

GENEMASK: Fast Pretraining of Gene Sequences to Enable Few-Shot Learning

Soumyadeep Roy^{a,*}, Jonas Wallat^b, Sowmya S Sundaram^b, Wolfgang Nejdl^b and Niloy Ganguly^a

^aIndian Institute of Technology Kharagpur

^bL3S Research Center, Germany

Abstract. Large-scale language models such as DNABert and LOGO aim to learn optimal gene representations and are trained on the entire *Human Reference Genome*. However, standard tokenization schemes involve a simple sliding window of tokens like *k-mers* that do not leverage any gene-based semantics and thus may lead to (trivial) masking of easily predictable sequences, and subsequently inefficient Masked Language Modeling (MLM) training. Therefore, we propose a novel masking algorithm, GENEMASK, for MLM training of gene sequences, where we randomly identify positions in a gene sequence as mask centers and locally select the span around the mask center with the highest Normalized Pointwise Mutual Information (NPMI) to mask. We observe that in the absence of human-understandable semantics in the genomics domain (in contrast, semantic units like words and phrases are inherently available in NLP), GENEMASK-based models substantially outperform the SOTA models (DNABert and LOGO) over four benchmark gene sequence classification datasets in five few-shot settings (10 to 1000-shot). More significantly, the GENEMASK-based DNABert model is trained for less than one-tenth of the number of epochs of the original SOTA model. We also observe a strong correlation between top-ranked PMI tokens and *conserved DNA sequence motifs*, which may indicate the incorporation of latent genomic information. The codes (including trained models) and datasets are made publicly available at <https://github.com/roysoumya/GeneMask>.

1 Introduction

Computational analysis of genomics has revolutionized the field of medical science [23], particularly with the advent of the *Human Reference Genome* [35]. As seen in recent studies [1, 45], deep learning has been applied to various genomic applications, such as protein structure analysis, gene expression data, and transcriptome analysis. The state-of-the-art pretrained models (DNABert [16] and LOGO [44]) in gene sequence classification tasks are widely used in literature [2, 24]. The input data for these tasks are often presented as a sequence of nucleotides. Each side of the double-helix DNA strand comprises the bases adenine (A), cytosine (C), guanine (G), and thymine (T). Similar to the NLP domain, the standard approach in gene sequence classification is to pretrain the transformer models by randomly masking (and predicting) tokens, the so-called masked language modeling objective (MLM). However, unlike words or sentences in languages, no clear semantically demarcated tokens are

present within the gene sequence. Therefore, to come to a workable solution, researchers randomly select a sequence of k nucleotides [16, 24]; for example, if $k = 3$, the sequence *TCG* can be selected from a gene sequence $\dots GATTCGATGC \dots$. However, the workable solution may be easy to guess if (say) *GATTCG* is a very common sequence because *GAT* is still unmasked and frequently co-occurs with the masked *TCG* sequence. This, in turn, may decelerate the training process and increase the pretraining time; for example, DNABert trains for 25 days on 8 NVIDIA 2080Ti GPUs. A possible solution to reduce the instances of ‘easy’ learning is to identify highly-correlated commonly occurring spans and mask them in their entirety so that the pretraining model does not consume precious computing cycles in predicting ‘easy’ cases.

A systematic approach to identify correlated spans can be based on the principle of *Pointwise Mutual Information* (PMI), which calculates the chance of a set of tokens spanning together vis-a-vis occurring independently; a high PMI score indicates a high correlation. In this work, we propose a novel masking algorithm, GENEMASK, for MLM training of gene sequences. We randomly identify positions in a gene sequence as mask centers and locally select the span around the mask center with the highest (modified) PMI to mask. However, PMI also favors rarely occurring correlated spans (where individual tokens may have a low frequency of occurrence), which are not easy cases as the model has not frequently seen such patterns. Hence we modify the PMI metric and impose a discounting factor to down-score rare correlated spans. Instead of randomly selecting tokens to mask, as done in DNABert (or LOGO), we use the PMI score to prioritize the *relevant* tokens within a given gene sequence. In the case of GENEMASK, we randomly chose positions in a gene sequence and locally chose the span with the highest PMI to mask. This is in contrast to the strategy adopted in NLP literature by Levine et al. [21], where they use the PMI score to formulate an absolute importance notion and consequently create a fixed masking vocabulary.

As current medical datasets often face data scarcity issues, the move towards personalized medicine requires models that perform well with limited training data at hand [37, 14]. We, therefore, evaluate our proposed GENEMASK-based MLM training strategy in the low-resource (few-shot) setting.

Our extensive experimentation shows that GENEMASK-based DNABert and LOGO improve over the standard random masking-guided DNABert and LOGO, respectively, in **few-shot settings** (10, 50, 100, 500 and 1000 training data points per class) over four benchmark datasets of gene sequence classification (two on promoters - *Prom-core*, *Prom-300*, one on enhancer - *Cohn-enh* and one on

* Corresponding Author. Email: soumyadeep.roy9@iitkgp.ac.in

splice sites - *Splice-40*). We posit that GENEMASK helps incorporate non-trivial genetic knowledge because we observe that GENEMASK-based DNABert pretrained for **10K steps even outperforms original DNABert pretrained for 120K steps** for all few-shot settings in case of Prom-300 and Cohn-enh dataset. In addition, we perform **motif¹ analysis** and observe a strong correlation between top-ranked PMI tokens and conserved DNA sequence motifs, providing a biological reason behind the performance improvement in GENEMASK-based DNABert. Finally, to alleviate the issue of the tremendous engineering effort needed to develop the experimental setup of gene sequence classification tasks, we make all the codes (including trained models), data, and appendix publicly available at <https://github.com/roysoumya/GeneMask>.

2 Background

Learning deep representations in the context of gene sequence modeling. Recent studies [27, 16, 44] represent gene sequences as k-mers, using a learned dense representation from an adapted BERT model. Mock et al. [25] broadly adopt the DNABert architecture for the task of taxonomy classification and model gene sequences as 3-mers and also include next sentence prediction along with MLM training loss. Instead of using k-mer representations, the *BigBird* model [46] trained a SentencePiece tokenizer on the Human Reference Genome and applied to the tasks of chromatin-profile prediction and promoter region prediction. However, the pretrained model weights of the *BigBird* model for the genomics setting are not publicly available, and the alternative of training from scratch is prohibitively costly for such a large model. Mo et al. [24] infuse domain knowledge into the model by proposing a multimodal pretraining setup comprising gene sequences and information on transcription factors and regions. The recent Enformer [1] utilizes a combination of convolutional and transformer layers to feed long sequences of one-hot encoded base pairs into the model. However, unlike gene sequence classification, which is the focus of this work, the Enformer model is used to predict gene expression tracks.

