DEEPGENE: AN EFFICIENT FOUNDATION MODEL FOR GENOMICS BASED ON PAN-GENOME GRAPH TRANSFORMER

A PREPRINT

Xiang Zhang*
School of Computer Science and Engineering
Southeast University
Nanjing, Jiangsu Province
x.zhang@seu.edu.cn

Mingjie Yang
School of Computer Science and Engineering
Southeast University
Nanjing, Jiangsu Province

Xunhang Yin
School of Computer Science and Engineering
Southeast University
Nanjing, Jiangsu Province

Yining Qian
School of Computer Science and Engineering
Southeast University
Nanjing, Jiangsu Province

Fei Sun
School of Life Science and Technology
Southeast University
Nanjing, Jiangsu Province

April 24, 2024

ABSTRACT

Motivation: Decoding the language of DNA sequences is a fundamental problem in genome research. Mainstream pre-trained models like DNABERT-2 and Nucleotide Transformer have demonstrated remarkable achievements across a spectrum of DNA analysis tasks. Yet, these models still face the pivotal challenge of (1) genetic language diversity, or the capability to capture genetic variations across individuals or populations in the foundation models; (2) model efficiency, specifically how to enhance performance at scalable costs for large-scale genetic foundational models; (3) length extrapolation, or the ability to accurately interpret sequences ranging from short to long within a unified model framework.
Results: In response, we introduce DeepGene, a model leveraging Pan-genome and Minigraph representations to encompass the broad diversity of genetic language. DeepGene employs the rotary position embedding to improve the length extrapolation in various genetic analysis tasks. On the 28 tasks in Genome Understanding Evaluation, DeepGene reaches the top position in 9 tasks, second in 5, and achieves the overall best score. DeepGene outperforms other cutting-edge models for its compact model size and its superior efficiency in processing sequences of varying lengths.
Availability and implementation: The datasets and source code of DeepGene are available at GitHub. (https://github.com/wds-seu/DeepGene)

1 Introduction

As a biological language, DNA shares considerable similarities with natural language in many aspects. Presently, the paradigm of modeling human DNA sequences as a natural language holds significant promise [Zhang et al., 2019, Liu et al., 2020, Alharbi and Rashid, 2022]. With the advent of Transformer-based models [Vaswani et al., 2017], numerous large-scale models trained on genetic corpora have emerged. These models employ continuous embeddings to model DNA sequences, demonstrating potential applications across various genomic analysis tasks, including understanding

gene regulation [Li et al., 2023], decoding enhancer complexity [Smith et al., 2023], predicting promoters [Lai et al., 2019] and forecasting gene expression [Avsec et al., 2021], etc.

Foundation models for genomics such as DNABERT [Ji et al., 2021], DNABERT-2 [Zhou et al., 2023], and Nucleotide Transformer [Dalla-Torre et al., 2023] have achieved state-of-the-art performance across a range of genomics tasks. DNABERT utilized the human reference genome GRCh38.p13 [Guo et al., 2017] for pre-training, adopting K-mer-based sequence tokenization. This approach performed outstandingly in tasks such as promoter prediction and the identification of transcription factor binding sites. As a follow-up work, DNABERT-2 expanded its training dataset to include multi-species genomes and switched to Byte Pair Encoding (BPE) [Sennrich et al., 2016] for DNA segmentation. Additionally, DNABERT-2 established the Genome Understanding Evaluation (GUE) benchmark, demonstrating superior performance over other leading foundation models. The Nucleotide Transformer, pre-trained on a combination of human pan-genome and multi-species genomes, standed out for its use of extensive data volumes and model sizes. Besides, numerous other pre-trained models have emerged [Fishman et al., 2023, Nguyen et al., 2023] in modeling DNA sequences.

Despite significant advancements in foundational models for genetics, several challenges remain: (1) **Reference Genome vs. Pan-genome:** DNABERT and DNABERT-2 were primarily trained on reference genomes. However, the pan-genome concept [Sherman and Salzberg, 2020] captures a broader spectrum of genetic variation within a species than a single reference genome can. It encompasses the diversity of the DNA language, including single nucleotide polymorphisms (SNPs), insertions, deletions, and structural variations across diverse individuals or populations. This breadth of diversity enables a more comprehensive understanding of species-specific genetic variation [Baaijens et al., 2022]. (2) **Performance vs. Costs:** While bigger genomic models tend to exhibit improved performance, the incremental gains begin to diminish as the size of the model and computational resource requirements increase. Finding an optimal balance between performance and cost is critical for the efficient training of genomic models on extensive DNA datasets. (3) **Short Sequence vs. Long Sequence:** The accurate decoding of genomic sequences requires an understanding of extensive contextual information spread across thousands of nucleotides [Zhao et al., 2023]. The maximum input sequence size for DNABERT and Nucleotide Transformer was limited to 512bp and 1000bp, respectively. Gena-LM [Fishman et al., 2023] extended the input sequence length to between 4.5k and 36k base pairs with slide attention [Zaheer et al., 2020], showing superior length extrapolation abilities compared to previous models. However, Gena-LM faced challenges in efficiently analyzing short sequences.

In this study, we introduce DeepGene, a Transformer-based architecture with BPE tokenization for DNA segmentation. The novel contributions lie in: (1) DeepGene is trained on a comprehensive dataset comprising 47 ancestrally diverse human pan-genomes. This dataset enables DeepGene to capture the extensive variability of genomic language, providing a more nuanced understanding of genetic diversity; (2) We incorporate the Minigraph representation to efficiently represent the complex graph structures inherent in pan-genomes. This method allows DeepGene to maintain a balance between computational performance and costs, outperforming other cutting-edge genomic models in this respect; (3) Rotary position embedding is applied in DeepGene to accurately learn the relative positions of bases [Huang et al., 2020], which significantly enhances the model's capability for length extrapolation, improving its performance across both short and long DNA sequences. Experiments on a broad range of datasets and tasks have demonstrated exceptional proficiency in interpreting and predicting genomic sequences, showcasing its potential to contribute valuable insights into the complex language of DNA.

The organization of the paper is as follows: Section 2 offers an in-depth description of the pan-genome dataset utilized as the foundational corpus for pre-training, the implementation of the Minigraph model to depict the graph structures within genomic sequences, and the technical details of the DeepGene model. In Section 3, we present the experimental outcomes of DeepGene on GUE among other tasks, engaging in a comparative analysis of its performance relative to other leading models in terms of efficiency and effectiveness. Section 4 provides a summary of our contributions in this research and proposes perspectives for future works.

