NATIONAL RESEARCH UNIVERSITY
HIGHER SCHOOL OF ECONOMICS

Faculty of Computer Science
Bachelor's Programme "Applied Mathematics and Informatics"

UDC 004.8

Research Team Project Report on the Topic:

Identifying Biological Relations in Human Genome Through Interpretation of Language Models

Submitted by the Students:

group #БПМИ223, 3rd year of study — Libina Yana Maksimovna
group #БПМИ223, 3rd year of study — Skvortsova Irina Vladislavovna

Approved by the Project Supervisor:

Borevskiy Andrey Olegovich
Research Fellow
Faculty of Computer Science, HSE University

Moscow 2025

Contents

Annotation

This work considers different large language models in terms of Z-DNA prediction task. We start with two models pre-trained on nucleotide data, DNABERT-2 and Gena-LM, fine-tune them on Kouzine et al dataset and interpret the results using xAI methods. Both of the fine-tuned models outperform previous deep learning approaches (DeepZ, Z-DNABERT) and confirm the significance of known sequences associated with Z-DNA formation like $\mathbf{poly(dGC)_2}$ and alternating purine-pyrimidine sequences in general.

Аннотация

В данной работе рассматриваются различные языковые модели в рамках задачи предсказания Z-ДНК структуры. Мы взяли две модели, предобученные на нуклеотидных последовательностях, DNABERT-2 и Gena-LM, дообучили на датасете Kouzine et al и проинтерпретировали результаты, используя методы xAI. Обе дообученные модели получили лучшее качество, чем предыдущие нейросетевые подходы (DeepZ, Z-DNABERT) и подтвердили значимость известных последовательностей, связанных с образованием ДНК, таких как $\mathbf{poly(dGC)_2}$ и в общем последовательностей с чередующимися пуринами-пиримидинами.

Keywords

Deep learning, bioinformatics, Large language models (LLM), Z-DNA, core promoters, Explainable AI (xAI), Parameter efficient fine-tuning (PEFT)

1 Introduction

Z-DNA is one of the DNA secondary structures. As well as regular right-handed B-DNA, it is a double-stranded helix, however it's main difference is that it is left-handed. The Z-DNA structure is of great interest in today's biological fields. Although it was discovered in 1979 [1], its functions and its full potential are yet to be fully unraveled. However, we already know that it has an impact on vital biological processes like transcription, recombination, gene regulation and evolution, and its formation recently has been widely associated with cell immune response [3, 12].

In addition to this, Z-DNA is found associated with Alzheimer's disease (AD) [10]. There is a hypothesis that Z-DNA is formed in promoter regions of some genes which regulation is crucial for AD. And Z-DNA formation has an impact on these genes' expression that causes the illness. Moreover, Z-DNA was found in hippocampal area of AD patiens, which makes the problem of its prediction crucial in terms of decease analysis.

Nevertheless, Z-DNA is not necessary a negative thing. There is a research where a new method of fighting severe cancer is introduced [16]. An experimental study was held and its conclusions are promising for fighting tumors, even those that are resistant to current therapies.

All the reasons listed above prove the need in finding performative tools to predict Z-flipons.

The task of predicting Z-DNA presence with language models is one of the developing areas of deep learning in bioinformatics. However, all current SOTA models are not pre-trained on this task. It is quite challenging to conduct experiments or make a genome map with Z-flipons because of the dynamic nature of Z-DNA. It may form for a few seconds, do its purpose and disassemble back to regular B-DNA. Because of this, there is a problem with high-quality datasets on the task, and existing models have a relatively small prediction score.

In this work we select two large language models architectures, DNABERT-2 and Gena-LM, evaluate them on a well-studied core promoter detection task and fine-tune them for Z-DNA prediction on Kouzine et al dataset [6]. As a result, both of the models outperform the current best deep learning approach Z-DNABERT (F1 = 0.83), yielding F1 = 0.8908 with DNABERT-2 and F1 = 0.8626 with GenaLM. Then, we interpret our models' predictions using Explainable AI methods.

