DNAGPT: A Generalized Pretrained Tool for Multiple DNA Sequence Analysis Tasks

Daoan Zhang[1,4], Weitong Zhang[3], Bing He[2], Jianguo Zhang[1*], Chenchen Qin[2*], Jianhua Yao[2*]

[1]Southern University of Science and Technology.
[2]Tencent AI Lab, Shenzhen, China.
[3]City University of Hong Kong.
[4]University of Rochester.

*Corresponding author(s). E-mail(s): zhangjg@sustech.edu.cn; chenchenqin@tencent.com; yaojianhua@tencent.com;
Contributing authors: dzhang52@ur.rochester.edu; weitzhang6-c@my.cityu.edu.hk; owenbhe@tencent.com;

Abstract

The success of the GPT series proves that GPT can extract general information from sequences, thereby benefiting all downstream tasks. This motivates us to use pre-trained models to explore the hidden information in DNA sequences. However, data and task requirements in DNA sequence analysis are complexity and diversity as DNA relevant data includes different types of information, such as sequences, expression levels, etc, while there is currently no model specifically designed for these characteristics. Hereby, we present DNAGPT, a generalized foundation model pre-trained on over 10 billion base pairs from 9 species which can be fine-tuned for any DNA sequence analysis task. Our model can simultaneously process or output DNA sequences and numbers. In addition, our unique token design allows users to design prompts according to their own task requirements, making it applicable to any type of task. We have evaluated our model on classification, regression, and generation tasks. We demonstrate that DNAGPT benefits from pre-training, and therefore can bring performance gains to any downstream task. Our model is not only a new attempt in the field of genomes analysis, but also provides a new direction for the application of foundation models in biology.

Keywords: DNA, Generative Pretrained Transformer, DNAGPT, Sequence analysis

1 Introduction

Originated from the same ancestor, different species have strong unities at different levels, from cell architecture [1], genetic material (DNA and RNA) [2], the central dogma principle [3] to the genetic codes and codons used. For this reason, it is substantial to build a generalized model that enables learning universal knowledge across species, which subserves versatile downstream tasks. As the most fundamental sequence information in biology, DNA sequences [4] contain rich, undiscovered biological information, especially those with large non-coding regions [5] that remain unexplored and are particularly worth investigating. There are diverse biological tasks involving DNA sequences, such as functional sequential regions discovery [6, 7] and specific sequence generation [8].

Recently, the emergence of foundation models [9–11] has revolutionized natural language understanding [12] by pre-training generalized models on large-scale datasets that can be tuned to accommodate various downstream tasks. However, compared to natural language understanding, DNA sequence analysis has a wider variety of task forms as well as data types which increase the difficulty in training a general model [13–16]. The previous attempt, DNABERT [17], involved pre-training on the human reference genome followed by fine-tuning on the downstream datasets, thereby enabling a single pre-trained model to generalize to different types of data. While this model can only handle tasks related to genomic signals and regions (GSR)[18–20] and cannot be utilized for other types of tasks. A failed example task type would be the regression of mRNA abundance from the DNA sequence[21]. In this context, the model needs to simultaneously input both the DNA sequence and the mRNA half-life values in order to output the corresponding mRNA abundance values. Another unsuccessful example is the generation of pseudo genomes [8], which requires the model to produce pseudo DNA sequences for research purpose that preserve certain biological properties and features found in natural genomic sequences. Unifying these intricate and diverse data types and task paradigms can reduce unnecessary algorithm design effort while allowing more tasks to benefit from pre-training, further paving the way for more profound discoveries and insights in DNA sequence analysis.

In this paper, we introduce a pre-training of multi-species DNA sequences and propose a symbolic language to unify the diverse tasks in DNA sequence analysis. We present DNAGPT, a generalized pretrained tool for DNA sequence analysis. As shown in in Figure. 1 **a** and **b**, DNAGPT utilizes reference genomes [22] from 9 species (*Arabidopsis_thaliana, Caenorhabditis_elegans, Bos_taurus, Danio_rerio, Drosophila_melanogaster, Escherichia_coli_gca_001721525, Homo_sapiens, Mus_musculus, Saccharomyces_cerevisiae*) for pre-training, with a total data size exceeding 10 billion base pairs (bps). During pre-training, in addition to the auto-regression task, we also imposed guanine-cytosine (GC) content prediction and sequence order prediction tasks based on the characteristics of DNA sequences. To generalize the fundamental knowledge learned during the pre-training phase to various downstream tasks [23, 24], we designed a symbolic language to encode various downstream tasks into sequence, thus unifying the paradigmatic differences between different tasks. We finetune DNAGPT towards a diverse panel of downstream tasks relevant to DNA sequence. Experimental results demonstrate that DNAGPT not only

benefits multiple downstream tasks across different species but also can handle various types of downstream tasks.

In summary, DNAGPT enhances DNA analysis tasks from different species by incorporating reference genomes from multiple species during pre-training. Additionally, the fine-tuning strategy based on the symbolic language enables DNAGPT to handle DNA analysis tasks with various paradigms, allowing for a seamless switching between downstream tasks without adding any training heads, making a wider application scope of pre-training models in the field of biology.

2 DNAGPT architecture

2.1 Model structure

We utilized a transformer-based [25] auto-regressive [26] decoder with the masked self-attention [27] module to train DNAGPT. To better deal with numerical information, we co-pretrain the DNA sequence and numbers end to end in a single model. Detailed network structure is presented in Figure. 1 **c**. Our model can deal with both discrete and continuous tokens where discrete tokens indicate the encoded DNA sequence and the continuous tokens indicate the encoded numbers. The sampled DNA sequence is first processed into a string of non-overlapped k-mers token input, then sent into the *Sequential Embedding Layer* to be encoded as embeddings. The numbers are sent directly into a *Numerical Embedding Layer* to be encoded as embeddings co-training with the DNA embeddings. Then we concatenate both of the two kinds of embeddings and send them into a GPT. The outputs of the GPT are also separated into two types of embeddings and processed by the *Classification Head* to classify different tokens and *Regression Head* to generate numbers. The structure of those heads is presented in Figure. 1 **d**.

2.2 Design of symbolic language

Currently, most DNA pre-training methods [17] simply use strategies from natural languages and do not consider the characteristics of DNA sequence and specific biological tasks in the model design. DNA sequence has no organizational structure as nature languages, which can be hierarchically divided into words, sentences, paragraphs, and other structures. The current data sampling strategy for pre-training models is non-interval and maximizes input length sampling, without considering that the input length of DNA sequences in the downstream tasks is not fixed. This leads to a significant difference in training paradigm between pre-training and downstream tasks. We first design a symbolic language architecture based on DNA sequences, where bps are treated as the smallest unit and non-overlapped k-mers instead of traditional kmers [28] are sampled to generate DNA words. To build up the sentence structure, we combine DNA words of variable lengths to form DNA sentences, and then construct a sentence-level pre-training task based on the sentences structure. After that, DNA sentences of variable lengths are also integrated to form the DNA paragraphs, where the length of a DNA paragraph is the maximum sequence length input into the GPT model.