2 Materials and Methods

2.1 Pan-genome Dataset

2.1.1 Pan-genome and Minigraph Representation

DNABERT and DNABERT-2 were trained using reference genomes, which were unable to capture the full spectrum of genetic variations. The base model of the Nucleotide Transformer was not only trained on a reference genome but was further trained with samples from thousands of diverse human genomes. While this method significantly expanded the dataset, it also introduced a redundancy rate of 98% among the samples [Dalla-Torre et al., 2023]. This high level of redundancy constrained the ability of the Nucleotide Transformer to learn the diversity of genetic language.

The pan-genome has emerged to encompass population diversity in genomic analyses. It offers a richer source of genetic diversity compared to traditional reference genomes. This enhanced diversity enables a more accurate and efficient understanding of the characteristics of genetic language [Sherman and Salzberg, 2020]. Pan-genome rectifies the biases of reference genomes by storing a representative set of diverse haplotypes and their alignments, leading to the creation of several pan-genomic graph models [Andreace et al., 2023] as de Bruijn graphs [Compeau et al., 2011] and variation graphs [Sirén, 2017]. Due to the natural advantages of graph structures, they are currently the most outstanding in handling large amounts of input data [Eizenga et al., 2020]. Pan-genome graph can solve the redundancy issue of pan-genomic data under the multiple sequence alignment format, and allows the model to focus more on the variant parts. De Bruijn graphs utilize k-mers from all haplotypes to form the graph, with each node representing a k-mer and k-1 overlapping bases between two connected nodes. Variation graphs, on the other hand, do not have a fixed base length on each node and use sequence graphs and path lists to store each input haplotype.There are two reasons we choose the variation graphs: (1) De Bruijn graphs exists lots of overlap between nodes because of k-mer tokenization. (2) The variable lengths of nodes in the variation graph make it suitable for further tokenization.

The Human Pangenome Reference Consortium recently presented a draft version of the human pan-genome [Liao et al., 2023]. The pan-genome graph in this article was based on the sequencing results of 47 ancestrally diverse humans and was constructed using the methods of Minigraph [Li et al., 2020], Minigraph-Cactus [Armstrong et al., 2020, Hickey et al., 2023], and the Pan-genome Graph Builder (PGGB) [Liao et al., 2023].

The pan-genome graph mentioned above are all variation graphs. Among them, Minigraph claims that there are no cycles in the graph, no overlap between nodes, and the data is in the rgfa format, which is simple to process. Therefore, we adopt this dataset and its Minigraph representation, which is illustrated in Fig. 1.

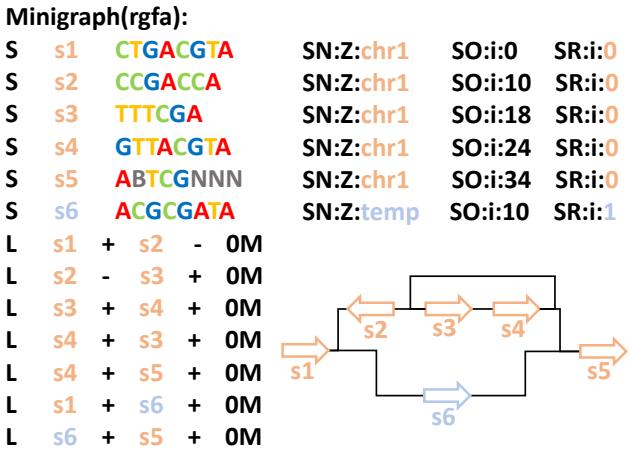

Figure 1: Example of a standard Minigraph representation of pan-genome from genome sequences of two different humans, while the total number of bases is less than 50. The “S" in the figure represents a single node consisting of a segment of nucleotides in a pan-genome. The “L" represents an edge connecting two nodes. “SN" represents the name of sequence from which the node is derived, and “Z" indicates that the content is printable string, including space. The “SO" means the offset, and “i" indicates that the content is signed integer. The “SR" is zero if the sequence of a given node is on the reference genome. rGFA disallows overlap between segments. “0M" in the figure means that the length of the overlapping segments between segments is 0. In this example, thick arrows denote segments and thin lines denote links. Different colors indicate different “SR". The arrow of segment does not indicate the direction, but indicates whether the segment itself or its reverse complent sequence is used.

The “+" in the line of “L" means the segment in node is the same as segment in “S". The “-" in the line of “L" means that the segment in node is the reverse complent segment in “S". For example, in Fig. 1, the segment in “s1" is “CTGACGTA". The segment in “s2" is “CCGACCA". In the first line of “L", the “s1+", “CTGACGTA" and the “s2-" ,“TGGTCGG", namely the reverse complent DNA segment is connected. There are non-ATCG bases in the nucleotide sequence. Common degenerate base codes are used here, such as “B" for “G" or “T" or “C" and “N" for any base.

The structure of Minigraph is a sequence graph, where the nodes represent sequences and are connected by undirected edges. However, the edges on the graph connecting two sequences are ordered because the sequence nodes are differentiated as front and back ends. Therefore, we can consider the direction of the edge as pointing from the back end of one sequence to the front end of another, thus equivalently representing the sequence graph as a directed graph shown as in Fig. 2a.

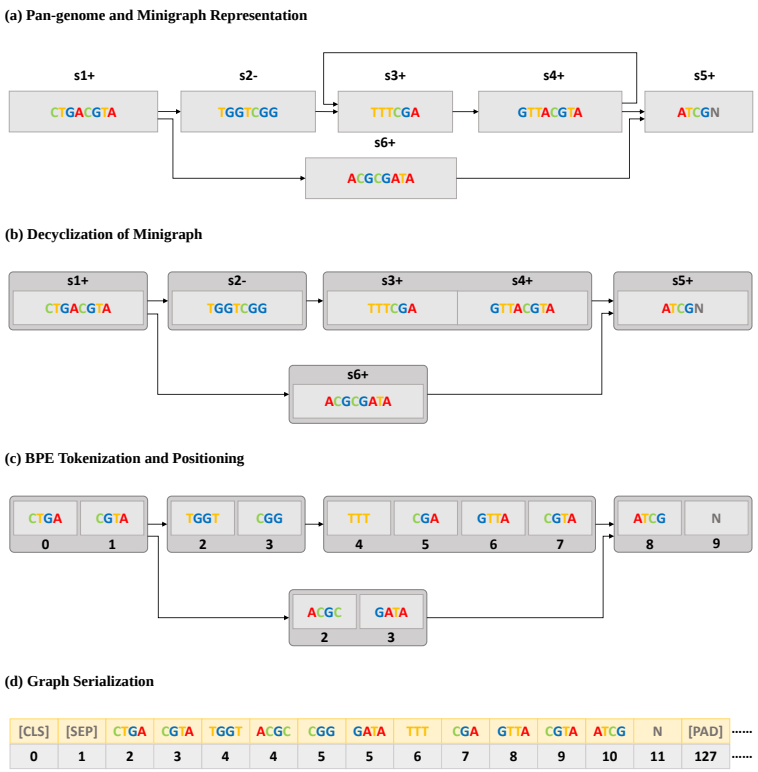

Figure 2: The pipeline of data processing. (a) The result of converting the sequence graph to a directed graph. It is worth noting that “s1+" namely the “s1" segment, “s2-" namely the reverse complent of the “s2" segment; (b) Decyclization with the strong connectivity algorithm; (c) BPE tokenization and adding positional information. (d) Serialize the graph so that it can be put into Transformer-based model training.