In section 3 we give an introduction to models that we use, Integrated Gradients method and parameter efficient fine-tuning with LoRA.

In section 4 we provide fine-tuning and interpretation results on core promoter detection task.

In section 5 we describe chosen methods for Z-DNA prediction task, present experiments' results and analyze the predictions with two interpretation methods.

For each section of the work model analysis, experiments, evaluation and interpretation were done by Yana Libina for DNABERT-2 and by Irina Skvortsova for Gena-LM.

Source code for the project is available at [14].

2 Literature Overview

2.1 DeepZ

One of the articles we paid our attention to was a publication by Maria Poptsova, Nazar Beknazarov and Seungmin Jin called "Deep learning approach for predicting functional Z-DNA regions using omics data" [2]. This work reveales a new computational method of forcasting Z-DNA regions called DeepZ – a deep learning model trained on a specific Z-DNA recognition dataset.

Initially the authors trained a set of RNN, CNN and hybrid CNN-RNN models for Z-DNA recognition and evaluated the results using area under the receiver operating characteristic curve (ROC AUC) and F1 score. The best results based on these metrics were achieved on an RNN model with 86.6% ROC AUC and 40.1% F1 scores. This model was chosen for the reference one and called DeepZ.

The authors developed a method of whole-genome annotation of Z-DNA area, assigning probability of belonging to Z-DNA area to each nucleotide.

The authors pointed that their attempts of Z-DNA prediction based exclusively on sequence-intrinsic information did not reach success. Hence, they made DeepZ capture not only sequence composition but also B-Z transition energy, epigenomics, transcriptomics and chromatin organization levels.

Though DeepZ outperformed some of the models that existed at the time of its creation, the results are still unsatisfactory in terms of the number of predicted Z-DNA regions.

2.2 GraphZ

Since graph architectures allow to retrieve valuable features from the neighbouring nodes, these networks have shown better performance on some genomic prediction tasks. That is why Maria Poptsova, Alan Herbert and Artem Voytetskiy [11] decided to test Graph Neural Networks for Z-DNA regions prediction. Three architectures were tested: Graph Convolutional Network (GCN), Graph Attention Network (GAT) and GraphSAGE.

For the purpose of training, DNA sequence was presented as a graph where each node was a nucleotide. Each node got certain features.

Though all the graph networks under study outperformed RNN networks and increased recall in 2.3-2.6 times for genome prediction, most architectures turned out to be more suitable for small datasets. In general, this method helped to improve results of Z-DNA prediction in

comparison with DeepZ on small datasets, but not on Kouzine et al dataset.

2.3 Z-DNABERT

The latest work on the topic by Dmitry Umerenkov et al presented a large language model for Z-DNA prediction called Z-DNABERT [9]. This model comes from the 6-mer DNABERT model [5], fine-tuned on the Kouzine et al dataset [6], containing experimental data with Z-DNA regions. In this dataset a label of 1 or 0 is assigned to every nucleotide, depending on the nucleotide's occurrence in Z-DNA. Unlike previous approaches, this model does not take into account any omics features, and is trained solely on nucleotide sequences. Z-DNABERT, by analogy with DNABERT, uses overlapping 6-mers tokenization. However, since the final result would be a prediction for each nucleotide, the model is trained with a token classification head. Each nucleotide label in the dataset is assigned to a k-mer that starts in this nucleotide, and for evaluation each prediction for a k-mer further becomes the prediction for the k-mer's first nucleotide.

3 Getting acquainted with current approaches

In this day and age many models for genome annotation have been created. Though not all of them show appropriate results, there is a suit of architectures that are SOTA for certain tasks. These tasks include estimating the promoter activity of a DNA sequence, predicting chromatin profiles, forecasting splicing sites, and many more.

We chose two existing models DNABERT2 and Gena-LM for further analysis. Our aim was to find a suitable model for Z-DNA regions prediction and fine-tune it for this task. Then, we interpret the results using xAI methods.