Random and PMI-masking for MLM training in NLP. The initial work applied *random token masking*, as performed by BERT [7], where 15% of the input tokens are chosen to be masked uniformly. Previous work has also investigated simultaneous masking of sequences of adjacent tokens which form either a whole word (*whole word masking* [36]) or an entity (*entity masking* [39]); the technique proves to be beneficial over randomly masking single, non-contiguous tokens. Joshi et al. [17] propose *random span masking* where random spans with lengths chosen from a geometric distribution are masked at random positions. This simple method outperforms the more involved entity masking approach [39]. When applied to gene sequences, especially entity and whole word masking, have the problem that there is no well-defined concept of entities or words in gene sequences. The *PMI-masking* approach [21] builds on the ideas of the entity and span masking where they treat collocated n-grams with high Pointwise Mutual Information scores analogous to an entity and thereby mask these spans of tokens together. The authors show that masking PMI tokens (i) accelerates training while matching end-of-pretraining performance in roughly half the training steps and (ii) improves upon previous masking approaches at the end of pretraining. Sadeq et al. [32] develop the *InforMask* masking strategy, where they measure the *informative relevance* of a word as the sum of PMI values between a masked word and all unmasked

words in the given sentence (higher values are prioritized for masking). However, the semantic equivalent of a sentence (in NLP) is not known in the case of gene sequence modeling.

3 Building blocks of SOTA models

We focus on the two SOTA transformer-based pretrained models named DNABert [16] and LOGO [44] that are adapted to the gene sequence modeling domain. Through MLM pretraining, these models learn powerful contextual representations for DNA fragments utilizing abundant unlabeled data from the *Human Reference Genome*, which contains around 3.2 billion base pairs over 24 chromosomes.

Implementation details of SOTA models and associated research challenges. The tokenization of gene sequences and MLM training is performed based on the author’s codebase [9]. However, they do not provide the dataset for pretraining or finetuning (downstream) tasks. Therefore, we follow the author’s description to construct the corresponding datasets, which is nontrivial. We explain the pretraining data creation process in this section and later describe the finetuning dataset creation process in Section 5.1.

Tokenization of gene sequences. The gene sequence is first converted into a k-mer representation, which is commonly used in the literature [16, 27]. The *k*-mer representation is a sliding window of length *k*. For example, *AGCACGCAG* in 6-mer representation leads to 3 tokens - *AGCACG*, *GCACGA*, *CACGAG*. The vocabulary comprises all combinations (4^k length) and five special tokens - *CLS*, *PAD*, *UNK*, *SEP*, *MASK*. According to the set-up chosen by SOTA models, we consider $k = 6$ for all the experiments, as it also provides a good trade-off between longer contextual information and manageable computational complexity [44]. As stated in the LOGO paper [44], 6-mers incorporate richer contextual information while keeping the memory and computational complexity manageable.

Pretraining data preparation. We obtain the Human Reference Genome from the Genome Reference Consortium Human Build 38 patch release 13 (GRCh38.p13) FASTA file [6] from the NCBI website. It serves as a large-scale corpus of unlabeled gene sequence data, which we use for MLM training to obtain a contextual representation of the 6-mer tokens. We perform the following steps to convert the Human Reference Genome to a form that DNABert (or LOGO) can use to train with the MLM objective:

- (a). For each chromosome *c* in the Human Reference Genome, we randomly choose the starting index between 1 and 1000 [46].
- (b). Given the chromosome number and its starting index (*ST*), we next determine the length of the DNA segment *L* as BERT has the limitation of accommodating a maximum of 512 tokens. We select *L* as 510 for 50% of the cases and a randomly selected length between 5 and 510 for the remaining 50% of cases [16].
- (c). We thus create a DNA segment comprising the base pairs between *ST* and *ST* + *L* of chromosome *c*, corresponding to *data point* in the *pretraining dataset*. We filter out DNA segments that contain bases other than A, T, C, or G.

MLM training. The SOTA models are trained with masked language modeling loss similar to BERT [7]. However, to mask a nucleotide, a contiguous sequence of tokens is masked to prevent information leakage, as each nucleotide is part of *k* consecutive k-mers. More formally, say a nucleotide is represented as $\text{DNA}[i]$, while a 6-mer token is represented as $\text{T}[i]$, equivalent to $\{\text{DNA}[i-2], \text{DNA}[i-1] \dots \text{DNA}[i+3]\}$; then the tokens $\text{T}[j], \forall (j)_{j=i-2}^{i+3}$ are masked. We enforce this mapping between a nucleotide and a 6-mer token in GENEMASK as the *MapNucleotideToKmerTokens* function in Algorithm 1. Given that $k = 6$ and 15% of tokens need to be

¹ Sequence motifs are short, recurring patterns in DNA that are presumed to have a biological function [8].

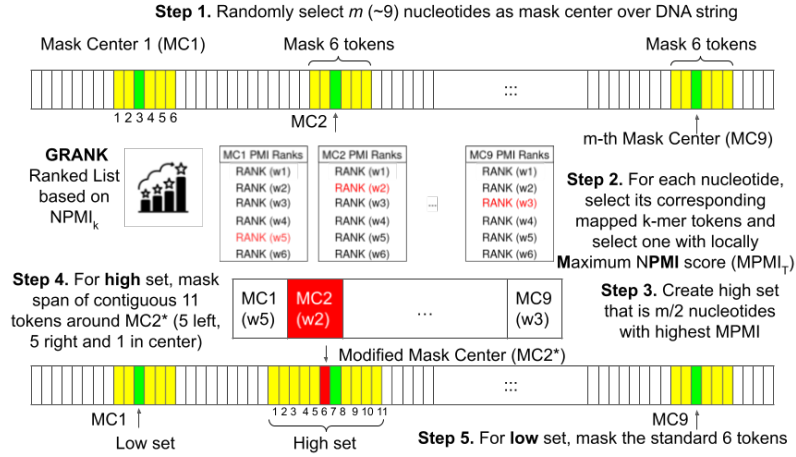


Figure 1. Method overview of GENEMASK, our proposed masking algorithm for MLM training of gene sequences, where we randomly select mask centers over the input DNA string and locally select the span around the mask center with the highest Normalized Pointwise Mutual Information ($NPMI_k$) to mask

masked [7], the MLM probability is set at: $15\%/6$; that is, 2.5% of the nucleotides are chosen for masking. (Since we are mainly working with 6-mer unless mentioned explicitly, $T[i] = T_6[i]$.)

4 Methodology

GENEMASK aims to mask the k ($=6$)-mers that co-occur much more than expected compared to their components (i.e., k -mers of shorter length, such as 4-mers or 5-mers). These spans then replace the (uniform) random masking strategy used by SOTA models. Removing highly correlated local contexts make the masked token prediction task more difficult, which may improve the pretraining efficiency [21]. However, GENEMASK significantly differs from the *PMI-masking* strategy [21] used in NLP, which first uses the (un-normalized) PMI score to formulate an absolute importance notion and subsequently creates a fixed masking vocabulary. To the best of our knowledge, this is the first work to develop a principled (statistical) approach to identify a highly correlated span of tokens in gene sequences based on normalized PMI score (Section 4.1) and then develop a novel masking algorithm (GENEMASK) for efficient MLM training of gene-based (pretrained) SOTA models like DNABert and LOGO (Section 4.2); Figure 1 provides the methodology overview.