2.1.2 Decyclization of Minigraph

Although many sources claim that the graph in the rgfa file of Minigraph is a directed acyclic graph (DAG) [Andreace et al., 2023], we still found a small number of cycles in the graph during the experiment. The position information of each node will be uncertain without removing them, which will be illustrated in detail below. In response to this situation, we used the strongly connected components algorithm to eliminate the cycles and obtained a directed acyclic graph (DAG) shown as in Fig. 2b.

2.1.3 BPE Tokenization and Positioning

Due to the possibility that a node contains thousands of base pairs, it is not feasible to consider a single node as the smallest processing unit. We utilize BPE for tokenization, splitting each node in DAG into multiple tokens. In this way, each token of every node will have positional information, which we calculate in the following manner.

$$position(i,j) = \begin{cases} \min\limits_{k \in P} position(k,0) + len(k) & j \neq 0 \\ position(i,0) + j - 1 & j = 0 \end{cases} \tag{1}$$

The $position(i,j)$ refers to the positional information of the j-th token in node i, and $len(k)$ denotes the total number of tokens in node k. P is the set of predecessor nodes of node i . It is important to note that if a node does not have a predecessor node, then its $pos(i,0)$ is 0. This is illustrated in Fig. 2c.

2.1.4 Graph Serialization

It is known that Transformer-based models have a maximum input length limit, which means that the number of tokens we can process at once is also limited, making it impossible to input an entire connected graph into the model.

Therefore, we sort all tokens in the same connected graph according to their position and partition the subgraphs to ensure that the size of each subgraph is 126 tokens as most. Special tokens such as [CLS], [SEP], [PAD], etc., are added to each subgraph until it is padded to 128 tokens. After adding the special tokens, these tokens also need to have positional information, with the [CLS] position set to 0, the [SEP] position set to 1, and the other [SEP] position set to 127. Additionally, the positions of all other tokens are adjusted accordingly, shifting them collectively to start from 2. This is illustrated in Fig. 2d.

2.1.5 Dataset Statistics

In this way, the graph structure of Minigraph complete the process of serialization, becoming the token and positional sequences that can be input into the model.

The Minigraph we use contains over 5 billion total base pairs, and the number of tokens obtained after segmentation is over 1 billion. Finally, about 10 million subgraphs are obtained by segmenting the token number and padding special tokens to ensure that each subgraph has 128 tokens. Each subgraph is pre-trained as a separate sample, the token of the subgraph and the positional information of each token are passed into the model. In other words, the input has already abstracted the graph information into sequence information, so there is no need to pass the edge information into the model.

2.2 DeepGene Model

2.2.1 Rotary Position Embedding

In order to enable the model to better learn the relative positions of gene sequences rather than absolute positions, this paper employed rotary position embedding [Su et al., 2024] on top of the previous multi-head attention mechanism. Based on BERT [Devlin et al., 2019], the rotary position embedding initially processes the Q and K vectors and incorporates absolute positional information into Q and K. Specifically, after mapping the input to Q and K, the Q and K vectors are rotated by multiplying with a rotary matrix R_m^d. The rotary matrix R_m^d contains positional sequence information. The representation of R_m^d is as follows:

$$R_m^d = \begin{pmatrix} cosm\theta_1 & -sinm\theta_1 & \dots & 0 & 0 \\ sinm\theta_1 & cosm\theta_1 & \dots & 0 & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ 0 & 0 & \dots & cosm\theta_{d/2} & -sinm\theta_{d/2} \\ 0 & 0 & \dots & sinm\theta_{d/2} & cosm\theta_{d/2} \end{pmatrix} \tag{2}$$

The m refers to the position of token, the d refers to the dimension of the hidden state, θ is the constant. After the rotary operation, incorporating the absolute positional information of Q and K into the attention calculation will fuse the attention matrix with relative positional information through the dot product of Q and K. This effectively achieves relative positional encoding in an absolute positional encoding manner. As shown in the following formula, the dot product obtained by multiplying the input vectors at positions m and n by the rotary matrix automatically incorporates the relative positional information of m and n into self-attention.

$$\begin{cases} q_m = R_m^d W_q x_m \\ k_n = R_n^d W_k x_n \end{cases} \tag{3}$$

$$q_m^T k_n = (R_m^d W_q x_m)^T (R_n^d W_k x_n) = x_m^T W_q R_{m-n}^d W_k x_n \tag{4}$$

The essence of positional encoding by rotary is to express relative positional information through absolute position when the data are interacting with attention, thereby allowing the model to learn the relative positions between gene sequence segments, rather than just absolute positions. Experiments have demonstrated that relative positional encoding has achieved better results compared to absolute positional encoding. In addition, it makes the model have length extrapolation, meaning that although the model is pre-trained on short sequences (e.g., 128bp), it can perform equally well on both short and long sequences.