3.1 Gena-LM

This model was analyzed by Irina Skvortsova.

Gena-LM is a transformer based model. It was created in 2023 and it represents a fine-tuned DNABERT [5] model with some changes in the attention layer. The problem of DNABERT, which is common for the majority of existing models applicable for biological tasks, is the length of a sequence that can be processed by the model at once. For DNABERT the length of the sequence makes 512 bp.

One of the key purposes of Gena-LM was to increase the length of input sequences. And with help of BPE tokenization and sparse attention mechanism the desired goal was reached. The length of the input sequence was improved up to 4096 bp.

There are several versions of Gena-LM model. In the following work gena-lm-bert-base-t2t was used.

3.1.1 Byte-Pair Encoding tokenization

For efficient tokenization authors used a greedy BPE algorithm. It allowed to detect the most commonly-found subsequences in genome and use them as individual tokens. It surprisingly occurred that retrieved tokens have a significant biological interpretation. They represent LINE or simple repeats – well-known repetitive elements in genome.

3.1.2 Sparse attention

The second modification that was implemented in GENA-LM is sparse attention mechanism that was used during fine-tuning (the base model was left with the classic attention mechanism).

The essence of sparse attention mechanism is in reducing the number of words that we look at to find some dependencies between tokens and increasing the length of the input sequences.

These two improvements allowed to extend the length of input models that had a positive effect on the quality of genome annotations. Though GENA-LM outperformed DNABERT, the authors outline that for the better prediction quality it is still required to enlarge the length of input sequences by some other methods of tokenization and embeddings creation.

3.2 DNABERT-2

This model was analyzed by Yana Libina.

DNABERT-2 was proposed in 2023 in [17]. This model is an improvement of DNABERT [5], and it adopts BERT's Transformer Encoder architecture. The authors make several suggestions on overcoming DNABERT's input length constraints, reducing memory and time expenditure and enhancing model's capability. Moreover, unlike previous DNA models, they train it not only on human and mouse genomes, but also on virus, fungi and yeast (135 species in total).

3.2.1 BPE

DNABERT-2 uses BPE for tokenization, instead of k-mer tokenization, with vocabulary size of $2^{16} = 4096$, which was selected experimentally.

3.2.2 Attention with linear biases

As explained in the paper, ordinary positional encodings "suffer from input length restriction or poor extrapolation capability when applied to sequences longer than training data" [17, p. 5]. Instead, *Attention with Linear Biases* (ALiBi) [7] is used. If K is the Key matrix, L is the input sequence length and q_i is the $i-th$ query, then the attention score of q_i is calculated as $A_{(i)} = softmax(q_i K + m * [-(i-1), \cdots, -2, -1, 0, -1, -2, \cdots, -(L-1-i)]$, where m is a constant, different for each attention head. Basically, ALiBi penalizes the distance between key-query pairs with penalty rate of m.

3.2.3 Flash Attention

Flash Attention is a technique used for making attention calculations more time- and memory-efficient. The idea is in splitting the Q, K and V matrices into blocks, so that they fit into fast SRAM, and all arithmetic operations are done with respect to those blocks. Moreover, they suggest recomputing large intermediate results during the backward pass.

However, the implementation of Flash Attention from the Triton library, which is used in the official DNABERT-2 repository, is incompatible with other libraries that we use. Because of that, we decided to set it aside and not use it for training, so a regular implementation of self-attention is used.

3.2.4 GUE

The authors point out the lack of a uniformed benchmark for DNA language models. The datasets that these LLMs are trained on are unbalanced - either too trivial. or too complex. Thus, they introduce the Genome Understanding Evaluation dataset. It consists of 36 datasets for different species and different genomic tasks. For humans there are 12 datasets in GUE of lengths 70-400bp on tasks like Core-Promoter Detection, Transcription Factor Prediction, Promoter Detection and Splice Site Detection, and 6 datasets in GUE$^+$ of length 5000bp on Enhancer Promoter Interaction task.