4.1 PMI-based Metric to Identify Correlated Spans in Gene Sequences

This work considers a single nucleotide equivalent to a single token in NLP. Thus an n -gram from the NLP domain is equivalent to a k -mer from the gene sequence modeling literature. We propose a novel strategy to adapt PMI-based scoring to our genomic setting, which can help us identify high PMI tokens to mask. *Pointwise Mutual Information* (PMI) quantifies how often two tokens occur compared to what is expected if they are independent. The PMI formula (proposed by Levine et al. [21]) when extended to k -mers (where $k > 2$) is:

$$PMI_k(w_1 \dots w_k) = \min_{\sigma \in \text{seg}(w_1 \dots w_k)} \log \frac{p(w_1 \dots w_k)}{\prod_{s \in \sigma} p(s)} \quad (1)$$

Here, $\text{seg}(w_1 \dots w_k)$ is the set of all contiguous segmentations of the k -mer “ $w_1 \dots w_k$ ” (excluding the identity segmentation). In a

valid segmentation (σ), the original sequence “ $w_1 \dots w_k$ ” can be divided into any number of partitions of positive (> 0) size. For example, say for $k = 6$, some of the possible valid segmentations are: “ $(w_1 \dots w_3), (w_4 \dots w_6)$ ” or “ $(w_1, w_2), (w_3), (w_4 \dots w_6)$ ”.

The PMI_k formulation many times favors tokens with lower frequency, that is, the number of times the k -mer gene sequence appears in the *Human Reference Genome*. We thus impose a **discounting factor** that penalizes rare tokens [30] because such PMI tokens will be more frequently selected as masked tokens during the MLM training stage. This may lead the model to over-emphasize corner cases and thus wrongfully mask ‘not-so-easy’ cases. We refer to it as the *Normalized PMI* ($NPMI_k$) formula, which we finally use for scoring all the individual n -gram sequences.

$$NPMI_k(w_1 \dots w_k) = PMI_k * \frac{\log f(w_1 \dots w_k)}{\log(c) + \log f(w_1 \dots w_k)} \quad (2)$$

Here, $f(w_1 \dots w_k)$ refers to the frequency of occurrence of the k -mer sequence of $w_1 \dots w_k$. c refers to the minimum frequency of occurrence (a constant value used as a threshold to remove rare tokens). In this paper, we focus only on computing $NPMI_k$ for all k -mer sequences where $k = 6$, and develop a **ranked list (GRANK) of all 6-mers** (4096 in total) based on the decreasing order of $NPMI_k$. Next, we discuss how we use the $NPMI_k$ scores as a measure to choose tokens to be masked during MLM training.

4.2 Masking Algorithm for Efficient Pretraining over Gene Sequences

GENEMASK aims to mask all the nucleotides simultaneously in the most correlated spans. Masking correlated spans minimizes information leakage and helps the system to learn deeper patterns. Side by side, we would like to preserve the benefit drawn out of the traditional random masking strategy. Therefore, we propose a novel linear-time masking algorithm, GENEMASK, for MLM training of gene sequences, where we randomly identify positions in a gene sequence as mask centers and locally select the span around the mask center with the highest $NPMI_k$ to mask. We present the details of our proposed GENEMASK strategy in Algorithm 1.

Algorithm 1: GENEMASK Algorithm

Input: DNA string made of 6-mer tokens with a maximum length of 510, Dictionary containing $NPMI_k$ values for all 6-mers
Output: *MaskTokenSet*: List of tokens to mask in the DNA string
Initialization: // $T[i]$ denotes a 6-mer token at i -th position in the DNA segment, $DNA[i]$ denotes the i -th nucleotide
 $MaskTokenSet \leftarrow \emptyset$
 $T[i] \leftarrow \{DNA[i-2] \dots DNA[i+3]\}$
Function MapNucleotideToKmerTokens (nucleotide position $id\ i$):
 $MappedTokens \leftarrow T[j], \forall (j)_{j=i-2}^{i+3}$
 return $MappedTokens$
Step 1: Randomly select m nucleotides as mask centers (MC) spread uniformly over the DNA string.
// In this step corresponding to each mask center, the neighboring nucleotide which locally has the highest $NPMI_k$ is chosen
Step 2: for each nucleotide in m mask centers **do**
 $PositionId :=$ Token index of nucleotide on the DNA segment given as input
 $KmerTokens := MapNucleotideToKmerTokens(PositionId)$
 // Select kmer with the highest $NPMI_k$ score, $T[\tau]$
 $T[\tau] \leftarrow \arg \max_{kmer \in KmerTokens} NPMI_k(kmer)$
 // Store the Locally Maximum NPMI score as $MPMI_\tau$
 $MPMI_\tau \leftarrow NPMI_k(T[\tau])$
Step 3: Divide the m nucleotides into two sets based upon their $MPMI$ scores, where the high set is the $m/2$ nucleotides with the highest $MPMI$ scores.
Step 4: for each nucleotide in high set **do**
 // Mask all the nucleotides in $T[\tau]$ (this ensures the masking of correlated spans together, 11 tokens in length)
 $MaskTokenSet \leftarrow MaskTokenSet \cup \forall (j)_{j=\tau-2}^{\tau+3}$
 $MapNucleotideToKmerTokens(j)$
for each nucleotide in low set **do**
 // Mask only the corresponding nucleotide $DNA[i]$ (this mimics a random masking strategy)
 $MaskTokenSet \leftarrow MaskTokenSet \cup MapNucleotideToKmerTokens(i)$
return $MaskTokenSet$

Determining the value of m : To mask six base pairs-long gene sequence that is all the nucleotides in a particular token $T[i]$, we need to mask a span of contiguous 11 tokens (6 mask centers, two tokens to the left and three tokens right) while as mentioned a single nucleotide induce masking of 6 tokens. Thus, the expected mask span length per mask center is computed as $0.5 * 6 + 0.5 * 11 = 8.5$. Subsequently, the *mlm probability* is updated from 2.5% to 1.765% ($= 15\%/8.5$). Hence when the DNA length is 512, $m \approx 9$.

Efficiency of GENEMASK algorithm. GENEMASK first samples m random nucleotides as mask centers and locally chooses the one with the highest $NPMI_k$ score within the span (fixed number of adjacent tokens) of the mask center. Thus the time complexity of GENEMASK becomes $O(m)$. Since $m = c * n$, where n is the input sequence length and c is a constant value (equal to 1.765% in our case), the time complexity of GENEMASK is: $O(m) = O(c * n) = O(n)$.

5 Experimental Setup

Here, we provide the dataset specifics, evaluation setup, model training, and baseline model details.

5.1 Datasets

We use four benchmark datasets of gene sequence classification (PROM-CORE, PROM-300, SPLICE-40, COHN-ENH) for evaluation purposes. The two datasets of promoter region prediction and splice site prediction are not directly available and involve significant effort (including a paper implementation) for their construction.

PROMOTER REGION PREDICTION (PROM-CORE AND PROM-300). A *promoter* is a DNA region typically located upstream of the gene, which is the site of transcription initiation (as defined by Zaheer et al. [46]). The task is to classify a given DNA fragment as a promoter or non-promoter sequence. However, we follow the instructions of the DeepPromoter [29] paper, including the negative data creation, since the authors do not provide the datasets. Thus, we obtained human TATA and non-TATA promoter data, i.e., including promoter sequences with and without a TATA box (a common promoter-related motif found between -30 to -25 bp (upstream) of a gene’s transcription start site), from the Eukaryotic Promoter Database [10], using the website API of the *EPD selection tool* [12]. We extracted -249 to $+50$ bp sequences around TSS for the **Prom-300 dataset** and -34 to $+35$ bp for the **Prom-core dataset**. We perform the standard train-test split of 70% and 30%, which leads to 53276 and 5920 data points, respectively.