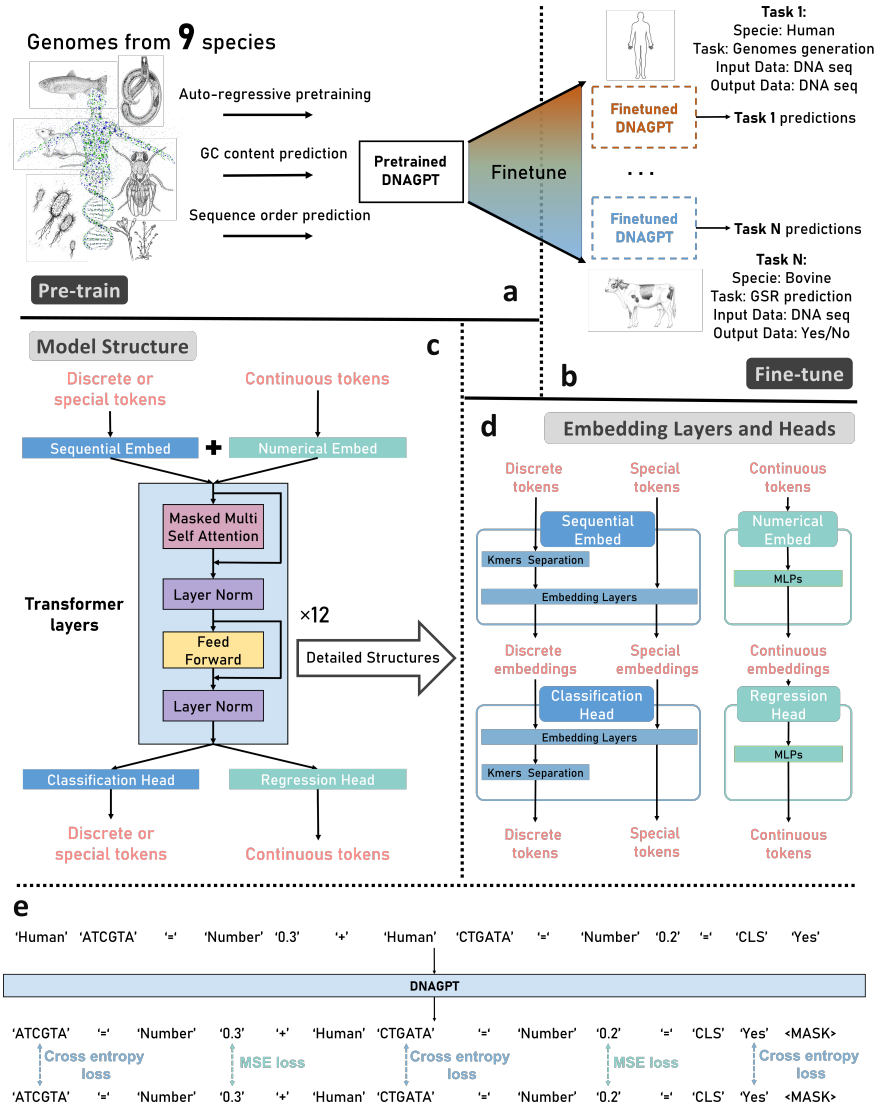

Fig. 1 Schematic of DNAGPT. **a**. The pre-training stage of DNAGPT. We utilize genomes from 9 species and design three pre-training tasks to jointly pre-train DNAGPT. **b**. The finetune stage of DNAGPT. After fine-tuning on the downstream task-related datasets, DNAGPT is able to handle specific tasks. Moreover, DNAGPT supports downstream tasks from different species, as well as various task and data formats. **c**. Model structure of DNAGPT. Different types of tokens are processed separately by different encoding heads, and then combined together as the input for the backbone. **d**. Details of the embedding layers and decoding heads. The figure illustrates the zoom-in view of different encoding heads. When processing the input data, we use different heads for mapping according to the data types. **e**. Model inputs, outputs and ground truth of DNAGPT. These tokens were split according to different properties into two cases using cross-entropy loss and mean squared error (MSE) loss.

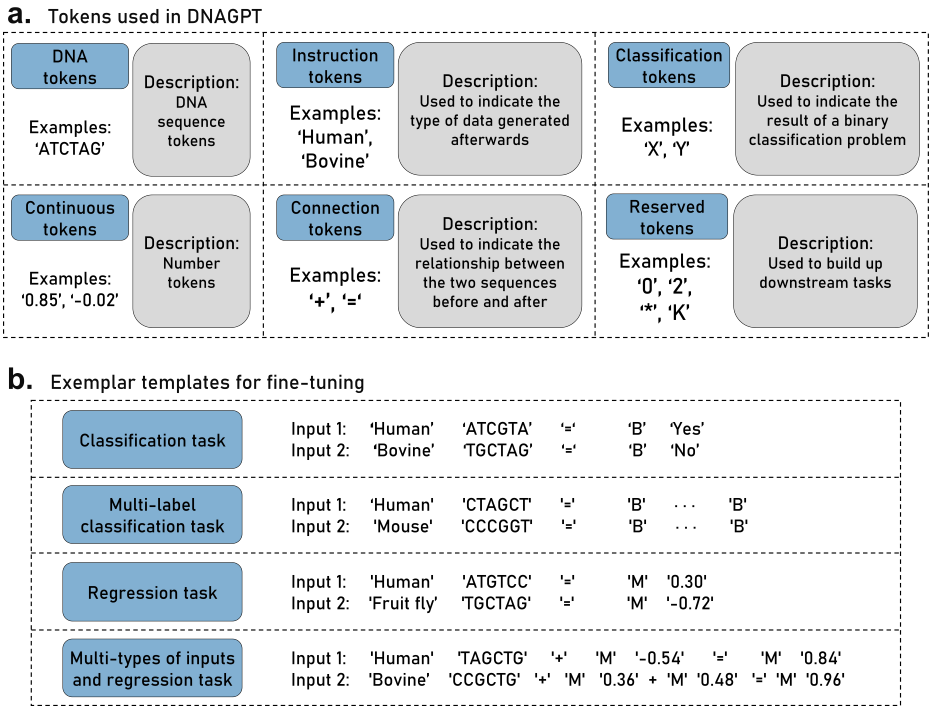

Fig. 2 Symbolic language of DNAGPT. **a**. Tokens used in DNAGPT. **b**. Exemplar templates of the input and label in DNAGPT pre-training and finetuning.

To build a hierarchical organization of DNA sequences, we design a symbolic language architecture. As shown in Figure. 2 **a**, the regular input and output tokens are *Discrete tokens* and *Continous tokens* which represent the DNA sequences and numbers respectively. Instruction tokens are used to give a prompt to the model about what are the next sequence of the tokens should the model output. Take an example, $'$Human$''$AATAAA$'$ indicates we encode a **human** AATAAA polyadenylation signals and $'$Bovine$''$AATAAA$'$ indicates we encode a **bovine** AATAAA polyadenylation signals. Similarly, $'$M$''$0.3155> indicates that we encode a number into the model and in $'$B$''$X$'$, $'$B$'$ is the instruction token of the binary classification where the *Classification tokens* $'$Yes$'$ indicates 'Yes' and $'$No$'$ indicates 'No'. Furthermore, to better construct connections, we use *Connection tokens* to form the connections of two series of tokens, where $'+'$ represent an aggregation of the two series of tokens and $'='$ represent a relation of input and output. Specifically, when we want to predict the expression level of mRNA from both DNA sequence and the mRNA half-life values, we can encode the inputs into $'$Human$''$ATCGTC$''+''$M$''$-0.3484$''='$M$''$0.9854$'$. This sequence indicates that we hope the model can generate the information from both of the $'$ATCGTC$'$ sequence and input numbers $'$-0.3484$'$ to output the result numbers $'$0.9854$'$. The reserved tokens includes numbers from $'0'$ to $'9'$, some unused uppercase letters like $'$K$'$, $'$L$'$, etc. and some special symbols like $'*'$ and $'/'$, etc. These reserved tokens can

be used to build up more exclusive tasks for DNA sequence analysis. The complete token list will be presented in the Figure. S1.

3 Multi-species pre-training

We pre-trained two versions of DNAGPT, one using the human reference genome and the other using reference genomes from all 9 species, named DNAGPT-H and DNAGPT-M, respectively.

3.1 Pre-training dataset

In order to integrate DNA sequence information from multiple species and allow downstream tasks to benefit from cross-species information, We jointly trained DNAGPT on DNA sequences from 9 species [22], including reference genomes from *Arabidopsis_thaliana, Caenorhabditis_elegans, Bos_taurus, Danio_rerio, Drosophila_melanogaster, Escherichia_coli_gca_001721525, Homo_sapiens, Mus_musculus, Saccharomyces_cerevisiae*

3.2 Pre-training tasks

Based on the proposed symbolic language, We can design the pre-training tasks for DNAGPT according to the characteristics of DNA sequence. We build up three tasks including the normal pre-training task of GPT and two specialized tasks for DNA sequence analysis.

Next token prediction

Next token prediction [29] is a classical pre-training task in NLP. GPT leverages this technique which can predict the next possible token based on the previous tokens. Recently, by adding more parameters and more training data, GPT-3 and GPT-4 demonstrate remarkable performance on various tasks. In DNAGPT, we also use the next token prediction strategy as the fundamental pre-training task.

Guanine-cytosine content prediction

To better understanding the input DNA sequence, we design a guanine-cytosine (GC) content prediction task. GC content plays a crucial role in transcriptome analysis as it can provide essential information about genome structure, such as structural variations [30] and transcriptional activity [31, 32]. In this task, we introduced numerical encoding by inputting the GC content as a number into DNAGPT. This approach allows for joint training of numerical data and sequences, enabling DNAGPT to adapt to a wider variety of data types and downstream tasks. Simultaneously, the DNA sequence for predicting the GC content was set to dynamic length. Specifically, we randomly inserted the GC content prediction task into the entire paragraph. The advantage of this approach is that it allows the model to learn a dynamic receptive field, enabling it to more flexibly adapt to different types of downstream tasks.

Sequence order prediction

We notice that the sequence order of DNA plays a significant role in gene expression [33] and transcription [34, 35]. Signals such as TATA box [36] and AATAAA PAS [37] often have a fixed order to guide expression or transcription. Therefore, we designed a sequence order prediction task. As DNA molecules have a double helix structure, composed of two complementary strands, we then randomly reverse the sequence and let the model predict whether our sequence has been reversed or not. This approach enables our model to focus on crucial, order-sensitive sequences in the input, thereby providing more heuristic information for downstream tasks.