2.2.2 Model Pre-training

The model architecture is shown in Fig. 3. During the pre-training process, the DeepGene model first processes graph data into sequences and then applies masking. After embedding, the input undergoes a RoPE-based multi-head attention

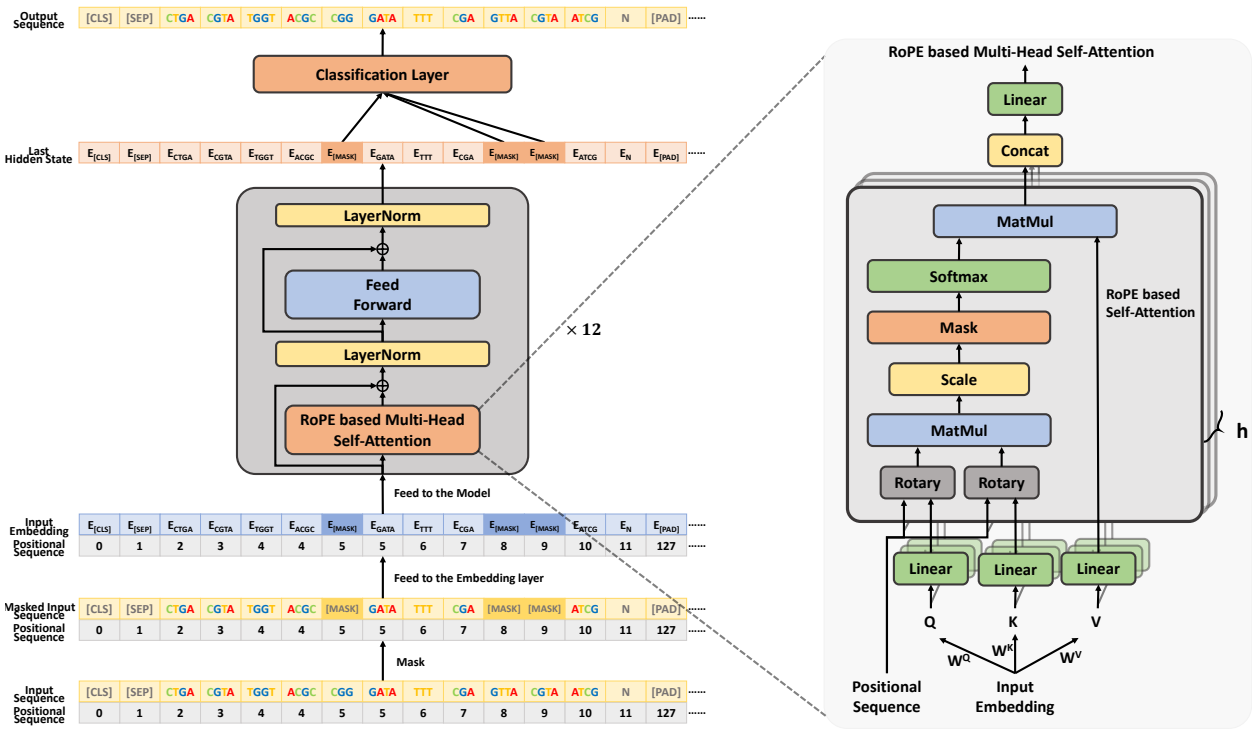

Figure 3: Details of architecture and characteristic of the DeepGene model. The sequence is tokenized by BPE algorithm as the input. The input passes embedding layer and is fed into 12 Rope Transformer blocks to obtain the relative poisition information. The MLM method, which predict the masked token is used to pre-train the model.

module. The detail of this module is shown in figure and illustrated in the rotary position embedding mentioned above. Subsequently, the model applies softmax to the attention matrix A, multiplies it with V, and proceeds with feedforward and layer norm operations. Ultimately, this process yields the representation of the gene sequence. Similar to BERT, the pre-training task adopts the classic MLM task to predict tokens that have been chosen randomly. The proportion of tokens chosen is 15% . Among these tokens, 80% of them are masked, 10% are randomly replaced and 10% remain unchanged.

The pre-training task was conducted for 20 epochs, with a batch size of 2,048. During the first 12 epochs, we implemented a warm-up phase, where the learning rate linearly increased from 0 to 4×10^{-4}. For the subsequent 8 epochs, the learning rate linearly decreased from 4×10^{-4} to 1.33×10^{-4}. We utilized 4 A100 (40G) GPUs for pre-training and completed the process in only four days, significantly faster than other models. We employed a model architecture similar to the BERT base, consisting of 12 layers with a hidden layer dimension of 768.

2.2.3 Model Fine-tuning

After completing pre-training, we fine-tuned the model on various downstream tasks to evaluate its performance. Since the corresponding downstream tasks and baselines are all sequence-based, although we utilized a DAG for pre-training, we conducted fine-tuning on sequences. To some extent, sequences can also be considered a special type of DAG. Due to the relatively small model size, we directly employed full-parameter fine-tuning. We not only conducted fine-tuning experiments on shorter gene sequences for downstream tasks but also, to validate the considerable length extrapolation capability of our model, we fine-tuned it on tasks involving longer sequences. We used the hidden layer vector of the [CLS] token at the beginning of the sequence to represent the DNA sequence. By employing an MLP as the classification head, we performed relevant classification tasks.

3 Experiments and Results

This section mainly describes the details of the experiments. The baseline models like DNABERT-2 are introduced first. Then, the dataset and experimental settings of the downstream task are shown. The performance and cost of DeepGene and baseline models are described in subsection three. Lastly, we discuss the ability of length extrapolation.

3.1 Baselines

DNABERT was the pioneering pre-trained model designed for genome sequences. This model was pre-trained on the human genome, consisting of four variants named DNABERT(3-mer), DNABERT(4-mer), DNABERT(5-mer), and DNABERT(6-mer). Compared to BERT, it has different vocabulary size depending on the choice of k-mer.

Nucleotide Transformer has a much larger size of models and datasets and uses pan-genome dataset. It has achieved state-of-art performance in many DNA analysis tasks. It has four variants too: NT-500M-human, NT-500M-1000g, NT-2500M-1000g and NT-2500M-multi. The first is pre-trained on human reference genome with the model size of 500M. The second is pre-trained on the pan-genome of 1000 humans with the model size of 500M. The third is pre-trained on the pan-genome of 1000 humans with the model size of 2500M. The last one is pre-trained on the genome of 850 species with the model size of 2500M.

DNABERT-2 is the model improved based on DNABERT. It is pre-trained on the multi-species genome. This model uses BPE as the tokenization method, utilizing many computational methods like the relative position embedding. It achieves great performance on many DNA analysis tasks.

3.2 Experiement Datasets and Settings

We have established two tasks, the first is Genomic Understanding and Evaluation (GUE), which refer to the task presented in DNABERT-2, focusing on the ability to complete downstream tasks of genes. The other one is Long Promoter Detection (LPD), evaluating the understanding of model on long DNA sequences.