We use the GUE dataset for evaluating our models on the core-promoter detection task.

3.3 Integrated Gradients method

Imagine, two inputs x_1 and x_2 differ only at one single dimension i and this difference leads to different class predictions of the model (i.e. $M(x_1) = 1$ and $M(x_2) = 0$). If this happens, we say that feature f_i has a non-zero attribution, which means that its value is crucial for the outcome/prediction of the model. This is called the sensitivity axiom.

Integrated Gradients is a widely-used model interpretability technique that was introduced in an article "Axiomatic attribution for deep networks" by Sundararajan, Mukund and Taly, Ankur and Yan, Qiqi in 2017[8]. This method calculates and returns attributions for each input feature of the model. It is suitable for any differentiable model and is frequently used for classification tasks.

The idea of this method is that in order to calculate importance of some feature of an input x, we choose some baseline x' and interpolate some number of inputs m between x and x'. Then we compute gradients for each interpolated input and average them with respect for each of those features. In this way, we get an integral approximation of the gradients. The following formula represents the main idea of the IG method (here M is our model).

$$InputXGradient_i(M, x, x') = (x_i - x'_i) \cdot \sum_{k=1}^{m} \frac{\partial M(x' + \frac{k}{m}(x - x'))}{\partial x_i} \cdot \frac{1}{m}$$

The value of this gradient can give a hint on the importance of each feature. For example, in terms of text classification IG will return scores for each word/token.

This technique has a realization in captum library, however in further analysis we used a module called transformers_interpret, which is mostly base on captum library. The reason for this is that out-of-the-box realization from captum does not work with attention layers, which are used in transformer architectures.

3.4 Parameter-Efficient-Fine-Tuning with LoRA

Parameter-Efficient-Fine-Tuning is an effective fine-tuning approach that helps to find balance between the speed of training, used computational resources and the performance of the model as well as the minimize risk of overfitting. The general idea of the method is to freeze some share of a pretrained model's parameters and fine-tune only a smaller subset of them.

One of the methods that help to achieve the described idea is LoRA (Low-Rank Adaptation). This approach was introduced in the article "LoRA: Low-rank adaptation of large language models"[4] published in 2021. LoRA allows to decompose the weight matrix of a model into its low-rank approximation: two matrices of lower intrinsic-rank $r << m, n$.

The initial weight matrix gets frozen, while those two approximation matrices are being updated during fine-tuning. Thus, it is possible to eliminate surplus weights that are insignificant for a certain task and significantly reduce the number of parameters.

4 Preliminary stage. Core promoters.

4.1 Problem statement

Firstly, we decided to evaluate the pre-trained models (GenaLM, DNABERT-2) on their original tasks on a unified dataset. We chose GUE dataset for core promoter sequences classification. Doing that, we wanted to understand if these models' architecture and results they provide are suitable for Z-DNA prediction task.

The dataset we used consisted of DNA-sequences, each sequence was either a core promoter area or not. Based on this each sequence was labeled by 1 or 0. This way, we used a sequence classification head on our models.

As proposed in [17], for evaluating models on the core promoter task we used the MCC metric, as well as F1 score.

4.2 Standard fine-tuning

We started with standard fine-tuning for our models for the described task. We downloaded open source files for pretrained Gena-Lm and ZDNABERT-2 models and tokenizers from Huggingface [13]. No additional limits on length of tokens were set.

Models were fine-tuned with hyperparameters show in Table 4.1.

For Gena-LM, the values of the parameters were taken from Gena-LM GitHub repository, where the authors also showed examples of classification tasks. For DNABERT-2, the hyperparameters were retrieved from the original paper.

Table 4.1: Fine-tuning hyperparameters. The values were taken from original papers and example notebooks from models' developers.

	lr	weight_decay	optimizer	warmup_ratio	batch_size	epochs
Gena-LM	2e-5	0.0	AdamW	0.1	32	10
DNABERT-2	5e-5	0.01	AdamW	0.1	32	10

Table 4.2 shows the best metrics achieved with standard fine-tuning of our models. For Gena-LM the highest MCC was 0.63 and the highest F1 score was 0.817. For DNABERT-2, best metrics were MCC=0.67 and F1=0.8328.