Meanwhile, unlike the global attention in the BERT [38], GPT models have unidirectional attention [39], i.e., the model can only infer and generate tokens from left to right. By reversing DNA sequences, the model can predict from a global perspective when inferring tokens, rather than just relying on the forward sequence.

Pre-training Loss

Thus, as shown in Figure. 1. **e**, we illustrate the model input, output, and ground truth for DNAGPT during pre-training. When calculating the loss for outputs of task next token prediction and sequence order prediction, we used cross-entropy loss, while for task GC ratio prediction, we employed mean squared error (MSE) loss.

4 Genomic signals and regions (GSR) recognition

4.1 DNAGPT is able of recognizing GSRs from any species.

Recognition of various genomic signals and regions (GSR) form DNA sequence is essential to the understanding of genomes. To address this issue, we finetune and evaluate our model on the recognition of polyadenylation signals (PAS) and translation initiation sites (TIS) of different organisms: human, mouse, bovine and fruit fly. To be specific, we follow the processing procedure in DeepGSR [18]. The DNA sequence lengths are set to 603 and 606 respectively for TIS and PAS. Then DeepGSR extracted 20,933, 18,693, 12,082, and 27,203 true PAS data; and 28,244, 25,205, 17,558, and 30,283 true TIS for human, mouse, bovine, and fruit fly, respectively. Details of the datasets are depicted in Chapter S1.2.

The recognition of GSR can be considered as a binary classification task. We evaluate the DNAGPT-H and DNAGPT-M on the recognition of both PAS (AATAAA variant and all variants) and TIS (with the ATG signal) in the human genome. We present the accuracy metric in Figure. 3 **a**, which shows that our model can steadily outperform the previous state-of-the-art methods. We further provide additional metric results in the Table. S2 for a more comprehensive evaluation. Notice that, GSRNET [20] utilizes the embedded features generated from the pretrained DNABERT model. DNAGPT can significantly outperform the modified DNABERT in the vast majority of tasks. To verify the generalization of DNAGPT, we further evaluate our model on other organisms,including mouse, fruit fly and bovine. Experimental results are presented in Figure. 3 **b**, **c** and **d**, respectively. Our DNAGPT outperform the GSRNET and DeepGSR in most cases. We also compared the results from DNAGPT and

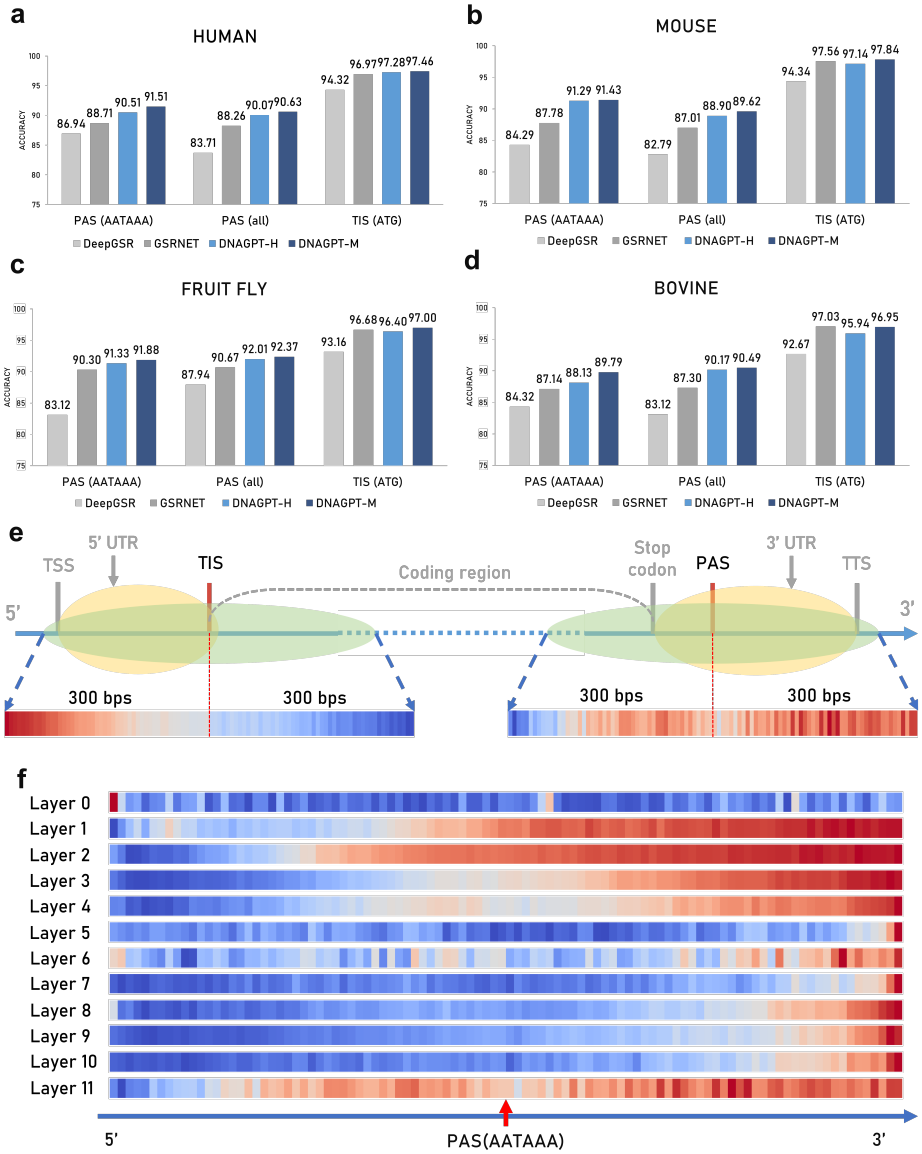

Fig. 3 Performance comparison between DNAGPTs and other methods on PAS and TIS recognition and explainability of DNAGPTs. We finetune and evaluate our DNAGPTs on the data from four organisms, including human (**a**), mouse (**b**), fruit fly (**c**) and bovine (**d**). In each subgraph, we compare the accuracy of the DNAGPT-H and DNAGPT-M with previous methods on the recognition of PAS (AATAAA), PAS (all) and TIS (ATG) from left to right. **e**. Attention maps of the final layer of DNAGPT-M with different inputs. The green region is the sampled input sequences and the GSRs are located in the middle of the sequence. The sequence length before and after the GSRs is 300 bps. **f**. Attention maps of each layer of DNAGPT-M with PAS(AATAAA) input.

DNABERT, which is included in the Figure. S2. DNAGPT-H outperforms about 3.5% and DNAGPT-M outperforms about 4.6% better than DNABER.

Compared with DNAGPT-H, which is pre-trained only on the human genome, DNAGPT-M, which includes additional DNA sequences from other species during pre-training, not only shows performance improvements on downstream tasks in species such as mouse, fruit fly, and bovine but also achieves better results in human GSR recognition tasks. This suggests that the DNA from other species provides a non-negligible enhancement to the understanding of human DNA.

Specifically, for TIS signal recognition, DNAGPT-M only makes a slight improvement compared to GSRNET. One potential explanation could be attributed to the characteristics of GPT. For PAS signal recognition, the region of interest to be identified is located after the PAS signal, while for TIS signal recognition, the fixed signal 'ATG' is the downstream of the region of interest. As the GPT is a unidirectional attention model that can only attend to information preceding the current input. The PAS signal provides a prompt to the GPT, enabling the model to better extract task-relevant information from the subsequent sequence, whereas the TIS signal located at the end of the sequence can not provide such a prompt. In conclusion, DNAGPT can handle any classification tasks as well as provide sufficient prior knowledge for DNA sequence understanding.

4.2 DNAGPT recognizes GSRs based on non-coding regions.

To explore the inner relations behind DNAGPT's ability to recognize GSRs, we visualized the attention map of final layer in DNAGPT-M's backbone. The input data were TIS and PAS (AATAAA) from humans, respectively. As shown in Figure. 3 **e**, we present the attention maps together with the real locations of GSRs. The green areas are the sampled regions, thus, both of the sampled TIS/PAS sequences contain coding and non-coding regions. TIS is located at a specific position on mRNA. It is situated in front of the coding region and is adjacent to the coding region. Therefore, we can observe that the primary reason DNAGPT-M can accurately identify TIS is that most of its attention is focused on the non-coding regions. Similarly, for PAS, since the location of PAS is in the middle of non-coding regions, DNAGPT-M has a broader attention range when recognizing PAS compared to identifying TIS.

At the same time, we also visualized the attention map of each layer in DNAGPT-M. The input sequence is PAS (AATAAA) sequence where the PAS site is located in the middle of the sequence. We can observe that almost all layers focus on the latter half of the area, with shallow and deep layers having a more widespread attention compared to middle layers. We can also notice that the attention map of the shallow areas is smoother than that of the deep areas. Although the attention range of the deep layers is as extensive as that of the shallow layers, the deep networks tend to focus on a few specific tokens rather than presenting a smooth state like the shallow attention map. This indicates that some regions in non-coding areas may be more critical for PAS recognition compared to other areas. We have also displayed the attention map for each layer with TIS data. The results can be found in the Figure. S3.