SPLICE DONOR AND ACCEPTOR SITE PREDICTION (SPLICE-40). We followed the same strategy as prior works [16, 41] for dataset construction. We extract 40 bp long sequences around the donor and acceptor sites of exons (randomly selected) as positive sequences. The exon’s position information (chromosome number, start index, and end index) is obtained from the corresponding gene annotation file. The task is to classify a given DNA fragment as a donor, acceptor, or non-splice site (non-overlapping intermediate sequences between exons) sequence. We perform the standard train-test split of 70 – 30%, which leads to 24300 and 3000 training and test data points, respectively.

ENHANCER COHN PREDICTION (COHN-ENH). An *enhancer* is a DNA sequence that can bind specific proteins and increase the chance of transcription of a particular gene. Here, the input is a DNA sequence of 500 bp in length and a binary classification task where the task is to classify a DNA fragment as an enhancer or non-enhancer sequence. This dataset has been adapted from Cohn et al. [4] and is made available as a benchmark dataset by Martinek et al. [22] in Github [13]; we use the same train-test split that leads to 20843 and 6948 data points as train and test datasets, respectively.

5.2 Evaluation Setup

We report the standard metrics of accuracy (used for performance comparison) and AUC (stands for Area Under the Receiver Operating Characteristic Curve) used for classification tasks where the class labels are balanced. We follow the standard evaluation setup used in the few-shot text classification setting [33, 34]. Thus, we assume not to have access to a validation dataset to optimize the hyperparameters and investigate the performance for different training set sizes (few-shot settings) $n = 10, 50, 100, 500$, and 1000, where n denotes the number of training data points per class. We report the mean and standard deviation of accuracy and AUC by running the experiments **ten** times by randomly choosing seed and n training (fine-tuning) data points per class in each run. We report the statistical significance results based on paired t-test for the performance improvement by GENEMASKBEST over ORI 10K model.

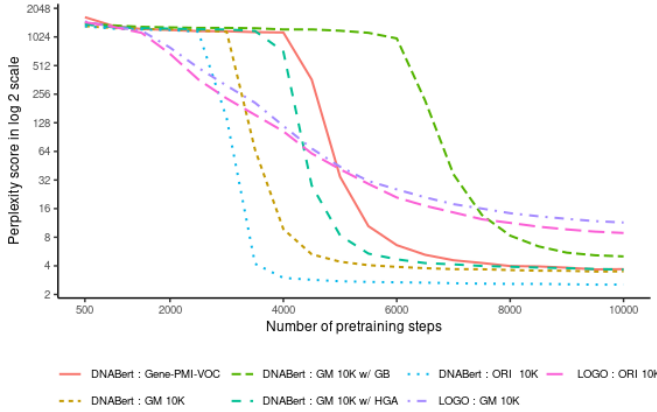


Figure 2. Perplexity score plot of GENEMASK (GM)

5.3 Training Details

In this paper, we train all the pretrained models and their variants for only 10000 steps, which takes about **2.5 days to complete for DNABert** and around **20 hours for LOGO** using four GTX 1080Ti 11GB GPUs. We select such a setup for two reasons — (i) to explore different pretrained model variants in a reasonable time because we observe that the perplexity of DNABert (SOTA model) has converged to a low score and is stable over the last 3000 pretraining steps (see Figure 2). (ii) As observed by Levine et al. [21], PMI masking learns fast and is thus quite efficient to reap the benefit even with fewer pretraining steps. The default number of warmup steps for DNABert is 10K steps out of 200K (5% of the maximum number of steps). Since we reduce the maximum pretraining steps limit from 200K to 10K, we set the number of warmup steps as 500 (5% of 10000 steps) to maintain the same ratio. We use the same values as the original SOTA models for the remaining hyperparameters (see section A.4 of Appendix to know more about training (including fine-tuning) details and hyperparameters). We observe in Figure 2 that the perplexity scores of the baseline models are lower. The same trend is also reported by Levine et al. [21]. However, the single-token perplexity values are not comparable between models with different masking strategies and, thus, do not indicate downstream performance.

5.4 Baseline Models

We evaluate our work on two gene transformer-based models - DNABert [16] and LOGO [44]. To perform a fair comparison, we train the baseline models with the same hyperparameter settings as GENEMASK. These models follow **random masking** during the MLM training step instead of the proposed GENEMASK strategy. We use the same model hyperparameters for pretraining GENEMASK and the baseline models to perform a fair comparison. We will refer to the SOTA models that use random masking as the original SOTA model (**ORI**) without any PMI-guided masking. We use the original DNABert model pretrained on 120K steps (**ORI 120K**) based on the pretrained model weights provided by [16] as a baseline model.

Fixed PMI-guided Masking Vocabulary (Gene-PMI-VOC): We undertook a significant effort to adapt the PMI-masking strategy in NLP proposed by Levine et al. [21] to the genomic setting which is a contribution in itself. We create a PMI-masking vocabulary ≈ 10

times the DNABert vocabulary size of 4101 tokens. We first select all possible k-mer sequences ($2 \leq k \leq 10$) whose frequency of occurrence is ≥ 10000 . We then rank them using our proposed PMI metric and select the top 40000 as the masking vocabulary. During masking, we randomly select the mask centers (nucleotides) and then use the masking vocabulary to tokenize the input gene sequence into PMI tokens. We mask an entire PMI token (2 to 10-mer) within which the selected mask center lies (based on tokenization); Algorithm 2 (appendix) describes our implementation in detail.

Pretraining Model Variants of GENEMASK. We explore two different pretraining setups — (i) Half Gradient Accumulation (task-independent) and (ii) Gene Boundary-aware Pretraining (task-specific, boosts performance over Splice-40 dataset).

(i). Half Gradient Accumulation (HGA): The *gradient accumulation steps* parameter is halved, reducing it from 25 (default DNABert configuration) to 12 steps, and consequently, it reduces the effective batch size (it is the product of per GPU train batch size, GPU count and gradient accumulation steps) by 50%. This aims to reduce the *generalization gap* issue that arises when the training batch size is too large [15, 18].

(ii). Gene Boundary-aware Pretraining (GB): In the pretraining data construction stage, as previously described in Section 3, we do not consider gene boundaries. Thus, it may happen that the DNA segment spans across different genomic entities and, since such spans have no semantic underpinning, may introduce noise during the masked token prediction step of MLM training. We obtain the gene boundary information from the gene annotation file, corresponding to the same Human Reference Genome, as described in the pretraining data construction part of Section 3. To limit a DNA segment within a single gene boundary, we use the minimum distance to the next gene boundary as an upper limit for determining the length of the DNA segment L ; each DNA segment corresponds to a single data point in the pretraining dataset.

6 Experimental Results

We compare the performance of our proposed GENEMASK strategy and random masking strategy-based SOTA models of DNABert and LOGO as shown in Table 1. We observe that the performance improvement due to GENEMASK strategy over random masking strategy (ORI 10K model) is more prevalent in lower data settings (10, 50-shot), reduces at higher data settings of 500 and 1000-shot (except for Splice-40), and ceases to exist when trained on full datasets, as evident from Figure 3 of Appendix. We consider the two additional pretraining model variations for DNABert - *Half Gradient Accumulation* and *Gene Boundary-aware Pretraining* and along with the base model referring to the GENEMASK model having the same hyperparameters as the baseline models such as ORI 10K. We present the best performance among the three models, such as GENEMASK, GENEMASKw/ HGA and GENEMASKw/ GB] as GENEMASKBEST model. The GENEMASK model is also referred to as GENEMASK 10K to indicate the number of training steps.