Table 4.2: Standard-fine tuning results on core promoter detection task.

	MCC	F1
Gena-LM	0.635	0.817
DNABERT-2	0.67	0.8328

We were satisfied with the results, as they were comparable with the benchmark on GUE presented in [17], and moved onto interpretation.

4.3 Interpretation

4.3.1 Initial method

For core promoter detection task, we used the Integrated Gradients method. We applied its realization from **transformers_interpret** python library. It extends the functional of **Captum** library, enabling it to work with models with attention layers, which is exactly our case. From **transformers_interpret** we used the **SequenceClassificationExplainer** class.

SequenceClassificationExplainer class recieves a sequence and for each token in this tokenized sequence returnes a score. The value of this score represents its impact in the final label prediction of the sequence. The score of SequenceClassificationExplainer can be either positive or negative. Positive scores indicate a token contributes positively towards the predicted class, while negative values mean a token contributes negatively towards the predicted class.

During the analysis we looked only at True Positive sequences that were truly predicted to be promoters. It meant that the higher the positive score of a token is, the more this token influenced on the decision of labeling the sequence as a promoter. The lower the negative score of a token is, the higher it impact was on trying to classify this sequence as non-promoter like.

After applying SequenceClassificationExplainer to all the sequences from test dataset, mean scores for all met tokens were counted. Each table was sorted in descending order by tokens' scores.

The results of interpretation are shown in Table 4.3 for 6 most significant tokens. Full results are available in our GitHub repository.

Table 4.3: Initial promoter interpretation results with token scores and occurrence count for IG method. Scores were rounded to 4 decimal places.

	Gena-LM			DNABERT-2		
№	Token	Score	Count	Token	Score	Count
1	AGAGTGAGAC	0.8405	1	TAATAAA	0.8676	1
2	TAACTTGC	0.7984	1	CTACAGA	0.7486	2
3	AGTCAGTC	0.7931	1	TGTGGCA	0.7255	1
4	ACTTAA	0.7895	1	GCACCTT	0.6955	1
5	TGATTGG	0.7886	1	GTCAAAA	0.6930	1
6	TTTTTTTTTTTT	0.7645	1	GAGGCCA	0.6819	2

In those files tokens are sorted by their mean scores in the descending order. A disturbing fact that can be noticed is that among first 45 tokens in the lists only a few tokens were counted more than once.

This means that the tokens in the list are so long that they repeat quite rarely in the sequences. The idea is to make tokens shorter to increase their occurrence in sequences and make the results of analysis more representative.

4.3.2 Experimenting with maximum token length

We tried setting two variants of maximum token sizes: 6 and 5. The shorted maximum length of tokens is, the longer the training is.

To reach the restriction in length we took the original tokenizer file for the model and removed all the tokens that were longer than needed. After this we also corrected the "merges", located in the tokenizer file: pairs of tokens used in BPE tokenization. we removed all the merges, sum of which exceeded the chosen maximum length of tokens.

Thus, we received a two files: one where all tokens were no longer than 6 and another where all tokens were no longer than 5.

Best obtained metrics are as shown in Table 4.4

Table 4.4: Standard-fine tuning results on core promoter detection task with maximum token length limited to 5 and 6.

	max_len=5		max_len=6		unlimited	
	MCC	F1	MCC	F1	MCC	F1
Gena-LM	0.6566	0.8249	0.6489	0.8109	0.635	0.817
DNABERT-2	0.6534	0.8142	0.6483	0.8257	0.67	0.8328

As can be derived from the values of metrics, the length of tokens didn't influence on the results significantly comparing with the original metrics where no additional restrictions on token length were imposed. However, it can be noted that for shorter tokens MCC is a bit higher.

For each token maximum length, we applied the Integrated Gradients method to get token significance scores.