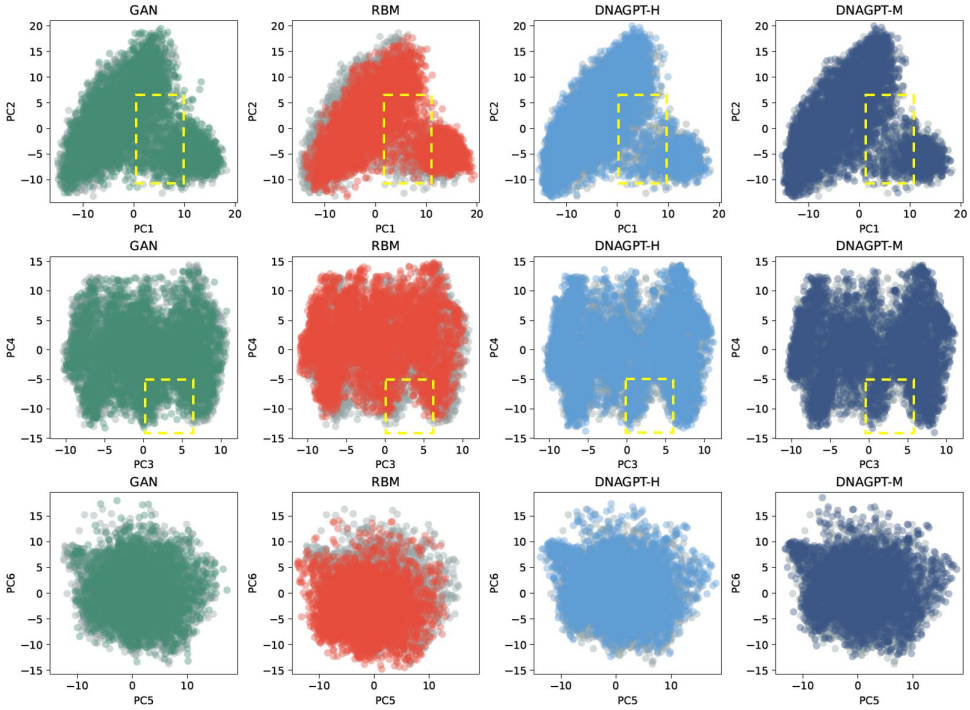

Fig. 4 Comparison of PCA results of artificial human genomes generation. **From left to right:** GAN(green) vs. real(gray); RBM(red) vs. real(gray); DNAGPT-H(light blue) vs. real(gray); DNAGPT-M(dark blue) vs. real(gray).

5 Artificial human genomes generation

As a primitive task of the GPT model, we further investigated whether DNAGPT could concentrate on essential information in DNA sequences during the generation of artificial human genomes (AGs). Artificial human genomes can protect genetic privacy and reduce the cost of genetic sample collection. Follow [8], we first finetune our DNAGPT on 5008 haplotypes from 1000 Genomes data [40] and then generate 5000 pseudo human genomes of 10000 Single Nucleotide Polymorphisms (SNPs) region for further analysis.

5.1 Analysis of artificial human genomes

We will evaluate DNAGPTs and comparison methods from the following perspectives: principal component analysis (PCA) [41]; allele frequency (AF) [42] and linkage disequilibrium (LD) [43]. The evaluation metric will include 2D Wasserstein distances [44] and correlation (r^2).

Principal component analysis

We reproduce and present the PCA analysis of the pseudo sequences generated from GAN, RBM together with our DNAGPTs in Figure. 4. Results show that all of the mentioned methods can capture the distribution of the real human genomes. In relative terms, PCA results of the sequences generated by RBM have larger deviations compared to those generated by other models. The pseudo sequences generated by GAN and our two DNAGPT models are very similar to the distribution of real sequences. However, as we have marked in the yellow dashed box, compared to GAN and RBM, both of the DNAGPT-H and DNAGPT-M models DNAGPT can produce fewer outliers, indicating that the DNAGPTs fits the original data distribution (represented by the gray points) more accurately. The Wasserstein distances of real genome distribution with GAN, RBM, DNAGPT-H and DNAGPT-M are 1.753. 3.432, 1.647 and 1.131, respectively.

Allele frequency analysis

Allele frequency analysis is a genetic analysis method used to determine the frequency of different alleles of a gene locus. In this analysis, we detect the frequency of SNPs within the 5,000 generated human genome sequences, each with a length of 10,000. In addition, the allele frequency at a polymorphic site depends on the variation of that site in all 5,000 cases. We conduct the analysis to the sequences generated by all the models. As shown in Figure. 5 **a**, only RBM has demonstrated a much diverse frequency distribution with a correlation of 0.94 compared to the real frequency. the DNAGPT-H, DNAGPT-M and GAN perform stably with a correlation of 0.99. We then visualize the correlation of those sites with allele frequency less than 0.2. As shown in Figure. 5 **b**, DNAGPT-H and DNAGPT-M outperform GAN (0.94) and RBM (0.83) with correlations of 0.95 and 0.96, respectively, indicating that DNAGPT-H and DNAGPT-M are better at capturing information from low-frequency alleles.

Linkage disequilibrium analysis

Linkage disequilibrium (LD) is a phenomenon in population genetics where the frequencies of two or more genetic markers (like alleles or genes) are associated with each other more often than would be expected by chance. We further analyzed all the generated sequences from this perspective. The first and seconde panels of Figure. 5 **c** illustrates the difference in LD values between human genomes generated by GAN and RBM compared to real genomes, respectively, where the third and fourth panels show the result from DNAGPT-H and DNAGPT-M. In the pictures, the lighter the color, the more similar the LD heat map is to real genomes. Among them, the LD of DNAGPT-H and DNAGPT-M is slightly weaker than that of real genomes, while GAN and RBM are stronger than the original genomes. Overall, the heat map performance of DNAGPT-H and DNAGPT-M is better than GAN and RBM, as their colors are lighter. We then visualized the average LD results based on the distance between SNPs. As shown in Figure. 6 **d**, it can be clearly seen that at short distances between SNPs, GAN and RBM cannot effectively capture the latent correlation. In contrast to that, DNAGPT-H and DNAGPT-M can relatively accurately fit the correlation of the real distribution, both in short and long distances ranges. Relatively speaking,

