4 Part 3: Weighted Voting and Final Prediction**Require:** Neighbor sets $\{I_j\}$, distances $\{D_j\}$, class labels of training set**Ensure:** Predicted labels $\{\hat{y}_j\}$

```

1: for  $j = 1$  to  $M$  do
2:   Initialize vote vector  $\mathbf{v}_j \leftarrow \mathbf{0} \in \mathbb{R}^C$ 
3:   for  $m = 1$  to  $|I_j|$  do
4:      $w_{j,m} \leftarrow \frac{1}{1 + e^{-D_{j,m}}}$ 
5:      $c \leftarrow$  class label of training index  $I_j[m]$ 
6:      $\mathbf{v}_j[c] \leftarrow \mathbf{v}_j[c] + w_{j,m}$ 
7:   end for
8:    $\hat{y}_j \leftarrow \arg \max_c \mathbf{v}_j[c]$ 
9: end for
10: Compute accuracy, F1-score, and runtime

```

▷ Sigmoid weighting

The overall classification procedure along with handcrafted features is divided into three main stages, described in Algorithms 2, 3, and 4.

Algorithm 2 outlines the *Feature Extraction and Hybrid Embedding Construction* step. Here, handcrafted features are computed from input sequences using a predefined set of functions, scaled via min-max normalization, and concatenated with normalized pretrained embeddings to form hybrid feature vectors for both training and test samples.

Algorithm 3 describes the *FAISS Indexing and Dynamic k-NN Retrieval* phase. We build an approximate nearest neighbor index using the training hybrid embeddings and perform batched retrieval for each test embedding. The number of neighbors considered is dynamically adjusted based on a similarity threshold, enabling adaptive neighbor selection.

Finally, **Algorithm 4** presents the *Weighted Voting and Final Prediction* step. Retrieved neighbors vote for the class label of each test sample, with weights computed by applying a sigmoid function to similarity scores. The predicted label corresponds to the class with the highest cumulative weighted vote. Performance metrics such as accuracy and F1-score are then computed to evaluate classification effectiveness.

Detailed Classification Reports on Genomic Benchmark Datasets (Embedding only)

DNABERT-2

Coding vs Intergenomic Seqs

Class	Precision	Recall	F1-score	Support
0	0.81	0.93	0.86	12500
1	0.91	0.78	0.84	12500
Accuracy			0.85	25000

Human or Worm

Class	Precision	Recall	F1-score	Support
0	0.95	0.87	0.91	12500
1	0.88	0.96	0.92	12500
Accuracy			0.91	25000

Drosophila Enhancers Stark

Class	Precision	Recall	F1-score	Support
0	0.73	0.63	0.67	865
1	0.67	0.77	0.72	865
Accuracy			0.70	1730

Mouse Enhancers Ensembl

Class	Precision	Recall	F1-score	Support
0	0.77	0.61	0.68	121
1	0.68	0.82	0.74	121
Accuracy			0.71	242

Human Enhancers Cohn

Class	Precision	Recall	F1-score	Support
0	0.71	0.68	0.70	3474
1	0.69	0.72	0.71	3474
Accuracy			0.70	6948

Human Enhancers Ensembl

Class	Precision	Recall	F1-score	Support
0	0.75	0.65	0.69	15485
1	0.69	0.78	0.73	15485
Accuracy			0.71	30970

Human Ensembl Regulatory

Class	Precision	Recall	F1-score	Support
0	0.50	0.50	0.50	21378
1	0.50	0.60	0.54	17476
2	0.81	0.67	0.73	18859
Accuracy			0.58	57713

Human OCR Ensembl

Class	Precision	Recall	F1-score	Support
0	0.65	0.59	0.62	17476
1	0.62	0.68	0.65	17476
Accuracy			0.63	34952

Human Non-TATA Promoters

Class	Precision	Recall	F1-score	Support
0	0.80	0.99	0.88	4119
1	0.99	0.79	0.88	4915
Accuracy			0.88	9034

HyenaDNA**Coding vs Intergenic Seqs**

Class	Precision	Recall	F1-score	Support
0	0.81	0.85	0.83	12500
1	0.84	0.80	0.82	12500
Accuracy			0.83	25000

Human or Worm

Class	Precision	Recall	F1-score	Support
0	0.82	0.84	0.83	12500
1	0.84	0.82	0.83	12500
Accuracy			0.83	25000

Drosophila Enhancers Stark

Class	Precision	Recall	F1-score	Support
0	0.63	0.52	0.57	865
1	0.59	0.70	0.64	865
Accuracy			0.61	1730

Mouse Enhancers Ensembl

Class	Precision	Recall	F1-score	Support
0	0.74	0.72	0.73	121
1	0.73	0.74	0.73	121
Accuracy			0.73	242