Full interpretation results are available in our GitHub repository.

Notable significant tokens for max-6 tokenization are shown in Table 4.5

Table 4.5: Interpretation results for token maximum length of 6 with IG scores and occurence counts on promoter task. Scores were rounded to 4 decimal places.

Gena-LM				DNABERT-2			
№	Token	Score	Count	№	Token	Score	Count
5	TAA	0.4212	147	10	GGAGCC	0.3095	51
14	CCCCC	0.3519	9	12	GCTG	0.2887	522
18	GGAAG	0.3234	71	19	GCTCTG	0.2590	41
24	TGACG	0.3015	22	29	GCCCTC	0.2371	8

Notable significant tokens for max-5 tokenization are shown in Table 4.6

Table 4.6: Interpretation results for token maximum length of 5 with IG scores and uccurence counts. Scores were rounded to 4 decimal places.

Gena-LM				DNABERT-2			
№	Token	Score	Count	№	Token	Score	Count
4	CCCCC	0.3153	8	1	CTATA	0.3495	1
5	GCAAC	0.3132	4	9	GGGGA	0.1392	56
14	CCC	0.2639	774	15	TTATA	0.1274	2
22	ATT	0.2333	192	16	GCGCA	0.1269	71
24	TTCCC	0.2315	124	18	CCGG	0.1154	104
25	TAA	0.2311	129	22	CTGCC	0.1047	54
27	CC	0.2283	774	23	GCAGC	0.1016	45
29	TCACC	0.2261	50	25	GGCTG	0.1002	161
32	ACTCC	0.2207	67	34	GGCAA	0.0818	14
36	TCC	0.2155	1015	39	GCCCC	0.0724	121

One of the biologically-known sequences that mark promoter sequences are TATA-boxes. TATA-boxes are non-coding DNA sequences that are found in core promoter regions of DNA. They satisfy the TATAWAW form where W can be either T or A. Looking at some of the most significant tokens in both variants of tokenization, we can note TAA sequence in both tables. Presumably, this sequence is the end of TATA-box as it corresponds to regex WAW and is met in promoter areas often.

According to the article "DNA promoter task-oriented dictionary mining and prediction model based on natural language technology" published in Nature [15] "*In the positive sample, most of the top 10 words in terms of importance are two-letter combinations, such as "GC", "GGC", and "CCC", among which "GC" ranks first with its significant frequency of occurrence and attention score, showing its prominent importance in the positive sample. These GC-rich sequences, such as "GGC" and "CCC", are particularly relevant because they often serve as binding sites for transcription factors like Sp1 and p53, which are critical for the regulation of gene expression.*"

For Gena-LM, in both tables we see high importance of sequences containing repeating cytosine nucleotide, namely CCC $TTCC$, CC, TCC, $TCACC$ in Table 4.6 and $CCCCC$ both in Tables 4.5 and 4.6.

For DNABERT-2, it is seen from Table 4.6 that the method succesfully captures TATA-boxes (*CTATA, TTATA*). In addition, both tables show the significance of the GC-rich sequences mentioned in [15] (*GCGCA, CTGCC, GGCTG, GGCAA, GCCCC, GGAGCC, GCTCTG, GC-*

$CCTC$).

Overall, for both models tokenization with maximum length of 5 led to more biologically meaningful results.

Thereby, the importance of GC-rich sequences in promoter prediction mentioned in [15], CCC sequences in particular, is statistically proven by Integrated Gradients analysis of Gena-LM and DNABERT-2 models.

5 Z-DNA prediction task

5.1 Problem statement

For finetuning on this task we used the hg38 Kouzine et al dataset. As explained in [9], "*Kouzine et al determined Z-formation by the overlap of unpaired thymines detected using permanganate/S1 nuclease sequencing (ssDNA-seq) with Z-DNA–forming sequences predicted by Z-HUNT3. The thymines subject to modification were used to define the two BZ junctions where B-DNA transition to Z-DNA.*", so it is an experimentally generated dataset.