3.2.1 Genomic Understanding and Evaluation

In order to verify that the base model has learned representations of DNA sequences from the pre-training corpus, we conducted a series of downstream task experiments to explore the model's capabilities across different downstream tasks.The data for the downstream tasks were sourced from DNABERT-2, which proposed the GUE (Genomic Understanding and Evaluation) assessment criteria. The GUE assessment criteria include a range of diverse classification tasks.The tasks encompass seven major categories and a total of 28 subtasks. These tasks cover not only the human genome but also the genomes of various species such as yeast, mice, and viruses. Furthermore, the tasks include both binary classification and multi-class classification. The sequence lengths range from 70bp to 1000bp. This comprehensive setup allows for a more thorough evaluation of the model's ability to model different genes.With the exception of the viral task, which uses f1-score as the evaluation metric, all other tasks use Matthews's Correlation Coefficient (MCC) as the evaluation metric. During fine-tuning, we employed full-parameter fine-tuning with a uniform batch size of 32. The weight decay is 0.01. We conducted warm-up for the first 50 steps of fine-tuning, where the learning rate linearly increased from 0 to 3×10^{-5}, and then maintained a constant learning rate for the remainder of the fine-tuning process. Every 200 steps during fine-tuning, the model was evaluated, and the best-performing model on the validation set across all evaluations was selected for testing.Due to variations in the data scale for different downstream tasks, the total number of fine-tuning steps differed for each task, as indicated in Tabel. 1.

Table 1: **The number of training steps for the following tasks.** Epigenetic Marks Prediction (EMP), Transcription Factor Prediction on the Human genome and the Mouse genome (TF-H and TF-M), Covid Variants Classification (CVC), tata dataset of Promoter Detection (PD-tata), notata and all datasets of Promoter Detection (PD-other), tata dataset of Core Promoter Detection (CPD-tata), notata and all datasets of Core Promoter Detection (CPD-other), and Splice Site Prediction (SSP).

	EMP	TF-M	CVC	TF-H	PD-tata	PD-other	CPD-tata	CPD-other	SSP
Num. Epochs	5	$<$ 2k steps	10	5	10	5	10	5	5

3.2.2 Long Promoter Detection

To validate that our model exhibits superior length extrapolation capability compared to absolute positional encoding models like DNABERT, we further designed experiments to assess length extrapolation ability.We chose the classic downstream task of promoter recognition to conduct the length extrapolation experiment. To obtain the dataset, we adopted the method used by Gena-LM for acquiring data related to promoter recognition downstream tasks. The original data was from EPDNew [Dreos et al., 2013]. We used the EPDNew select tool to fetch human promoter sequences (hg38). To effectively demonstrate the impact of different lengths on the model's capabilities, we selected

three different lengths of promoter sequence fragments for experimentation. Subsequently, we employed the Gena-LM method to generate corresponding negative examples and partitioned the data into training, validation, and testing sets in a 3:1:1 ratio.

The specific settings of the obtained promoter sequence dataset are detailed in Tabel. 2.

Table 2: **The illustration of promoter detection dataset** The data in different lengths are sampled from different positions on the stable sequences. All of these three dataset has the same scale.

Length	Begin	End	Train / Dev/ Test
60bp	-29bp	30bp	
300bp	-249bp	50bp	35518/11839/11839
2000bp	-1000bp	999bp	

The DNABERT model's pre-training corpus has a length of 512 tokens, while our model's pre-training corpus has a length of 128 tokens. However, considering the difference in tokenization methods, DNABERT's 512 tokens use k-mers with overlapping segments, meaning that the number of bases modeled by 512 tokens does not exceed 512bp. In contrast, our model utilizes BPE tokenization, where there is no overlap between tokens, and roughly the same number of bases is covered by 128 tokens. Therefore, the longest sequences encountered during pre-training for both models are essentially equivalent. Thus, conducting length extrapolation experiments between them is reasonable.

Considering that DNABERT has multiple versions, and different k-mers may have a certain influence on length extrapolation capability, we chose to include two versions, 3mer and 6mer, in the experiment. These represent the smallest and largest k-mer versions, respectively. To ensure a fair comparison between DNABERT and DeepGene in terms of sequence representation capability, we used an identical classification head for downstream task evaluation. Specifically, we utilized the feature vector of the [CLS] token in the last hidden layer of the pre-trained model as the feature representation for the entire sequence. This was followed by the connection of an MLP classification head for binary classification tasks. The f1-score is used as evaluation metric. During fine-tuning, we employed full-parameter fine-tuning with a uniform batch size of 128. The weight decay is 0.01. For the first 50 steps of fine-tuning, we implemented warm-up, where the learning rate linearly increased from 0 to 3×10^{-5}, and then maintained a constant learning rate for the remainder of the fine-tuning process. For each downstream task length, we conducted fine-tuning for three epochs, evaluating the model every half epoch, and selecting the best-performing model on the validation set for testing.

3.3 The Balance of Performance and Costs on GUE

We categorized the downstream tasks by type and averaged the results for tasks within the same category, as shown in Tabel. 3.

The table below shows the results for the 28 tasks. It can be observed that our model's performance varies significantly compared to the baseline across different downstream tasks. While we achieved state-of-the-art (SOTA) results in some tasks, our model's performance was mediocre in certain other tasks. The DeepGene model significantly outperforms other models on the Yeast sequence task and also ranks near the top for tasks involving.

Our model performs mediocrely on most human sequences, which may be due to the following reasons: (1) The tasks involving Human sequences are based on data from the human reference genome, while our model is trained on pan-genomic data, leading to a certain bias. (2) During the process of serializing the pan-genomic graph, the serialized corpus does not perfectly correspond to the original pan-genomic graph, resulting in some information loss and introducing noise to a certain extent. However, conversely, the biases and noise present in the data have, to some extent, enhanced the model's generalization capability, leading to improved performance on downstream tasks in other species.

By averaging the results of the 28 downstream tasks and considering various factors such as model size, training data, and other aspects, we can derive the following comprehensive evaluation table (Tabel. 4).

From the table, it is evident that our model achieved comparable results to other models despite having a smaller parameter count and significantly less training data and volume. We achieved first place in 9 tasks and second place in 5 tasks. The overall average score has increased by 0.5% compared to the suboptimal Nucleotide Transformer, while the parameter volume of our model have dropped by 95% compared to the suboptimal model. We believe this demonstrates that training on graph sequences after generalizing the genome to graphs can, to some extent, more effectively utilize this data.