Human Enhancers Ensembl

Class	Precision	Recall	F1-score	Support
0	0.72	0.66	0.69	15485
1	0.69	0.75	0.72	15485
Accuracy			0.70	30970

Human Enhancers Cohn

Class	Precision	Recall	F1-score	Support
0	0.68	0.65	0.66	3474
1	0.66	0.69	0.68	3474
Accuracy			0.67	6948

Human Ensembl Regulatory

Class	Precision	Recall	F1-score	Support
0	0.69	0.86	0.77	21378
1	0.87	0.72	0.79	17476
2	0.84	0.73	0.78	18859
Accuracy			0.78	57713

Human OCR Ensembl

Class	Precision	Recall	F1-score	Support
0	0.63	0.58	0.60	17476
1	0.61	0.66	0.63	17476
Accuracy			0.62	34952

Human Non-TATA Promoters

Class	Precision	Recall	F1-score	Support
0	0.75	0.89	0.81	4119
1	0.89	0.76	0.82	4915
Accuracy			0.82	9034

Nucleotide Transformer**Human or Worm**

Class	Precision	Recall	F1-score	Support
0	0.89	0.92	0.90	12500
1	0.92	0.88	0.90	12500
Accuracy			0.90	25000

Coding vs Intergenic Seqs

Class	Precision	Recall	F1-score	Support
0	0.90	0.80	0.85	12500
1	0.82	0.91	0.86	12500
Accuracy			0.85	25000

Drosophila Enhancers Stark

Class	Precision	Recall	F1-score	Support
0	0.70	0.58	0.64	865
1	0.64	0.75	0.69	865
Accuracy			0.66	1730

Mouse Enhancers Ensembl

Class	Precision	Recall	F1-score	Support
0	0.77	0.81	0.79	121
1	0.80	0.75	0.77	121
Accuracy			0.78	242

Human Enhancers Cohn

Class	Precision	Recall	F1-score	Support
0	0.66	0.66	0.66	3474
1	0.66	0.66	0.66	3474
Accuracy			0.66	6948

Human Ensembl Regulatory

Class	Precision	Recall	F1-score	Support
0	0.49	0.59	0.54	21378
1	0.51	0.37	0.43	17476
2	0.69	0.72	0.71	18859
Accuracy			0.57	57713

Human Non-TATA Promoters

Class	Precision	Recall	F1-score	Support
0	0.75	0.86	0.80	4119
1	0.86	0.75	0.81	4915
Accuracy			0.80	9034

Human Enhancers Ensembl

Class	Precision	Recall	F1-score	Support
0	0.70	0.63	0.66	15485
1	0.66	0.74	0.70	15485
Accuracy			0.68	30970

Human OCR Ensembl

Class	Precision	Recall	F1-score	Support
0	0.60	0.57	0.58	17476
1	0.59	0.63	0.61	17476
Accuracy			0.60	34952

Classification Reports on New Independent Test Set

Enhancer Classification

HyenaDNA (Embedding only)

Class	Precision	Recall	F1-score	Support
0	0.73	0.47	0.57	200
1	0.61	0.82	0.70	200
Accuracy			0.65	400

DNABERT-2 (Embedding only)

Class	Precision	Recall	F1-score	Support
0	0.65	0.66	0.65	200
1	0.65	0.64	0.65	200
Accuracy			0.65	400

HyenaDNA (Fine-tuned)

Class	Precision	Recall	F1-score	Support
0	0.56	0.77	0.65	200
1	0.63	0.40	0.49	200
Accuracy			0.58	400

DNABERT-2 (Fine-tuned)

Class	Precision	Recall	F1-score	Support
0	0.57	0.87	0.69	200
1	0.73	0.34	0.47	200
Accuracy			0.61	400

Promoter Classification**HyenaDNA (Embedding only)**

Class	Precision	Recall	F1-score	Support
0	0.86	0.87	0.87	27731
1	0.82	0.80	0.81	19811
Accuracy			0.84	47542

DNABERT-2 (Embedding only)

Class	Precision	Recall	F1-score	Support
0	0.95	0.54	0.69	27731
1	0.60	0.96	0.74	19811
Accuracy			0.72	47542

DNABERT-2 (Fine-tuned)

Class	Precision	Recall	F1-score	Support
0	0.92	0.88	0.90	27731
1	0.85	0.90	0.87	19811
Accuracy			0.89	47542

HyenaDNA (Fine-tuned)

Class	Precision	Recall	F1-score	Support
0	0.80	0.82	0.81	27731
1	0.73	0.71	0.72	19811
Accuracy			0.77	47542