Performance comparison among different masking strategies. In the case of *DNABert*, GENEMASKBEST mostly (two out of three times) outperforms all the baseline models across the four datasets of Prom-core, Prom-300, Cohn-enh, and Splice-40, and in the remaining case where it does not achieve the best performance, it is always the second-best ranked model (see Table 1, the second best-performing model is highlighted in yellow). We observe that the task-specific performance improvement between GENEMASKBEST and random masking (ORI 10K model variant) in *DNABert*, in terms of

Data per class	Model type	Prom-core		Prom-300		Cohn-enh		Splice-40	
		Accuracy	AUC	Accuracy	AUC	Accuracy	AUC	Accuracy	AUC
DNABert as Base model									
10	ORI 120K	0.606 ± 0.045	0.661 ± 0.064	0.638 ± 0.070	0.708 ± 0.088	0.582 ± 0.030	0.631 ± 0.044	0.404 ± 0.019	0.577 ± 0.021
	Gene-PMI-VOC	0.6 ± 0.047	0.652 ± 0.067	0.653 ± 0.065	0.731 ± 0.066	0.596 ± 0.048	0.66 ± 0.062	0.412 ± 0.02	0.599 ± 0.02
	ORI 10K	0.586 ± 0.051	0.641 ± 0.082	0.601 ± 0.065	0.657 ± 0.095	0.579 ± 0.047	0.655 ± 0.058	0.409 ± 0.017	0.588 ± 0.018
	GENEMASKBEST	0.602 ± 0.058	0.655 ± 0.078	0.676 ± 0.054**	0.779 ± 0.074**	0.622 ± 0.050*	0.701 ± 0.038*	0.412 ± 0.026	0.591 ± 0.036
50	ORI 120K	0.687 ± 0.024	0.756 ± 0.03	0.808 ± 0.019	0.89 ± 0.013	0.638 ± 0.020	0.679 ± 0.028	0.472 ± 0.048	0.651 ± 0.049
	Gene-PMI-VOC	0.649 ± 0.058	0.738 ± 0.033	0.800 ± 0.027	0.893 ± 0.02	0.645 ± 0.014	0.706 ± 0.016	0.522 ± 0.071	0.703 ± 0.062
	ORI 10K	0.653 ± 0.058	0.718 ± 0.064	0.789 ± 0.059	0.882 ± 0.04	0.634 ± 0.031	0.689 ± 0.044	0.512 ± 0.014	0.687 ± 0.017
	GENEMASKBEST	0.678 ± 0.026	0.744 ± 0.026	0.815 ± 0.02	0.905 ± 0.013	0.654 ± 0.017*	0.713 ± 0.011†	0.519 ± 0.027	0.696 ± 0.023
100	ORI 120K	0.712 ± 0.009	0.781 ± 0.012	0.842 ± 0.014	0.915 ± 0.012	0.669 ± 0.017	0.736 ± 0.022	0.507 ± 0.059	0.683 ± 0.063
	Gene-PMI-VOC	0.697 ± 0.011	0.767 ± 0.013	0.835 ± 0.017	0.912 ± 0.014	0.65 ± 0.051	0.737 ± 0.011	0.605 ± 0.017	0.779 ± 0.015
	ORI 10K	0.695 ± 0.014	0.765 ± 0.017	0.842 ± 0.018	0.923 ± 0.009	0.668 ± 0.015	0.736 ± 0.011	0.543 ± 0.027	0.72 ± 0.024
	GENEMASKBEST	0.708 ± 0.013*	0.779 ± 0.015*	0.847 ± 0.029	0.920 ± 0.020	0.67 ± 0.017	0.737 ± 0.013	0.577 ± 0.019**	0.751 ± 0.017**
500	ORI 120K	0.743 ± 0.008	0.819 ± 0.007	0.883 ± 0.006	0.951 ± 0.005	0.698 ± 0.009	0.776 ± 0.011	0.429 ± 0.07	0.608 ± 0.077
	Gene-PMI-VOC	0.738 ± 0.021	0.82 ± 0.017	0.884 ± 0.004	0.948 ± 0.008	0.692 ± 0.011	0.759 ± 0.011	0.639 ± 0.025	0.809 ± 0.021
	ORI 10K	0.752 ± 0.007	0.831 ± 0.003	0.888 ± 0.007	0.958 ± 0.002	0.696 ± 0.009	0.767 ± 0.008	0.665 ± 0.035	0.834 ± 0.022
	GENEMASKBEST	0.753 ± 0.005	0.831 ± 0.003	0.89 ± 0.006	0.957 ± 0.002	0.698 ± 0.006	0.771 ± 0.007	0.692 ± 0.016*	0.851 ± 0.01†
1000	ORI 120K	0.758 ± 0.006	0.835 ± 0.004	0.895 ± 0.005	0.957 ± 0.005	0.700 ± 0.009	0.769 ± 0.009	0.496 ± 0.109	0.673 ± 0.107
	Gene-PMI-VOC	0.759 ± 0.007	0.834 ± 0.006	0.895 ± 0.004	0.96 ± 0.004	0.698 ± 0.007	0.766 ± 0.009	0.504 ± 0.02	0.689 ± 0.016
	ORI 10K	0.765 ± 0.004	0.839 ± 0.005	0.901 ± 0.003	0.964 ± 0.002	0.705 ± 0.005	0.776 ± 0.006	0.651 ± 0.019	0.821 ± 0.015
	GENEMASKBEST	0.766 ± 0.007	0.843 ± 0.006	0.898 ± 0.005	0.962 ± 0.002	0.706 ± 0.005	0.778 ± 0.006	0.709 ± 0.012**	0.865 ± 0.008**
LOGO as Base model									
10	ORI 10K	0.506 ± 0.017	0.557 ± 0.064	0.502 ± 0.005	0.557 ± 0.028	0.53 ± 0.043	0.599 ± 0.067	0.333 ± 0.012	0.565 ± 0.022
	GENEMASK 10K	0.54 ± 0.049†	0.582 ± 0.088†	0.519 ± 0.04	0.594 ± 0.086	0.553 ± 0.063	0.649 ± 0.067*	0.353 ± 0.017*	0.558 ± 0.028
50	ORI 10K	0.565 ± 0.048	0.635 ± 0.037	0.618 ± 0.037	0.663 ± 0.05	0.627 ± 0.007	0.676 ± 0.006	0.438 ± 0.019	0.61 ± 0.021
	GENEMASK 10K	0.611 ± 0.04**	0.676 ± 0.017**	0.668 ± 0.014**	0.733 ± 0.022**	0.635 ± 0.006†	0.692 ± 0.009**	0.465 ± 0.018**	0.636 ± 0.022*
100	ORI 10K	0.628 ± 0.014	0.677 ± 0.017	0.646 ± 0.033	0.694 ± 0.039	0.638 ± 0.008	0.691 ± 0.011	0.447 ± 0.049	0.624 ± 0.043
	GENEMASK 10K	0.644 ± 0.008*	0.695 ± 0.012*	0.705 ± 0.023**	0.783 ± 0.023**	0.639 ± 0.006	0.695 ± 0.008	0.465 ± 0.021	0.651 ± 0.02†
500	ORI 10K	0.693 ± 0.010	0.754 ± 0.007	0.751 ± 0.034	0.817 ± 0.037	0.629 ± 0.006	0.679 ± 0.007	0.355 ± 0.027	0.629 ± 0.014
	GENEMASK 10K	0.69 ± 0.006	0.748 ± 0.009	0.835 ± 0.008**	0.907 ± 0.008**	0.648 ± 0.004**	0.709 ± 0.04**	0.443 ± 0.025**	0.644 ± 0.015*
1000	ORI 10K	0.705 ± 0.014	0.769 ± 0.013	0.808 ± 0.015	0.884 ± 0.016	0.652 ± 0.008	0.709 ± 0.008	0.504 ± 0.02	0.689 ± 0.016
	GENEMASK 10K	0.723 ± 0.005	0.785 ± 0.006	0.855 ± 0.004**	0.925 ± 0.004**	0.660 ± 0.004*	0.727 ± 0.004**	0.507 ± 0.011	0.701 ± 0.007†