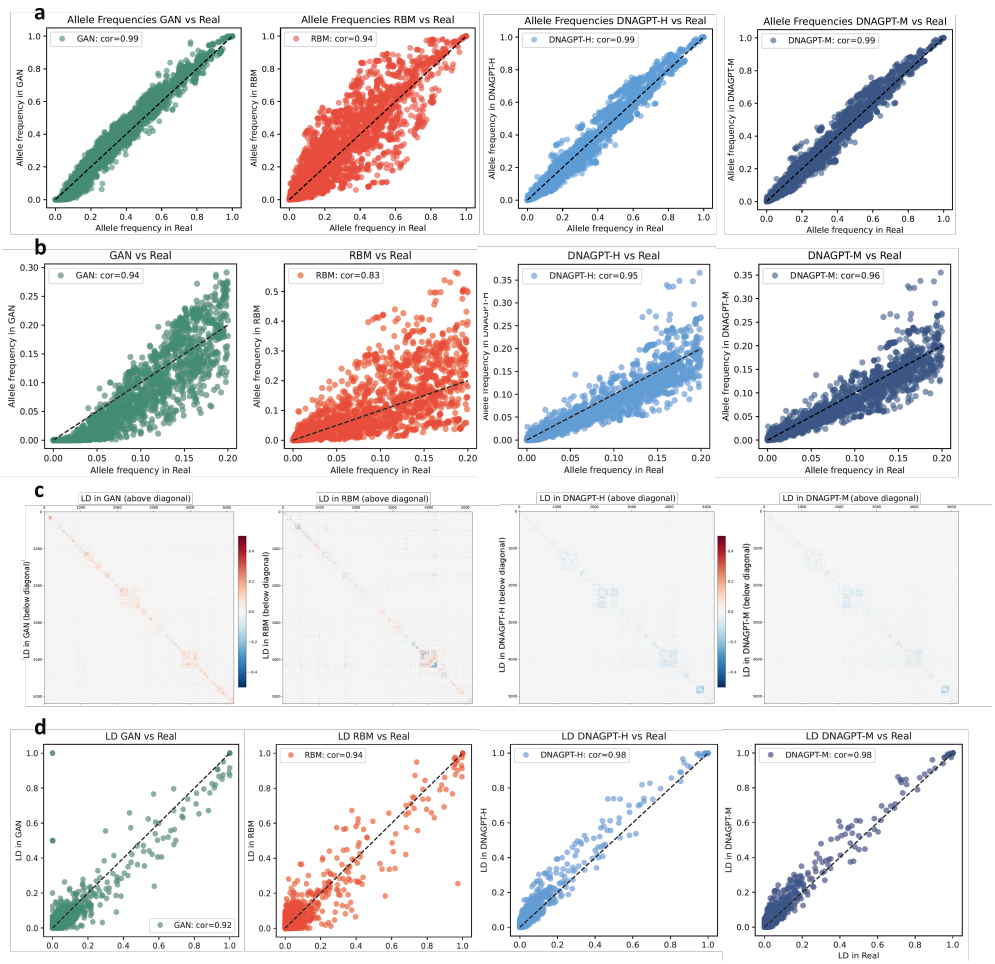

Fig. 5 We placed the results of the same method in one column, specifically, the first column is GAN vs. real; the second column is RBM vs. real; the third column is DNAGPT-H vs. real; the fourth column is DNAGPT-M vs. real. Each row represents: **a**. Correlations of allele frequency between real genomes and artificial genomes. **b**. Correlations of allele frequency between real genomes and artificial genomes, specific on the sites with allele frequency less than 0.2 in the real genomes. **c**. Normalized correlation matrices of SNPs. We subtracted the correlation matrix of each method from the real genomes. The lighter the color, the closer the artificial genomes are to the real genomes. **d**. Correlations of LD between real genomes and artificial genomes.

DNAGPT-M is slightly more similar to real genomes compared to DNAGPT-H. The above conclusions can also be verified through an comparison of correlation values. We present the correlation distributions in Figure. 5 **d**. The correlation between real and generated LD across GAN and RBM is 0.92 and 0.94 and both of the two DNAGPTs can achieve the score of 0.98.

Pairwise haplotype distances analysis

Pairwise haplotype distances refer to the genetic distances between different haplotypes within a genome. When calculating the distances, we typically compare the differences in the alleles at corresponding loci between two haplotypes. In this analysis, we first calculate the pairwise distance distributions with-in each cluster of generated genomes (GAN vs GAN, RBM vs RBM, DNAGPT-H vs DNAGPT-H, DNAGPT-M vs DNAGPT-M), defined as *Within-cluster*, then the pairwise distance distributions between real genomes and generated genomes of each methods (GAN vs Real, RBM vs Real, DNAGPT-H vs Real, DNAGPT-M vs Real) are defined as *Between-cluster*. Then we calculate the Wasserstein distances between the two types of distributions with the with-in distribution of real genomes (Real vs Real). We present the Wasserstein distances in Figure. 6 **a**, the *Within-cluster* shows the discrepancy between the pairwise distance within the distribution of each method and the actual distribution. Among them, the GAN's distribution has the largest gap compared to the actual distribution, followed by DNAGPT-H and DNAGPT-M. The genomes generated by RBM have the smallest discrepancy from real genomes. The *Between-cluster* reflects the discrepancy between the pairwise distance distribution of genomes generated by each method and real genomes. The genomes generated by DNAGPT-H and DNAGPT-M are the most similar to real genomes, while RBM performs the worst, followed closely by GAN.

5.2 Test time adjustment of DNAGPT

When a trained DNAGPT generates the DNA sequence, we can control the randomness of the output sequence by adjusting the *generation temperature*. The generation temperature range from 0 to infinity. The higher the generation temperature, the more random the generated sequence will be. In the experiments mentioned earlier, our default generation temperature was 0.8. In this section, we will adjust the generation temperature to 1.2 to validate the performance of DNAGPT under different generation temperatures. We compared DNAGPT-M with a generation temperature of 1.2 to that with a generation temperature of 0.8. The results are shown in the Figure. 6 **c** and **d**. Figure. 6 **c** shows the wasserstein distance, correlations of allele frequency, and correlations of linkage disequilibrium with real distribution. Figure. 6 **d** shows wasserstein distance of pairwise haplotype distance distribution (within-cluster and between-cluster). We can find that a larger generation temperature allows DNAGPT to maintain the correlation of allele frequency and linkage disequilibrium virtually unchanged while increasing the distance from the real distribution. It also increases the Wasserstein distance of pairwise haplotype distance distribution, indicating that a larger generation temperature makes the generated DNA sequences more diverse, and the gap from the original distribution will slightly increase. Therefore, users can adjust the generation temperature according to their needs, thereby controlling the diversity and authenticity of the generated sequences.

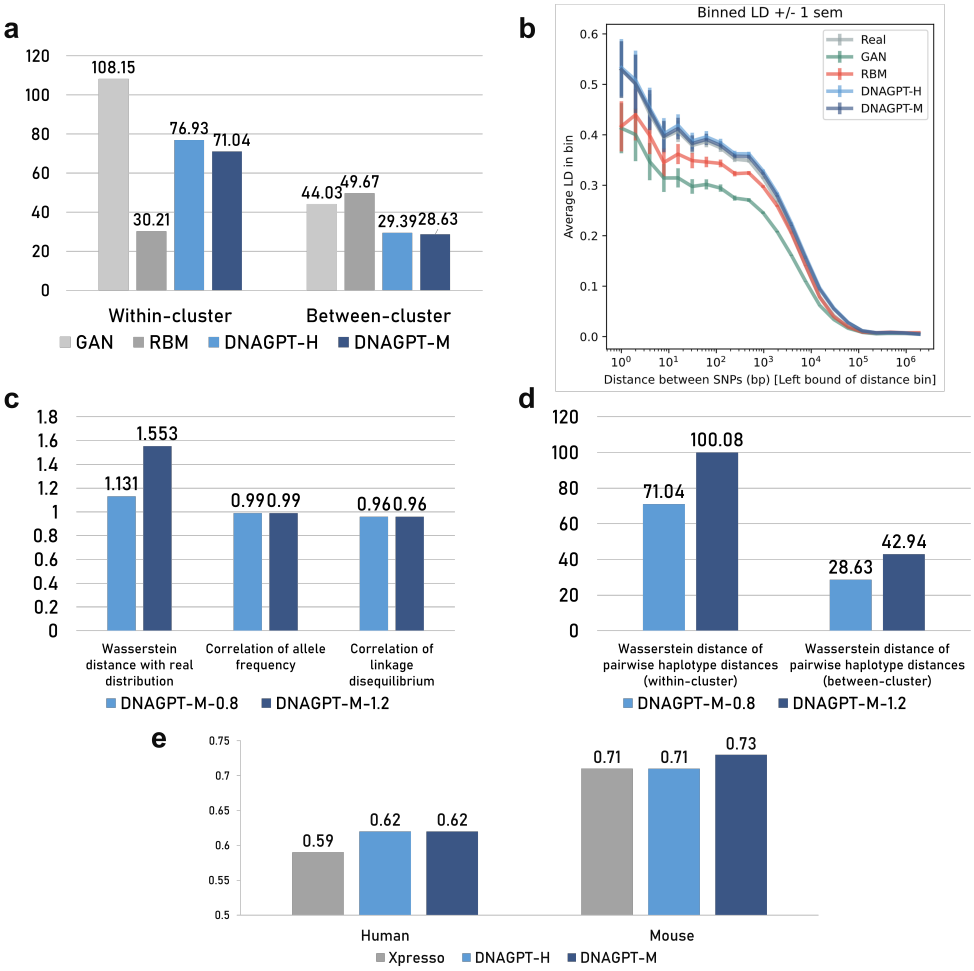

Fig. 6 **a**. Wasserstein distances of pairwise haplotype distance distribution. **b**. Average LD as a function of SNP distance after removing sites that are fixed in at least in one dataset. Pairwise SNP distances were stratified into 50 bins and for each distance bin, the correlation was averaged over all pairs of SNPs belonging to the bin. Green: GAN; Red: RBM; Light blue: DNAGPT-H; Dark blue: DNAGPT-M. **c**. Comparisons of wasserstein distance, Correlation of allele frequency, and Correlation of linkage disequilibrium for DNAGPTs with generation temperature 0.8 and 1.2 respectively. **d**. Comparisons of wasserstein distance of pairwise haplotype distance distribution for DNAGPTs with generation temperature 0.8 and 1.2 respectively. **e**. Performance comparison for DNAGPTs and other methods on mRNA abundance prediction. The reported r^2 results show that, compared to mouse genomes, tasks on human genomes can benefit more by utilizing both DNAGPTs.