We divided the dataset into sequences of equal length and used them for training. For DNABERT-2, the length of sequences was chosen to be 128, and for GenaLM it was 512. As the data is heavily imbalanced, we included sequences with and without Z-DNA labels in the ratio of 1:3 (negative samples were chosen randomly). For now, we used stratified train/val/test splits with ratios 0.56/0.24/0.2 respectfully. In our further work for a publication we will use stratified cross-validation.

Z-DNA prediction task differs a lot from the promoter-classification task, since the dataset looks different. Each nucleotide in each sequence in the Z-DNA dataset is assigned by a label (1 or 0), depending on whether it is a part of the Z-flipon. Thus, the task transforms from whole-sequence classification to nucleotide classification.

5.2 Data transformation method

Our first step was to tokenize each sequence and predict if each token is a part of Z-DNA sequence or not. For each experiment we decided to choose a certain threshold (namely, 0.6 or 0.3) and assign a label to each token in a sequence by the following logic: if the percentage of 1-labeled nucleotides corresponding to the label is higher than the chosen threshold, then the whole token as labeled with 1. The token is labeled with 0 otherwise.

For example, let token $ATCT$ be a part of a major sequence $ACT\mathbf{ATCT}GT$.... Let the corresponding sequence labels be 000**0111**10.... Let the threshold be equal to 0.6.

The percentage of 1-labels in the labels corresponding to the $ATCT$ token is $\frac{3}{4} = 0.75 > 0.6$. Therefore, token $ATCT$ in this sequence is labeled as 1.

Another reasonable question is tokens of what maximum length should be used in tokenization. Of course, we can leave the pre-trained tokenizer untouched, however, out-of-the-box tokenizers might contain really long tokens that will be met only once or twice in the whole Kouzine dataset. Such small appearance in the dataset is problematic for analysis and interpretation.

Now the time for nucleotide prediction comes. Currently, we have described only the process of token classification, but remember that our main task is single nucleotide prediction. Since our models have BPE tokenization, this task is particularly tricky for them. The main question was how to transform token predictions to nucleotide ones.

We decided to transform predictions in the following manner: for each nucleotide that is a part of the token we assign the same label as the one of the token. For instance, let token $CGCGC$ in the sequence $AT\mathbf{CGCGC}GA...$ be a 1. Than all the corresponding nucleotides in the sequence will be 1: $**\mathbf{11111}**$. Here a star stands for other labels derived from neighbouring tokens. Such method may seem bold, but with our approach to shortening tokens and with the right threshold, this way of assigning labels does not heavily affect the result.

5.3 Chosen metrics

For model performance evaluation we chose the following metrics: F1 and Mathews Correlation Coefficient. Though we also added precision in the benchmark, the first two named metrics are key for our task.

In particular, F1 is important for imbalanced datasets and Kouzine is an example of such dataset. The reason for this is that F1 score tends to find balance between precision and recall. It is vital for us, because it is quite ease for our models to reach high accuracy on the data (even if the model predicts all-negative labels) but we still want to maintain satisfactory precision.

MCC is also a metric which accounts for class imbalance. This metric was used for comparing models on the GUE dataset in [17], so we decided to keep it here as well.

5.4 Standard fine-tuning experiments

Experiments with DNABERT-2 were done by Yana and experiments with GenaLM were done by Ira.

Firstly, we needed to rewrite the Dataset formation to put it correctly into the transformers Trainer and implement the idea of thresholds in token labeling. The difficult aspect here was that all the preparations of original sequences were complex in timing and our initial attempt to form the Dataset were processed too long. That is why in the current realization all sequences in the Dataset are processed at the time of them being retrieved and called by their index.

As DNABERT-2 was not originally suitable for token classification task, a custom token classification head was written, using DNABERT-2's source code. GenaLM is suitable for token classification out of the box.

The two hyperparameters that we were experimenting with are maximum token length and labeling threshold.