Table 3: **The performance of all the models on the GUE benchmark.**

	Epigenetic Marks Prediction									
	H3	H3K14ac	H3K36me3	H3K4me1	H3K4me2	H3K4me3	H3K79me3	H3K9ac	H4	H4ac
DNABERT(3-mer)	74.15	42.07	48.49	42.95	31.34	28.92	60.12	50.48	78.27	38.60
DNABERT(4-mer)	73.03	41.88	48.03	41.06	30.66	25.31	59.77	51.44	78.28	36.40
DNABERT(5-mer)	73.40	40.68	48.29	40.65	30.67	27.10	59.61	51.11	77.27	37.48
DNABERT(6-mer)	73.10	40.06	47.25	41.44	32.27	27.81	61.17	51.22	79.26	37.43
NT-500M-human	69.67	33.55	44.14	37.15	30.87	24.06	58.35	45.81	76.17	33.74
NT-500M-1000g	72.52	39.37	45.58	40.45	31.05	26.16	59.33	49.29	76.29	36.79
NT-2500M-1000g	74.61	44.08	50.86	43.10	30.28	30.87	61.20	52.36	79.76	41.46
NT-2500M-multi	**78.77**	56.20	**61.99**	**55.30**	36.49	40.34	64.70	56.01	**81.67**	49.13
DNABERT-2	78.27	52.57	56.88	50.52	31.13	36.27	67.39	55.63	80.71	50.43
DeepGene	76.69	**59.69**	58.63	49.55	**50.54**	**51.22**	67.54	**61.61**	78.56	53.11
DeepGene(top-3)	77.34	**59.69**	58.63	49.55	**50.54**	**51.22**	**67.86**	**61.61**	78.89	**56.47**

	Transcription Factor Prediction(Human)					Transcription Factor Prediction(Mouse)				
	0	1	2	3	4	0	1	2	3	4
DNABERT(3-mer)	67.95	70.90	60.51	53.03	69.76	42.31	79.10	69.90	55.40	41.97
DNABERT(4-mer)	67.90	73.05	59.52	50.37	71.23	49.42	79.95	72.62	51.79	44.13
DNABERT(5-mer)	66.97	69.98	59.03	52.95	69.26	42.45	79.32	62.22	49.92	40.34
DNABERT(6-mer)	66.84	70.14	61.03	51.89	70.97	44.42	78.94	71.44	44.89	42.48
NT-500M-human	61.59	66.75	53.58	42.95	60.81	31.04	75.04	61.67	29.17	29.27
NT-500M-1000g	63.64	70.17	52.73	45.24	62.82	39.26	75.49	64.70	33.07	34.01
NT-2500M-1000g	66.31	68.30	58.70	49.08	67.59	48.31	80.02	70.14	42.25	43.40
NT-2500M-multi	66.64	70.28	58.72	51.65	69.34	**63.31**	83.76	71.52	69.44	47.07
DNABERT-2	**71.99**	**76.06**	**66.52**	**58.54**	**77.43**	56.76	**84.77**	79.32	66.47	**52.66**
DeepGene	68.48	69.87	63.99	52.90	72.82	49.52	82.04	81.42	**71.47**	45.38
DeepGene(top-3)	70.01	71.11	64.51	53.63	73.39	49.52	82.04	**82.56**	**71.47**	46.18

	Promoter Detection			Core Promoter Detection			Virus	Splice
	all	notata	tata	all	notata	tata	Covid	Reconstruct
DNABERT(3-mer)	90.44	93.61	69.83	**70.92**	69.82	78.15	62.23	84.14
DNABERT(4-mer)	89.54	92.65	66.78	69.00	70.04	74.25	59.87	84.05
DNABERT(5-mer)	90.16	92.45	69.51	69.48	69.81	76.79	50.46	84.02
DNABERT(6-mer)	90.48	93.05	61.56	68.90	70.47	76.06	55.50	84.07
NT-500M-human	87.71	90.75	78.07	63.45	64.82	71.34	50.82	79.71
NT-500M-1000g	89.76	91.75	78.23	66.70	67.17	73.52	52.06	80.97
NT-2500M-1000g	90.95	93.07	75.80	67.39	67.46	69.66	66.73	85.78
NT-2500M-multi	**91.01**	94.00	**79.43**	70.33	**71.58**	72.97	**73.04**	**89.35**
DNABERT-2	86.77	**94.27**	71.59	69.37	68.04	74.17	71.02	84.99
DeepGene	86.01	92.61	63.13	66.78	66.57	77.53	70.91	82.25
DeepGene(top-3)	86.01	93.15	64.85	66.78	66.74	**79.62**	70.91	82.71

3.4 The Length Extrapolation on LPD

The final experimental results are shown in Fig. 5.

(a) is the experimental result, and it can be seen that, when averaging the results of the three lengths, our model performs the best. Although in the 60bp and 300bp length tasks, our model's performance compared to DNABERT is not satisfactory. Notably, in the results for 2000bp, our model's performance far exceeds that of DNABERT, indicating that we are significantly superior to DNABERT in modeling longer distances.

(b) The trend chart shows how the performance changes with length and provides further insights. With the gradual increase in length, it is evident that DNABERT's performance continues to improve, indicating that longer sequences indeed contribute to promoter recognition. However, it is also observed that the 3mer, which performs the best at 60bp, performs the worst at 2000bp; conversely, what performs the worst at 60bp excels at 2000bp. Therefore, from the

Table 4: **The scale of model and data of each mainstream model and their average performance.** The comparative table shows that our model achieves great performance that is on the same level with current mainstream models while using smaller parameters and pre-train data.

Model	Num. Params. ↓	Trn. Tokens	Pre-Trn. Datasize (bp)	Num. Top-2 ↑	Ave. Scores ↑
DNABERT(3-mer)	86M	122B	3B	1/1	61.62
DNABERT(4-mer)	86M	122B	3B	0/1	61.14
DNABERT(5-mer)	87M	122B	3B	0/0	60.05
DNABERT(6-mer)	89M	122B	3B	0/1	60.51
NT-500M-human	480M	50B	3B	0/0	55.43
NT-500M-1000g	480M	50B	115B	0/1	58.23
NT-2500M-1000g	2537M	300B	115B	0/2	61.41
NT-2500M-multi	2537M	300B	174B	10/9	66.93
DNABERT-2	117M	262B	32.5B	8/5	66.80
DeepGene	85M	26B	5B	5/4	66.82
DeepGene(top-3)	85M	26B	5B	9/5	**67.43**