6 mRNA expression levels prediction

We then investigated whether DNAGPT could extract more abundant information from DNA sequences by attempting to predict the mRNA expression levels of corresponding genes directly from genomic sequence information. Following Xpresso [21],

we utilize 18,377 and 21,856 genes in human and mouse respectively and held out 1000 genes in each specie for testing. CAP-Analysis Gene Expression (CAGE) is used to refine the annotations. We further test our model to predict the mRNA abundance from the genomic sequences together with the mRNA half-lives. As mentioned in the last line of Figure. 2 **b**. We combined the genomic sequences with the mRNA half-lives in a single sequence and add a number tokens to predict the expression level of the mRNA abundance. We present the r^2 (Coefficient of determination) metric in Figure. 6 **e**. DNAGPT-H outperforms Xpresso with an increase of over 3% for human mRNA abundance prediction and is comparable or slightly better than Xpresso with a r^2 of 0.7144 compared to 0.71. While maintaining the performance of human mRNA expression levels prediction compared to DNAGPT-H, DNAGPT-M improved the results on the mouse species from 0.71 to approximately 0.73.

The input format of this task can be regarded as a unique special form in biology sequence analysis. Previously, these types of tasks could only be solved by specialized models designed by experts. However, the use of DNAGPT can obviate the need for designing more diverse and complex models, while also providing potential guidance for downstream tasks from pre-trained information.

7 Discussion

In summary, we have developed a unidirectional attention model called DNAGPT for DNA sequence analysis to accommodate various types of downstream tasks across multiple species. We conducted the pre-training on reference genomes from as many as 9 different species. Meanwhile, we introduced joint training of numbers and sequences during the pre-training process. In order to better encode the relationships between inputs and outputs for various tasks, we designed a set of symbolic languages to assist the model in understanding the data of inputs and outputs. For the pre-training tasks, to better understand the uniqueness of DNA sequences, in addition to the next-token prediction task in GPT, we also introduced two pre-training tasks: GC content prediction and sequence order prediction. Finally, we utilized the symbolic language to compile mixed inputs and outputs of DNA sequences and numbers from 9 species, incorporating the three pre-training tasks, jointly trained a 0.1 billion parameters GPT model.

In the genomic signals and regions recognition task, when given a DNA sequence from different species as input, DNAGPT can determine with higher accuracy whether the sequence is a genuine genomic signal or region. Furthermore, after a small amount of data fine-tuning, DNAGPT can handle simultaneous inputs of DNA sequences and mRNA half-lives to predict mRNA expression levels. In the Artificial human genomes generation task, the human genomes generated by DNAGPT rank highly in a variety of evaluation metrics, indicating that DNAGPT can effectively comprehend and extract underlying relationships and information within genomes.

Biological research involves a large amount of data and numerous data relationships, along with complex and diverse data and task formats. This provides a solid foundation for applying pre-trained large-scale models in biological research.

Our DNAGPT provides a comprehensive solution for the application of pre-trained large-scale models in biology, bringing a new perspective to biological model research.

8 Methods

Pre-training of DNAGPT

For DNAGPT-M, we collected reference genome information of 9 species from the Ensembl database, including *arabidopsis_thaliana, caenorhabditis_elegans, bos_taurus, danio_rerio, drosophila_melanogaster, escherichia_coli_gca_001721525, homo_sapiens, mus_musculus, saccharomyces_cerevisiae*. Subsequently, we removed the mitochondrial genomes from the majority of the species in the preprocessing procedure. After preprocessing, the number of bps in the genome of each species is: arabidopsis_thaliana (119146348 bps), caenorhabditis_elegans (100272607 bps), bos_taurus (2628394923 bps), danio_rerio (1345101833 bps), drosophila_melanogaster (137547960 bps), escherichia_coli_gca_001721525 (5176750 bps), homo_sapiens (3088286401 bps), mus_musculus (2723414844 bps), saccharomyces_cerevisiae (12071326 bps). The total amount of bps is 10159412992. During the data sampling stage, we employed a non-overlapped k-mers sampling strategy to handle DNA sequence data. While sampling, we removed sequences with an 'N'(denoted as "not detected") content ratio greater than 0.05. Moreover, we performed random flipping with a probability of 0.5. we then encoded each input DNA sequence and numerical information according to the symbolic language and the pre-training tasks we designed.

DNAGPT-M consists of 12 layers of transformer blocks based on unidirectional attention, with each layer containing 12 attention heads and a hidden layer size of 768. The trained parameters in the model is 0.1 billion. The learning rate is set to 1e-4 with a cosine decay scheduler. The weight decay is set to 1e-2. The optimizer we choose is AdamW with the betas is set to (0.9, 0.95) and momentum is set to 0.937. We employed mixed precision for pre-training. The model was pre-trained for 15 epochs. The pre-training of the model on 8 Nvidia V100 32GB GPUs takes approximately one day.

Similar to DNAGPT-M, DNAGPT-H used the same model as well as the hyper-parameters, but the pretraining data changed from genomes of 9 species to only the human reference genome.

Non-repeating k-mers sampling

A k-mer consists of k nucleotides. When sampling with a kmer-based strategy, previous methods involved overlapped sampling, that is, regardless of the value of k, the shift during each sampling is always 1. With a fixed input token count N in the model, the effective length of the DNA sequence input using this strategy is $N-k+1$. In the non-overlapped k-mers strategy, the shift is equal to K. In this way, under the same input token length N, the input base length of the model can be extended from $(N+k-1)$ to $(k \times N)$, effectively increasing the DNA sequence input length by approximately k times compared to the current model.

Finetuning of DNAGPT

When fine-tuning DNAGPTs, we first select different model architectures based on the task format of the downstream task. That is, as shown in Figure.1 **c** and **d**, while ensuring a completely identical model backbone, we choose suitable encoding heads. For classification and generation tasks, we only need to select sequential heads and classification for input and output encoding. For regression tasks and more complex composite tasks, we need to incorporate numerical heads and regression heads for joint encoding. After determining the model architecture, we load the pre-trained weights into the model, and the weights of unused heads will be discarded. Then we can finetune DNAGPTs using the data from downstream tasks. During fine-tuning, we use the same hyperparameters across all downstream tasks. Hyperparameters used for fine-tuning were as follows: max learning rate, 5×10^{-5}; learning scheduler, cosine with warmup; optimizer, AdamW; warmup epoch, 5; weight decay, $1e-1$; batch size, 8;

In the genomic signals and regions recognition, we used the sequential head and classification head. The metrics we used are ACC (Accuracy), F1 (F1 score), MCC (Matthews Correlation Coefficient), Precision and Recall. We will report the complete results in the Table. S2. In mRNA expression levels prediction, we simultaneously invoked the sequential head and numerical head to handle the input of sequences and numbers. For the output expression level, we used the regression head to output the values. In artificial human genomes generation, we used only the sequential head and Classification Head to handle input and output sequences. During fine-tuning, we added a stop symbol at the last position of the input sequence to determine the model's output length. When generating test sequences, we removed all sequences that did not generate stop symbols and those with incorrect stop symbol positions in the post-processing step. For test time adjustment, without any further training and without changing any other hyper-parameters, we generated the test sequences solely by adjusting the generation temperature. This experiment demonstrated that by adjusting the generation temperature, we can use a fixed model to generate more diverse sequences without requiring any additional training.

References

[1] Tipper, D.J., Strominger, J.L.: Mechanism of action of penicillins: a proposal based on their structural similarity to acyl-d-alanyl-d-alanine. Proceedings of the National Academy of Sciences **54**(4), 1133–1141 (1965)

[2] Giegé, R., Jühling, F., Pütz, J., Stadler, P., Sauter, C., Florentz, C.: Structure of transfer rnas: similarity and variability. Wiley Interdisciplinary Reviews: RNA **3**(1), 37–61 (2012)

[3] Crick, F.: Central dogma of molecular biology. Nature **227**(5258), 561–563 (1970)

[4] Shendure, J., Ji, H.: Next-generation dna sequencing. Nature biotechnology **26**(10), 1135–1145 (2008)

[5] Andolfatto, P.: Adaptive evolution of non-coding dna in drosophila. Nature

437(7062), 1149–1152 (2005)

[6] Andrews, G., Fan, K., Pratt, H.E., Phalke, N., §, Z.C., Karlsson, E.K., Lindblad-Toh, K., Gazal, S., Moore, J.E., Weng, Z.: Mammalian evolution of human cis-regulatory elements and transcription factor binding sites. Science **380**(6643), 7930 (2023)

[7] Wang, R., Jiang, Y., Jin, J., Yin, C., Yu, H., Wang, F., Feng, J., Su, R., Nakai, K., Zou, Q., *et al.*: Deepbio: an automated and interpretable deep-learning platform for high-throughput biological sequence prediction, functional annotation and visualization analysis. Nucleic Acids Research **51**(7), 3017–3029 (2023)

[8] Yelmen, B., Decelle, A., Ongaro, L., Marnetto, D., Tallec, C., Montinaro, F., Furtlehner, C., Pagani, L., Jay, F.: Creating artificial human genomes using generative neural networks. PLoS genetics **17**(2), 1009303 (2021)

[9] Luo, R., Sun, L., Xia, Y., Qin, T., Zhang, S., Poon, H., Liu, T.-Y.: Biogpt: generative pre-trained transformer for biomedical text generation and mining. Briefings in Bioinformatics **23**(6), 409 (2022)