Table 1. Performance comparison for gene sequence classification tasks where all values are rounded to 3 decimal places. The best-performing performance value is written in **bold**, and the second-best-performing value is highlighted in yellow. ORI refers to the SOTA model with random masking. Paired t-test was conducted to check whether the performance improvement of GENEMASK-based models over ORI 10K model is statistically significant or not; ** for p -value < 0.01 , * for $p < 0.05$, and † for p -value < 0.1 .

average accuracy over all few-shot settings for each task, is the highest for Splice-40 followed by Prom-300. Since the GENEMASK (w/ GB) pretraining model variant is explicitly targeted for Splice-40, it highlights the effectiveness of such task-specific interventions.

In the case of LOGO, we do not explore different pretraining model variants and instead compare between ORI 10K and GENEMASK 10K model (both of them have exactly the same model hyperparameters). We observe that GENEMASK 10K (base model) outperforms random masking-based SOTA models (ORI) in all settings across four benchmark tasks as shown in Table 2. The highest percentage improvement in terms of average accuracy over all few-shot settings is Splice-40 with 8.32%, followed by Prom-300, Prom-core, and Cohn-enh at 7.67%, 3.72% and 1.95% respectively. However, we observe the performance improvement of GENEMASK to be higher for LOGO as compared to DNABert for all tasks except the Cohn-enh task.

k-shot	10	50	100	500	1000
DNABert	2.94%	0.93%	0.73%	0.40%	1.85%
LOGO	4.92%	5.87%	3.90%	7.74%	2.85%

Table 2. Percentage improvement in average accuracy over four datasets of GENEMASK 10K over ORI 10K model.

GENEMASK 10K versus ORI 10K model. In the case of DNABert and LOGO, ORI-10K almost always performs much inferior to any variant of GENEMASK, although trained for the same number of epochs. Table 2 shows the percentage improvement in average accuracy across four datasets of GENEMASK 10K over the ORI 10K model for various k -shot setups. It indicates that GENEMASK-based masking strategy is more beneficial for lightweight models like LOGO; heavier models like DNABert might automatically learn a certain amount of span correlations (like PMI) informa-

tion, thus diminishing the independent impact of GENEMASK. From now onwards, we will discuss only DNABert-based models and compare the performance of GENEMASKBEST with the baseline models.

GENEMASKBEST versus Gene-PMI-VOC model. We observe that GENEMASKBEST almost consistently outperforms the Gene-PMI-VOC model, except for the 10, 50, and 100-shot settings of the Splice-40 task. We further observe that the performance of Gene-PMI-VOC is quite unstable as it even performs poorer than the ORI 10K model in most cases (12 out of 20 DNABert settings in Table 1). A possible reason for such inconsistent performance is its sensitivity to the (arbitrary) size of its masking vocabulary, created using a PMI score to formulate an absolute importance notion. However, determining the optimal vocabulary size in a principled manner is difficult due to the absence of human-understandable semantics in genomics (like words and phrases in NLP). A suboptimal choice may lead to difficulty in guaranteeing diverse masking patterns, thus ushering in an inefficient MLM training regime.

k-shot	10	50	100	500	1000
ORI 120K	3.58%	2.46%	2.64%	10.17%	8.15%

Table 3. Percentage improvement in average accuracy over four datasets of GENEMASKBEST over ORI 120K model in case of DNABert.

GENEMASKBEST versus ORI-120K model. Table 3 shows the percentage improvement in average accuracy scores (over four datasets) of GENEMASKBEST over the original DNABert model trained for 120K steps (ORI 120K model). GENEMASKBEST improves decently over ORI 120K in low data settings (100-shot and below) but the performance rises in 500 and 1000-shot settings, which is contrary to our standard observation (all datasets except Splice-40). This happens due to the Splice-40 task, where the performance improvement due to GENEMASKBEST increases at higher

data settings (0.692 versus 0.429 and 0.709 versus 0.496 for GENEMASKBEST and ORI 120K model respectively) as GENEMASK (w/ GB) is effective at that setting. Another interesting observation is that ORI 120K model outperforms GENEMASKBEST on shallow data settings (10, 50, and 100-shot) of the Prom-core task, whereas GENEMASKBEST outperforms again in 500 and 1000-shot settings. This may be because Prom-core is an easier task with a much shorter context (70 base pairs (bp) in length as compared to 300 bp for Prom-300 and 500 bp for Cohn-enh), whereby the ORI 120K model simply memorizes the gene sequence patterns instead of actually learning intrinsic (or extra) knowledge of gene sequences. Hence, the effect of memorization recedes in higher data settings (above 500-shot), as seen in Figure 3 of Appendix. The reason behind the improvement in the Splice-40 task at a higher k -shot is also similar.

Deconstructing GENEMASKBEST. We compare the performance of three model variants of GENEMASK. We observe from Table 4 that half gradient accumulation (HGA) helps to improve the model performance in 10 and 50-shot settings (10 and 50-shot for Prom-300, 50-shot for Cohn-enh, and 10-shot for Splice-40). It is empirically observed that if large batch size is used to train deep neural networks, the trained models appear to generalize poorly [15, 18]. HGA reduces the effective batch size by 50% and mitigates the *generalization gap* issue in such low data settings (10 and 50-shot). However, beyond the 100-shot setting, the impact is negative or marginal. We observe that when we consider gene boundaries during pretraining data construction (GB), the performance consistently improves for the Splice-40 task. The primary reason may be that the Splice-40 task involves identifying exon boundaries, which can benefit strongly from gene boundaries as side information.