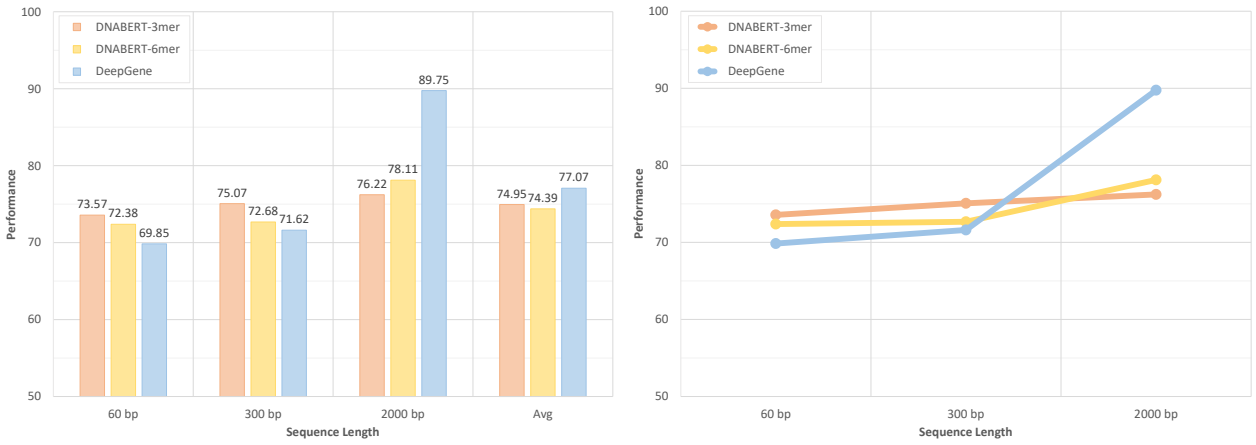

Figure 4: The result of DeepGene and DNABERT on the promoter detection data in different lengths. It is obvious that with the length increasing, the performance of the DeepGene model soar while DNABERT increases slowly. On the 2000bp task, DeepGene achieves more than 10% scores compared with DNABERT 3-mer and 6-mer.In addition, the deepgene model achieves better performance on average compared to the DNABERT.

perspective of length extrapolation, the 6mer is slightly better than the 3mer, and our model is far superior to both. This corroborates that our use of relative positional encoding is indeed more effective in modeling relatively longer sequences compared to DNABERT's absolute positional encoding. Particularly, when the sequence length exceeds the maximum length encountered during pre-training, the disparity becomes even more pronounced, showcasing the model's length extrapolation capability.

3.5 Discussion

However, there are still some areas for improvement: (1) There is certainly room for improvement in the BPE tokenization method, and whether there is a more biologically meaningful tokenization method; (2) The edges of the pan-genomic graph of Minigraph are sparse, and the number of haplotypes contained in the Minigraph we use is limited. (3) Only human pan-genomic data is used, whether using multi-spiece pan-genomic data is better; (4) We only verified that the model has a certain length extrapolation capability, but for longer sequences, relying solely on the model's extrapolation capability will be unrealistic. Necessary modifications to the model from the attention layer are strongly needed such as sparse attention.

4 Conclusion and Future Work

Existing gene language models have paid little attention to pan-genomic data, and even when using pan-genomic corpora, they are based on multiple sequence alignment. Given this situation, we use the Minigraph and extracted token and positional information from the pan-genomic graph data through a series of operations including DAG construction, BPE Tokenization, and Serialization.

In order to endow the model with better length extrapolation capability we selected the roformer model based on relative positional encoding as the basis for pre-training. Ultimately, under the premise of less training data and shorter training time, we achieved performance comparable to other baseline models on the GUE evaluation criteria. We achieved top-2 results in half of the tasks. Furthermore, through LPD experiments, it was indeed proven that our model has better length extrapolation capability compared to DNABERT with absolute positional encoding.

In future work, the following method is significant to investigate: (1) Pan-genome dataset with more individuals is recommended, as the Nucleotide Transformer uses pan-genome dataset with more than one thousand individuals for pre-train. (2) The slide attention mechanism linear time complexity is needed for extremely long sequences. (3) For better results, it is reasonable to merge the genome from other species to enhance the model ability. With the rapid development of genomics, we hope that the techniques we introduce will promote the application, development and improvement of large-scale language models in the field of genomics.

5 Acknowledgments

We sincerely thank Professor Du Zhenfang, Ms. Sun Jinghan, Mr. Xia Qi, and Ms. Yang Wenjie for their invaluable suggestions on this work.

References

Zhiqiang Zhang, Yi Zhao, Xiangke Liao, et al. Deep learning in omics: a survey and guideline. *Briefings in Functional Genomics*, 18(1):41–57, February 2019. ISSN 2041-2649, 2041-2657. doi:10.1093/bfgp/ely030. URL https://academic.oup.com/bfg/article/18/1/41/5107348.

Jianxiao Liu, Jiying Li, Hai Wang, et al. Application of deep learning in genomics. *Science China Life Sciences*, 63 (12):1860–1878, December 2020. ISSN 1674-7305, 1869-1889. doi:10.1007/s11427-020-1804-5. URL https://link.springer.com/10.1007/s11427-020-1804-5.

Wardah S. Alharbi and Mamoon Rashid. A review of deep learning applications in human genomics using next-generation sequencing data. *Human Genomics*, 16(1):26, July 2022. ISSN 1479-7364. doi:10.1186/s40246-022-00396-x. URL https://doi.org/10.1186/s40246-022-00396-x.

Ashish Vaswani, Noam Shazeer, Niki Parmar, et al. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. URL https://proceedings.neurips.cc/paper/7181-attention-is-all.

Zhongxiao Li, Elva Gao, Juexiao Zhou, et al. Applications of deep learning in understanding gene regulation. *Cell Reports Methods*, 3(1), 2023. URL https://www.cell.com/cell-reports-methods/pdf/S2667-2375(22)00289-2.pdf. Publisher: Elsevier.

Gabrielle D. Smith, Wan Hern Ching, Paola Cornejo-Páramo, et al. Decoding enhancer complexity with machine learning and high-throughput discovery. *Genome Biology*, 24(1):116, May 2023. ISSN 1474-760X. doi:10.1186/s13059-023-02955-4. URL https://genomebiology.biomedcentral.com/articles/10.1186/s13059-023-02955-4.

Hong-Yan Lai, Zhao-Yue Zhang, Zhen-Dong Su, et al. iProEP: A Computational Predictor for Predicting Promoter. *Molecular Therapy - Nucleic Acids*, 17:337–346, September 2019. ISSN 2162-2531. doi:10.1016/j.omtn.2019.05.028. URL https://www.cell.com/molecular-therapy-family/nucleic-acids/abstract/S2162-2531(19)30161-1. Publisher: Elsevier.

Žiga Avsec, Vikram Agarwal, Daniel Visentin, et al. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature Methods*, 18(10):1196–1203, October 2021. ISSN 1548-7091, 1548-7105. doi:10.1038/s41592-021-01252-x. URL https://www.nature.com/articles/s41592-021-01252-x.