[10] Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A.C., Lo, W.-Y., et al.: Segment anything. arXiv preprint arXiv:2304.02643 (2023)

[11] Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al.: Llama: Open and efficient foundation language models. arXiv preprint arXiv:2302.13971 (2023)

[12] Bommasani, R., Hudson, D.A., Adeli, E., Altman, R., Arora, S., Arx, S., Bernstein, M.S., Bohg, J., Bosselut, A., Brunskill, E., et al.: On the opportunities and risks of foundation models. arXiv preprint arXiv:2108.07258 (2021)

[13] Consortium, E.P., *et al.*: An integrated encyclopedia of dna elements in the human genome. Nature **489**(7414), 57 (2012)

[14] Chen, K., Zhao, H., Yang, Y.: Capturing large genomic contexts for accurately predicting enhancer-promoter interactions. Briefings in Bioinformatics **23**(2), 577 (2022)

[15] Wang, Z., Zhang, Y., Liu, Y., Xiong, S., Wang, M., Zhou, J., Gong, M.: Towards a better understanding of tf-dna binding prediction from genomic features. Computers in Biology and Medicine, 105993 (2022)

[16] Lee, D., Yang, J., Kim, S.: Learning the histone codes with large genomic windows and three-dimensional chromatin interactions using transformer. Nature Communications **13**(1), 6678 (2022)

[17] Ji, Y., Zhou, Z., Liu, H., Davuluri, R.V.: Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. Bioinformatics **37**(15), 2112–2120 (2021)

[18] Kalkatawi, M., Magana-Mora, A., Jankovic, B., Bajic, V.B.: Deepgsr: an optimized deep-learning structure for the recognition of genomic signals and regions. Bioinformatics **35**(7), 1125–1132 (2019)

[19] Guo, Y., Zhou, D., Li, P., Li, C., Cao, J.: Context-aware poly (a) signal prediction model via deep spatial–temporal neural networks. IEEE Transactions on Neural Networks and Learning Systems (2022)

[20] Zhu, G., Fan, Y., Li, F., Choi, A.T.H., Tan, Z., Cheng, Y., Li, K., Wang, S., Luo, C., Liu, H., et al.: Gsrnet, an adversarial training-based deep framework with multi-scale cnn and bigru for predicting genomic signals and regions. Expert Systems with Applications, 120439 (2023)

[21] Agarwal, V., Shendure, J.: Predicting mrna abundance directly from genomic sequence using deep convolutional neural networks. Cell reports **31**(7), 107663 (2020)

[22] Cunningham, F., Allen, J.E., Allen, J., Alvarez-Jarreta, J., Amode, M.R., Armean, I.M., Austine-Orimoloye, O., Azov, A.G., Barnes, I., Bennett, R., *et al.*: Ensembl 2022. Nucleic acids research **50**(D1), 988–995 (2022)

[23] Le, N.Q.K., Ho, Q.-T., Nguyen, V.-N., Chang, J.-S.: Bert-promoter: An improved sequence-based predictor of dna promoter using bert pre-trained model and shap feature selection. Computational Biology and Chemistry **99**, 107732 (2022)

[24] Jin, J., Yu, Y., Wang, R., Zeng, X., Pang, C., Jiang, Y., Li, Z., Dai, Y., Su, R., Zou, Q., *et al.*: idna-abf: multi-scale deep biological language learning model for the interpretable prediction of dna methylations. Genome biology **23**(1), 1–23 (2022)

[25] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

[26] Bollerslev, T.: Generalized autoregressive conditional heteroskedasticity. Journal of econometrics **31**(3), 307–327 (1986)

[27] Li, Z., Chen, Z., Yang, F., Li, W., Zhu, Y., Zhao, C., Deng, R., Wu, L., Zhao, R., Tang, M., *et al.*: Mst: Masked self-supervised transformer for visual representation. Advances in Neural Information Processing Systems **34**, 13165–13176 (2021)

[28] Melsted, P., Pritchard, J.K.: Efficient counting of k-mers in dna sequences using

a bloom filter. BMC bioinformatics **12**(1), 1–7 (2011)

[29] Gillioz, A., Casas, J., Mugellini, E., Abou Khaled, O.: Overview of the transformer-based models for nlp tasks. In: 2020 15th Conference on Computer Science and Information Systems (FedCSIS), pp. 179–183 (2020). IEEE

[30] Geoffroy, V., Herenger, Y., Kress, A., Stoetzel, C., Piton, A., Dollfus, H., Muller, J.: Annotsv: an integrated tool for structural variations annotation. Bioinformatics **34**(20), 3572–3574 (2018)

[31] Meyers, B.C., Tingey, S.V., Morgante, M.: Abundance, distribution, and transcriptional activity of repetitive elements in the maize genome. Genome Research **11**(10), 1660–1676 (2001)

[32] Dillon, L.W., Kumar, P., Shibata, Y., Wang, Y.-H., Willcox, S., Griffith, J.D., Pommier, Y., Takeda, S., Dutta, A.: Production of extrachromosomal microdnas is linked to mismatch repair pathways and transcriptional activity. Cell reports **11**(11), 1749–1759 (2015)

[33] Basehoar, A.D., Zanton, S.J., Pugh, B.F.: Identification and distinct regulation of yeast tata box-containing genes. Cell **116**(5), 699–709 (2004)

[34] Remmele, C.W., Xian, Y., Albrecht, M., Faulstich, M., Fraunholz, M., Heinrichs, E., Dittrich, M.T., Müller, T., Reinhardt, R., Rudel, T.: Transcriptional landscape and essential genes of neisseria gonorrhoeae. Nucleic acids research **42**(16), 10579–10595 (2014)

[35] Korhonen, J.A., Gaspari, M., Falkenberg, M.: Twinkle has 5-¿ 3 dna helicase activity and is specifically stimulated by mitochondrial single-stranded dna-binding protein. Journal of Biological Chemistry **278**(49), 48627–48632 (2003)

[36] Juo, Z.S., Chiu, T.K., Leiberman, P.M., Baikalov, I., Berk, A.J., Dickerson, R.E.: How proteins recognize the tata box. Journal of molecular biology **261**(2), 239–254 (1996)

[37] McLauchlan, J., Gaffney, D., Whitton, J.L., Clements, J.B.: The consensus sequence ygtgttyy located downstream from the aataaa signal is required for efficient formation of mrna 3 termini. Nucleic acids research **13**(4), 1347–1368 (1985)

[38] Devlin, J., Chang, M.-W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)

[39] Choromanski, K., Likhosherstov, V., Dohan, D., Song, X., Gane, A., Sarlos, T., Hawkins, P., Davis, J., Mohiuddin, A., Kaiser, L., et al.: Rethinking attention

with performers. arXiv preprint arXiv:2009.14794 (2020)

[40] Consortium, .G.P., *et al.*: A global reference for human genetic variation. Nature **526**(7571), 68 (2015)

[41] Abdi, H., Williams, L.J.: Principal component analysis. Wiley interdisciplinary reviews: computational statistics **2**(4), 433–459 (2010)

[42] Boehnke, M.: Allele frequency estimation from data on relatives. American journal of human genetics **48**(1), 22 (1991)

[43] Reich, D.E., Cargill, M., Bolk, S., Ireland, J., Sabeti, P.C., Richter, D.J., Lavery, T., Kouyoumjian, R., Farhadian, S.F., Ward, R., *et al.*: Linkage disequilibrium in the human genome. Nature **411**(6834), 199–204 (2001)

[44] Arjovsky, M., Chintala, S., Bottou, L.: Wasserstein generative adversarial networks. In: International Conference on Machine Learning, pp. 214–223 (2017). PMLR

[45] Lizio, M., Harshbarger, J., Shimoji, H., Severin, J., Kasukawa, T., Sahin, S., Abugessaisa, I., Fukuda, S., Hori, F., Ishikawa-Kato, S., *et al.*: Gateways to the fantom5 promoter level mammalian expression atlas. Genome biology **16**, 1–14 (2015)

S1 Supplementary

S1.1 All tokens used in DNAGPT

Token names	Tokens	Description
Discrete tokens	Examples: <ATCTAG>	DNA sequence tokens
Continuous tokens	Examples: 0.85, -0.02	Number tokens
Classification tokens	True: <Y>; False: <X>;	Used to indicate the result of a binary classification problem
Instruction tokens	Arabidopsis_thaliana: <I>; Caenorhabditis_elegans: <K>; Bos_taurus: <V>; Danio_rerio: <L>; Drosophila_melanogaster: <O>; Escherichia_coli_gca_001721525: <Q>; Homo_sapiens: <R>; Mus_musculus: <S>; Saccharomyces_cerevisiae: <U>; Classification tokens: <B>: Continuous tokens: < M>;	Used to indicate the type of data generated afterwards
Connection tokens	Information fusion token: <+>; Output token: <=>;	Used to indicate the relationship between the two sequences before and after
Reserved tokens	<0>; <1>; <2>; <3>; <4>; <5>; <6>; <7>; <8>; <9>; <->; <*>; </>; <&>; <|>; <!>; <P>; <W>; <Z>	Used to build up downstream tasks (Example: <P> can be used for padding)

Fig. S1 All the tokens we used in our DNAGPT. We present the token name, the tokens (For cases with a large number of tokens, we only provided some examples.) and the token description.