k-shot	Model Type	Prom-core	Prom-300	Cohn-enh	Splice-40
10	GENEMASK	0.602 ± 0.058	0.625 ± 0.09	0.622 ± 0.050	0.392 ± 0.02
	GENEMASKw/ HGA	0.585 ± 0.065	0.676 ± 0.054	0.604 ± 0.068	0.412 ± 0.026
	GENEMASKw/ GB	0.568 ± 0.05	0.593 ± 0.038	0.602 ± 0.043	0.398 ± 0.015
50	GENEMASK	0.678 ± 0.026	0.781 ± 0.097	0.648 ± 0.016	0.505 ± 0.018
	GENEMASKw/ HGA	0.677 ± 0.033	0.815 ± 0.02	0.654 ± 0.017	0.448 ± 0.095
	GENEMASKw/ GB	0.669 ± 0.025	0.793 ± 0.022	0.641 ± 0.012	0.519 ± 0.027
100	GENEMASK	0.708 ± 0.013	0.843 ± 0.027	0.655 ± 0.052	0.561 ± 0.027
	GENEMASKw/ HGA	0.694 ± 0.043	0.847 ± 0.029	0.67 ± 0.017	0.555 ± 0.079
	GENEMASKw/ GB	0.695 ± 0.014	0.829 ± 0.016	0.661 ± 0.008	0.577 ± 0.019
500	GENEMASK	0.753 ± 0.005	0.89 ± 0.006	0.698 ± 0.006	0.669 ± 0.017
	GENEMASKw/ HGA	0.75 ± 0.008	0.887 ± 0.004	0.674 ± 0.058	0.662 ± 0.026
	GENEMASKw/ GB	0.746 ± 0.006	0.877 ± 0.005	0.675 ± 0.005	0.692 ± 0.016
1000	GENEMASK	0.766 ± 0.004	0.898 ± 0.005	0.706 ± 0.005	0.709 ± 0.012
	GENEMASKw/ HGA	0.766 ± 0.007	0.898 ± 0.004	0.699 ± 0.008	0.69 ± 0.025
	GENEMASKw/ GB	0.756 ± 0.005	0.889 ± 0.005	0.688 ± 0.004	0.693 ± 0.02

Table 4. Performance comparison in terms of accuracy among GENEMASK-guided DNABert model variants at 10000 steps. GENEMASK represents the hyperparameter settings of baseline models as ORI 10K model

7 Domain-specific Model Explainability

Motifs are repetitive units within a *Human Reference Genome*, having a certain biological significance. We check whether the correlated tokens identified by GENEMASK are, in fact (part of) such units. We also perform a preliminary evaluation on the task of identifying functional genetic variants in Section A.6.1.

Associating top-20 ranked 6-mers based on Normalized-PMI_k score with conserved DNA sequence motifs. We analyze whether highly ranked PMI tokens resemble meaningful concepts by checking their overlap with known motifs. We performed a Google search with the following query template: [“6-mer name” *DNA sequence motif*]. *AATCTC* is a 6-mer, “*” are used as a Google wildcard to indicate that the term *AATCTC* must always be present in search

Dataset	Motifs (Consensus Logo)	Normalized PMI rank (out of 4096)
Prom-core	nTATAAAr	242
Cohn-enh	GTGGCTsw	126
Prom-core, Cohn-enh	nCyyCCTCCn*	1, 11, 52, 175, 186
Prom-core	sCwGCAGCn	259, 516, 540, 570, 628
Cohn-enh	ksCTGGGm	5, 17, 20, 21, 71
Cohn-enh	TTTTTTTTTn	8

Table 5. PMI-based rankings based on Normalized-PMI_n score for motifs in finetuning datasets. A motif of length five matches as a sub-string to multiple 6-mers and thus mention the top five ranks

results. We only considered the first page of Google results to determine whether a particular 6-mer is mentioned in biomedical literature. Among the *top 20 ranked* 6-mers, we observed that all except 2 (*CCAGGC* - rank nine, *GCCTGG* - rank ten) are indeed previously mentioned in the biomedical literature. Since the GENEMASK strategy favors the top-ranked PMI tokens that correlate well with the known DNA sequence motifs of enhancers and promoters, this knowledge translates to improved performance in low data settings (10, 50, and 100-shot) on the Prom-core, Prom-300, and Cohn-enh datasets.

PMI-based rankings capture motifs present in finetuning datasets. We use the R package *rGADEM* [11] to perform de novo motif discovery and obtain a total of 12 motifs from the Prom-core and Cohn-enh datasets, which are mostly concentrated around lengths 5, 6, and 7. We present our best matches and their corresponding ranks in the 6-mer PMI ranked list (*RANK*) in Table 5 (see Table 9 for the complete list, their corresponding consensus logos in Figures 4 and 5). We observe that most of the 6-mers that match the discovered motifs are ranked very high. We further observe that our top-1 ranked 6-mer is present in motifs of both enhancers and promoters based on de-novo motif discovery and is mentioned in previous biomedical literature [3]. The TATA box, a well-known motif for promoters (row 1 of Table 5) is ranked at 242; the best TATA box motif is *TATATA* with a PMI rank of 15.

8 Conclusion

In this paper, we develop a PMI- induced novel masking algorithm, GENEMASK, for MLM training of gene sequences which ensures a substantial speedup of 10x and performance improvement over the random masking strategy of SOTA models (DNABert and LOGO) in various few-shot settings. Further we showed that for a certain task (Splice-40), respecting gene boundaries during pretraining data construction improves the performance even further, confirming the basic thesis of the importance of incorporating domain-specific information during pretraining. This is further corroborated when we observe a strong correlation between top-ranked NPMI tokens and conserved DNA sequence motifs, implying that GENEMASK is able to capture meaningful semantic structure in gene sequences. Finally, we must mention that gene sequence classification is a challenging problem and requires a tremendous engineering effort even to develop the experimental setup. Therefore, the codes (including trained models), datasets, and appendix are made publicly available at <https://github.com/roysoumya/GeneMask>. The elaborate setup developed will help us undertake several **future work**, for example, we will explore time-variant MLM training strategies for gene sequences to achieve better few-shot performance, such as incorporating a decaying masking ratio, which showed good performance on GLUE and SQuAD datasets in NLP domain [43].

Acknowledgements

Soumyadeep Roy is supported by the Institute Ph.D. Fellowship at the Indian Institute of Technology Kharagpur. Soumyadeep Roy and Niloy Ganguly were also affiliated with L3S Research Center, Germany while conducting this work. This research was funded by the Federal Ministry of Education and Research (BMBF), Germany, under the project LeibnizKILabor with grant No. 01DD20003.