Yanrong Ji, Zhihan Zhou, Han Liu, et al. DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome. *Bioinformatics*, 37(15):2112–2120, August 2021. ISSN 1367-4803, 1460-2059. doi:10.1093/bioinformatics/btab083. URL https://academic.oup.com/bioinformatics/article/37/15/2112/6128680.

Zhihan Zhou, Yanrong Ji, Weijian Li, et al. DNABERT-2: Efficient Foundation Model and Benchmark For Multi-Species Genome, June 2023. URL http://arxiv.org/abs/2306.15006. arXiv:2306.15006 [cs, q-bio].

Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, et al. The Nucleotide Transformer: Building and Evaluating Robust Foundation Models for Human Genomics, March 2023. URL https://www.biorxiv.org/content/10.1101/2023.01.11.523679v2. Pages: 2023.01.11.523679 Section: New Results.

Yan Guo, Yulin Dai, Hui Yu, et al. Improvements and impacts of GRCh38 human reference on high throughput sequencing data analysis. *Genomics*, 109(2):83–90, March 2017. ISSN 0888-7543. doi:10.1016/j.ygeno.2017.01.005. URL https://www.sciencedirect.com/science/article/pii/S0888754317300058.

Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural Machine Translation of Rare Words with Subword Units, June 2016. URL http://arxiv.org/abs/1508.07909. arXiv:1508.07909 [cs].

Veniamin Fishman, Yuri Kuratov, Maxim Petrov, et al. GENA-LM: A Family of Open-Source Foundational Models for Long DNA Sequences. *bioRxiv*, pages 2023–06, 2023. URL https://www.biorxiv.org/content/10.1101/2023.06.12.544594.abstract. Publisher: Cold Spring Harbor Laboratory.

Eric Nguyen, Michael Poli, Marjan Faizi, et al. HyenaDNA: Long-Range Genomic Sequence Modeling at Single Nucleotide Resolution, November 2023. URL http://arxiv.org/abs/2306.15794. arXiv:2306.15794 [cs, q-bio].

Rachel M. Sherman and Steven L. Salzberg. Pan-genomics in the human genome era. *Nature Reviews Genetics*, 21(4):243–254, April 2020. ISSN 1471-0064. doi:10.1038/s41576-020-0210-7. URL https://www.nature.com/articles/s41576-020-0210-7. Publisher: Nature Publishing Group.

Jasmijn A. Baaijens, Paola Bonizzoni, Christina Boucher, et al. Computational graph pangenomics: a tutorial on data structures and their applications. *Natural Computing*, 21(1):81–108, March 2022. ISSN 1567-7818, 1572-9796. doi:10.1007/s11047-022-09882-6. URL https://link.springer.com/10.1007/s11047-022-09882-6.

Liang Zhao, Xiaocheng Feng, Xiachong Feng, et al. Length Extrapolation of Transformers: A Survey from the Perspective of Position Encoding, December 2023. URL http://arxiv.org/abs/2312.17044. arXiv:2312.17044 [cs].

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, et al. Big bird: Transformers for longer sequences. *Advances in neural information processing systems*, 33:17283–17297, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html.

Zhiheng Huang, Davis Liang, Peng Xu, et al. Improve Transformer Models with Better Relative Position Embeddings, September 2020. URL http://arxiv.org/abs/2009.13658. arXiv:2009.13658 [cs].

Francesco Andreace, Pierre Lechat, Yoann Dufresne, et al. Construction and representation of human pangenome graphs. preprint, Bioinformatics, June 2023. URL http://biorxiv.org/lookup/doi/10.1101/2023.06.02.542089.

Phillip E. C. Compeau, Pavel A. Pevzner, and Glenn Tesler. How to apply de Bruijn graphs to genome assembly. *Nature Biotechnology*, 29(11):987–991, November 2011. ISSN 1546-1696. doi:10.1038/nbt.2023. URL https://www.nature.com/articles/nbt.2023. Publisher: Nature Publishing Group.

Jouni Sirén. Indexing Variation Graphs. In *2017 Proceedings of the Meeting on Algorithm Engineering and Experiments (ALENEX)*, Proceedings, pages 13–27. Society for Industrial and Applied Mathematics, January 2017. doi:10.1137/1.9781611974768.2. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611974768.2.

Jordan M. Eizenga, Adam M. Novak, Jonas A. Sibbesen, et al. Pangenome Graphs. *Annual Review of Genomics and Human Genetics*, 21(1):139–162, August 2020. ISSN 1527-8204, 1545-293X. doi:10.1146/annurev-genom-120219-080406. URL https://www.annualreviews.org/doi/10.1146/annurev-genom-120219-080406.

Wen-Wei Liao, Mobin Asri, Jana Ebler, et al. A draft human pangenome reference. *Nature*, 617(7960):312–324, May 2023. ISSN 1476-4687. doi:10.1038/s41586-023-05896-x. URL https://doi.org/10.1038/s41586-023-05896-x.

Heng Li, Xiaowen Feng, and Chong Chu. The design and construction of reference pangenome graphs with minigraph. *Genome Biology*, 21(1):265, December 2020. ISSN 1474-760X. doi:10.1186/s13059-020-02168-z. URL https://genomebiology.biomedcentral.com/articles/10.1186/s13059-020-02168-z.

Joel Armstrong, Glenn Hickey, Mark Diekhans, et al. Progressive Cactus is a multiple-genome aligner for the thousand-genome era. *Nature*, 587(7833):246–251, 2020. URL https://www.nature.com/articles/s41586-020-2871-y. Publisher: Nature Publishing Group UK London.

Glenn Hickey, Jean Monlong, Jana Ebler, et al. Pangenome graph construction from genome alignments with Minigraph-Cactus. *Nature biotechnology*, pages 1–11, 2023. URL https://www.nature.com/articles/s41587-023-01793-w. Publisher: Nature Publishing Group US New York.

Jianlin Su, Murtadha Ahmed, Yu Lu, et al. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. URL https://www.sciencedirect.com/science/article/pii/S0925231223011864. Publisher: Elsevier.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, et al. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding, May 2019. URL http://arxiv.org/abs/1810.04805. arXiv:1810.04805 [cs].

René Dreos, Giovanna Ambrosini, Rouayda Cavin Périer, et al. EPD and EPDnew, high-quality promoter resources in the next-generation sequencing era. *Nucleic Acids Research*, 41(D1):D157–D164, January 2013. ISSN 0305-1048. doi:10.1093/nar/gks1233. URL https://doi.org/10.1093/nar/gks1233.