There are 6 categories of tokens we used in building up the whole DNAGPT structure. The *Discrete tokens* are the DNA sequences encoded with kmers sampling strategy. For example, if we utilize 6-mers sampling and only consider the encoding of 'A, C, G, T, N', then the total amount of discrete tokens are 5^6 which is 15625. When is comes to the *Continous tokens*, as the numbers are uncountable, we directly input the numbers into the *Numerical head* and *Regression head* to encode and decode them as the continuous tokens. For common binary classification tasks, we We established two special tokens to distinguish True from False. The *Instruction tokens* are used to identify the input and output type of sequence. For DNA sequence from different species, we assigned an instruction token for each species. Specifically, we also assign

instruction tokens for Classification tokens and Continuous tokens which can prompt the model to generate corresponding special tokens separately. In biological sequences, there is no natural logical relationship between tokens like in natural language. In the design of DNAGPT tokens, to enable the model to understand the relationships between sequences, we designed two connection tokens to guide the relationships between sequences before and after the connection tokens. Here, '+' represents the fusion of preceding and succeeding information, and '=' represents the cause-effect relationship, with the input being before '=' and the output being after '='. Finally, in order to better adapt to different types of downstream tasks, we also preset some reserved tokens for use when fine-tuning downstream tasks.

S1.2 Datasets

S1.2.1 Genomic signals and regions recognition

The datasets used for genomic signals and regions recognition are cDNA data. We extracted both polyadenylation signals (PAS) and translation initiation sites (TIS) from four genomes. For the Homo sapiens (human) genome, the human assembly GRCh37 (also known as hg19) was employed, while the primary assembly GRCm38 was used for the Mus musculus (mouse) genome. The cDNA data for these genomes were sourced from the Mammalian Gene Collection (MGC). For the Bos taurus (bovine) genome, the assembly Bos_taurus_UMD_3.1.1 was utilized, with the cDNA data being downloaded from the Ensembl organization. Finally, for the Drosophila melanogaster (fruit fly) genome, Release_6 – annotation release Dmel_Release_6.01 was employed, and the cDNA data was obtained from FlyBase. The sampling method is as follows: first, locate the positions of GSRs, then extract 300 bps of sequence from both before and after the GSRs, and concatenate them together. It is important to note that the GSR motif will be removed during preprocessing to ensure that the model can recognize GSRs based solely on the information near the GSR motif, rather than the GSR itself. For the negative samples, the sampled sequences should satisfied the following requirements:

(1) Sequences with the same motifs but not related to polyadenylation and translation processes.

(2) Sequences are sampled from the chromosome whose average GC-content was nearest to the entire genome's average GC-content.

Consequently, negative data for human, mouse, bovine, and fruit fly were extracted from chromosomes 21, 13, 28, and X, respectively.

The amounts of positive samples for each datasets are shown in Table. S1.

Table S1 Amounts of positive samples for different datasets.

GSRs	Human	Mouse	Bovine	Fruit fly
TIS	28244	25205	17558	30283
PAS(AATAAA)	11,302	11,393	7,862	18,641
ALL	20,933	18,693	12,082	27,203

S1.2.2 Artificial human genomes generation

For artificial human genomes generation, we utilized 1000 Genomes data [40] as the fine-tuning dataset. There are 2504 individuals (5008 haplotypes) in the dataset and the data we used is a dense 10000 SNP range/region from chromosome 15. When evaluating, the model produced 5000 sequences of SNPs for analysis. All our analyses were conducted on the generated data.

S1.2.3 mRNA expression levels prediction

The dataset is composed of human protein-coding gene sequences located upstream and downstream of the transcription start site (TSS). The promoter of the gene is found in the sequence upstream of the TSS, while the exons and introns of the gene are found downstream. The input sequences are sourced from the Xpresso[21]. In this dataset, the TSS positions were meticulously revised by the authors of Xpresso using Cap Analysis Gene Expression (CAGE) [45], a technique for determining the actual TSS location. The Xpresso dataset consists of 18,377 genes, divided into 16,377 for training, 1,000 for validation, and 1,000 for testing. The maximum length of a gene's TSS sequence is set to 20,000 base pairs. We set the sample range according to the default parameters in xpresso, which is from 3000 to 13500. Additionally, the Xpresso DNA input includes half-life features that provide general information about the gene, such as gene length and the number of introns. The default feature input is an 8-bit array.

S1.3 Other results of genomic signals and regions recognition

Full results of genomic signals and regions recognition

We show in the Table. S2 the results of DNAGPT-M on various datasets, including different metrics. DNAGPT-M demonstrates stable results across different GSR recognition datasets from various species.

Table S2 Full results of genomic signals and regions recognition.

Task name	acc(%)	f1(%)	mcc(%)	precision(%)	recall(%)
Human_PAS(AATAAA)	91.51	91.49	82.99	91.52	91.47
Human_PAS(all)	90.63	90.64	81.28	90.64	90.64
Human_TIS(ATG)	97.46	97.46	94.92	97.47	97.46
Mouse_PAS(AATAAA)	91.43	91.41	82.83	91.40	91.43
Mouse_PAS(all)	89.62	89.62	79.24	89.63	89.61
Mouse_TIS(ATG)	97.84	97.84	95.68	97.85	97.83
Fruitfly_PAS(AATAAA)	91.88	91.87	83.84	91.96	91.88
Fruitfly_PAS(all)	92.37	92.38	84.76	92.38	92.38
Fruitfly_TIS(ATG)	97.00	97.00	94.01	97.00	97.00
Bovine_PAS(AATAAA)	89.79	89.77	79.65	89.89	89.76
Bovine_PAS(all)	90.49	90.49	80.99	90.49	90.49
Bovine_TIS(ATG)	96.95	96.95	93.90	96.95	96.95

Comparisons between DNAGPT and DNABERT

We evaluate DNAGPTs and DNABERT on the human PAS dataset. The result demonstrate that DNAGPT can better capture the relative information of GSR in the sequence.

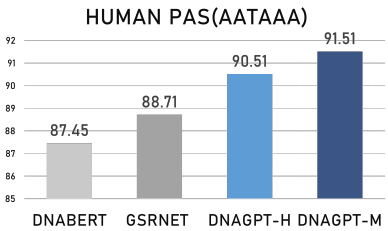

Fig. S2 We compared DNAGPTs and DNABERT on human PAS(AATAAA) datsets. Both of the DNAGPTs outperform DNABERT.

Attention maps of DNAGPT-M with TIS input

In the Figure. S3, we display the attention maps of each layer of DNAGPT-M with TIS input. Interestingly, compared to the attention map with PAS as input, the information focused on by the model in the shallow layers is more consistent, with a notable difference only in Layer 1. In the later layers, the attention map for TIS input starts to focus on information from tokens in earlier positions, i.e., non-coding region information. This suggests that the information the model focuses on in the shallow layers is more approximate, but in the deep networks, it can more precisely pinpoint the locations of important tokens.

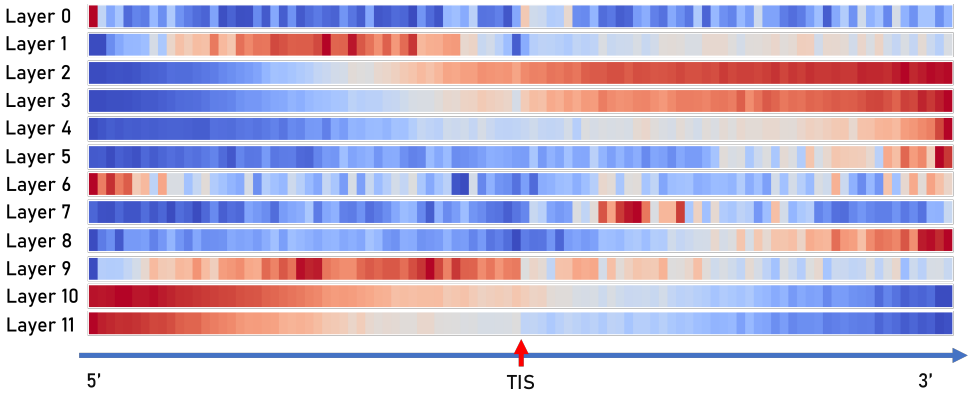

Fig. S3 Attention maps of each layer of DNAGPT-M with TIS input.