References

- [1] Žiga Avsec, Vikram Agarwal, et al., ‘Effective gene expression prediction from sequence by integrating long-range interactions’, *Nature Methods*, **18**(10), 1196–1203, (Oct 2021).
- [2] Sarkhan Badirli, Zeynep Akata, et al., ‘Fine-grained zero-shot learning with dna as side information’, in *Advances in Neural Information Processing Systems*, volume 34, pp. 19352–19362, (2021).
- [3] B.K. Chow, V. Ting, et al., ‘Characterization of a novel liver-specific enhancer in the human prothrombin gene.’, *Journal of Biological Chemistry*, **266**(28), 18927–18933, (1991).
- [4] Dikla Cohn, Or Zuk, et al., ‘Enhancer identification using transfer and adversarial deep learning of dna sequences’, *bioRxiv*, (2018).
- [5] ENCODE Project Consortium, ‘An integrated encyclopedia of dna elements in the human genome’, *Nature*, **489**(7414), 57–74, (2012).
- [6] Genome Reference Consortium. Genome reference consortium human build 38 patch release 13 (grch38.p13), 2019.
- [7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, ‘BERT: pre-training of deep bidirectional transformers for language understanding’, *CoRR*, **abs/1810.04805**, (2018).
- [8] Patrik D’haeseleer, ‘What are dna sequence motifs?’, *Nature Biotechnology*, **24**(4), 423–425, (Apr 2006).
- [9] Dnabert github repository, 2019.
- [10] René Dreos, Giovanna Ambrosini, et al., ‘EPD and EPDnew, high-quality promoter resources in the next-generation sequencing era’, *Nucleic Acids Research*, **41**(D1), D157–D164, (11 2012).
- [11] Arnaud Droit, Raphael Gottardo, Gordon Robertson, and Leiping Li, *rGADEM: de novo motif discovery*, 2022. R package version 2.44.1.
- [12] Epd selection tool of epd eukaryotic promoter database, 2022.
- [13] Katarina Gresova, Vlastimil Martinek, et al., ‘Genomic benchmarks: A collection of datasets for genomic sequence classification’, *bioRxiv*, (2022).
- [14] Eric B. Hekler, Predrag V. Klasnja, et al., ‘Why we need a small data paradigm’, *BMC Medicine*, **17**, (2019).
- [15] Elad Hoffer, Itay Hubara, et al., ‘Train longer, generalize better: closing the generalization gap in large batch training of neural networks’, in *Advances in Neural Information Processing Systems*, volume 30, (2017).
- [16] Yanrong Ji, Zhihan Zhou, et al., ‘DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome’, *Bioinformatics*, **37**(15), 2112–2120, (02 2021).
- [17] Mandar Joshi, Danqi Chen, et al., ‘SpanBERT: Improving Pre-training by Representing and Predicting Spans’, *Transactions of the Association for Computational Linguistics*, **8**, 64–77, (01 2020).
- [18] Nitish Shirish Keskar, Dheevatsa Mudigere, et al., ‘On large-batch training for deep learning: Generalization gap and sharp minima’, in *5th International Conference on Learning Representations, ICLR 2017, Conference Track Proceedings*, (2017).
- [19] Anshul Kundaje, Wouter Meuleman, et al., ‘Integrative analysis of 111 reference human epigenomes’, *Nature*, **518**(7539), 317–330, (2015).
- [20] Melissa J. Landrum, Jennifer M. Lee, et al., ‘ClinVar: public archive of relationships among sequence variation and human phenotype’, *Nucleic Acids Research*, **42**(D1), D980–D985, (11 2013).
- [21] Yoav Levine, Barak Lenz, et al., ‘Pmi-masking: Principled masking of correlated spans’, in *International Conference on Learning Representations*, (2021).
- [22] Vlastimil Martinek, David Cechak, et al., ‘Fine-tuning transformers for genomic tasks’, *bioRxiv*, (2022).
- [23] Amy L. McGuire, Stacey Gabriel, et al., ‘The road ahead in genetics and genomics’, *Nature Reviews Genetics*, **21**(10), 581–596, (Oct 2020).
- [24] Shentong Mo, Xi Fu, et al., ‘Multi-modal self-supervised pre-training for regulatory genome across cell types’, *CoRR*, **abs/2110.05231**, (2021).
- [25] Florian Mock, Fleming Kretschmer, et al., ‘Bertax: taxonomic classification of dna sequences with deep neural networks’, *BioRxiv*, (2021).
- [26] Marius Mosbach, Maksym Andriushchenko, et al., ‘On the stability of fine-tuning BERT: misconceptions, explanations, and strong baselines’, in *9th International Conference on Learning Representations, ICLR 2021*. OpenReview.net, (2021).
- [27] Patrick Ng, ‘dna2vec: Consistent vector representations of variable-length k-mers’, *CoRR*, **abs/1701.06279**, (2017).
- [28] Ngoc Giang Nguyen, Vu Anh Tran, et al., ‘Dna sequence classification by convolutional neural network’, *Journal of Biomedical Science and Engineering*, **9**(05), 280, (2016).
- [29] Mhaned Oubounyt, Zakaria Louadi, Hilal Tayara, and Kil To Chong, ‘Deepromoter: Robust promoter predictor using deep learning’, *Frontiers in Genetics*, **10**, (2019).
- [30] Patrick Pantel and Dekang Lin, ‘Discovering word senses from text’, in *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD ’02*, p. 613–619, (2002).
- [31] Alec Radford, Jeffrey Wu, et al., ‘Language models are unsupervised multitask learners’, *OpenAI blog*, **1**(8), 9, (2019).
- [32] Nafis Sadeq, Canwen Xu, and Julian McAuley, ‘InforMask: Unsupervised informative masking for language model pretraining’, in *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 5866–5878, Abu Dhabi, United Arab Emirates, (December 2022). Association for Computational Linguistics.
- [33] Timo Schick and Hinrich Schütze, ‘Exploiting cloze-questions for few-shot text classification and natural language inference’, in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pp. 255–269, Online, (April 2021).
- [34] Timo Schick and Hinrich Schütze, ‘It’s not just size that matters: Small language models are also few-shot learners’, in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2339–2352, Online, (June 2021).
- [35] Valerie A. Schneider, Tina Graves-Lindsay, et al., ‘Evaluation of grch38 and de novo haploid genome assemblies demonstrates the enduring quality of the reference assembly’, *bioRxiv*, (2016).
- [36] Rico Sennrich, Barry Haddow, et al., ‘Neural machine translation of rare words with subword units’, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, Volume 1: Long Papers*, (2016).
- [37] Torgyn Shaikhina and Natasha A. Khovanova, ‘Handling limited datasets with neural networks in medical applications: A small-data approach’, *Artif. Intell. Medicine*, **75**, 51–63, (2017).
- [38] S. T. Sherry, M.-H. Ward, et al., ‘dbSNP: the NCBI database of genetic variation’, *Nucleic Acids Research*, **29**(1), 308–311, (01 2001).
- [39] Yu Sun, Shuohuan Wang, et al., ‘Ernie: Enhanced representation through knowledge integration’, *arXiv preprint arXiv:1904.09223*, (2019).
- [40] Peter M. Visscher, Naomi R. Wray, Qian Zhang, Pamela Sklar, Mark I. McCarthy, Matthew A. Brown, and Jian Yang, ‘10 years of gwas discovery: Biology, function, and translation’, *The American Journal of Human Genetics*, **101**(1), 5–22, (2017).
- [41] Ruohan Wang, Zishuai Wang, et al., ‘Splicefinder: ab initio prediction of splice sites using convolutional neural network’, *BMC Bioinformatics*, **20**(23), 652, (Dec 2019).
- [42] Atsuki Yamaguchi, George Chrysostomou, et al., ‘Frustratingly simple pretraining alternatives to masked language modeling’, in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 3116–3125, (November 2021).
- [43] Dongjie Yang, Zhuosheng Zhang, and Hai Zhao, ‘Learning better masking for better language model pre-training, 2023.
- [44] Meng Yang, Lichao Huang, et al., ‘Integrating convolution and self-attention improves language model of human genome for interpreting non-coding regions at base-resolution’, *Nucleic Acids Research*, **50**(14), e81–e81, (05 2022).
- [45] Tianwei Yue and Haohan Wang, ‘Deep learning for genomics: A concise overview’, *ArXiv*, **abs/1802.00810**, (2018).
- [46] Manzil Zaheer, Guru Guruganesh, et al., ‘Big bird: Transformers for longer sequences’, in *Advances in Neural Information Processing Systems*, volume 33, pp. 17283–17297, (2020).