After the experiments on promoter classification it was concluded that there is no need in leaving BPE tokenizer with no restrictions imposed. For comparison, the Z-DNABERT model uses 6-mers in its realization. For GenaLM, three maximum token lengths were tested: 4, 5 and 6. For DNABERT-2, maximum lengths of 5 and 6 were tested.

As for the threshold, the question was how high should it be and how its value will influence the quality of single-nucleotide prediction. We tested thresholds of values 0.3, 0.6, 0.66 and 0.7. These thresholds were chosen from the possible nucleotide ratios, e.g. for threshold 0.7 we allow no more than one negative nucleotide per positive token of length 3, 5 or 6. We try to not choose too high of a threshold in order to maintain balance between negative nucleotides and positive tokens in general. Because our models have non-overlapping tokenization, this is important - the Z-DNA region might be on the edge of two neighbouring tokens, but not fully covering them. If the threshold is too high, we would treat both of the tokens as negatives and skip a Z-DNA region. So, we want to allow some "mistakes", but not too many.

To transform token predictions to nucleotide ones a script was written. At first it formed single nucleotide predictions and than compared them with true nucleotide labels. The quality of prediction was measured with F1, MCC and precision metrics.

Each variant of the model was trained for 10 epochs and evaluated token-wise on each epoch. Then, the best checkpoint for each model was evaluated nucleotide-wise. For each model, fine-tuning took about 2 hours.

The baseline we were orienting at was the result of Z-DNABERT model, that reached F1 score of 0.83 on nucleotide prediction task.

Table 5.1 shows the results of experiments.

Table 5.1: Standard finetuning results. Metrics are computed for nucleotide prediction task on validation set. Highest obtained F1 scores are in bold.

	GenaLM			DNABERT − 2		
	F1	MCC	Precision	F1	MCC	Precision
max_len = 4, thr = 0.7	0.8057	0.8039	0.4219	-	-	-
max_len = 5, thr = 0.3	0.797	0.731	0.798	-	-	-
max_len = 5, thr = 0.6	0.8492	0.8425	0.8644	0.8437	0.8378	0.8080
max_len = 5, thr = 0.66	0.855	0.8491	0.8733	0.8433	0.8371	0.8155
max_len = 5, thr = 0.7	**0.8614**	0.8534	0.9009	**0.8482**	0.8454	0.7751
max_len = 6, thr = 0.6	0.816	0.8081	0.8398	0.8263	0.8203	0.7777
max_len = 6, thr = 0.66	0.815	0.813	0.796	0.8329	0.8261	0.8456
max_len = 6, thr = 0.7	-	-	-	0.8263	0.8190	0.8219

As seen from Table 5.1, the best overall result were obtained from Gena-LM. It was reached by a three-staged model. Firstly, it was fine-tuned on the task of Z-DNA prediction on sequence of length 510 bp with threshold equal to 0.3. Secondly, a pretrained model was trained on the same task with sequence size 110 bp and threshold 0.66. At last, the model was trained on sequences of 70 bp length with a threshold=0.7. In all cases max_len = 5. This configuration yielded F1=0.8626 and MCC=0.8551 on validtion set.

For DNABERT-2 the best result was achieved on a configuration fine-tuned with token maximum length of 5 and label threshold of 0.7, yielding F1=0.8482 and MCC=0.8454 on validation set.

It can be observed that for both models the best max_len parameter turned out to be 5, and for both models the optimal threshold is between 0.6 and 0.7.

The standard fine-tuning took some time, so we decided to try to optimize it.

5.5 Parameter-Efficient-Fine-Tuning with LoRA

We decided to experiment and apply LoRA in the task of fine-tuning. It should have minimized the time on training and eliminate overfitting, though the applicability of a LoRA-trained model was questionable as the quality of predictions was expected to be lower.

We launched LoRA with the following hyperparameters.

```
config = LoraConfig(
    r=16,
    lora_alpha=32,
    lora_dropout=0.05,
    bias="none",
    task_type="TOKEN_CLS"
)
```

We applied LoRA fine-tuning for models configurations that achieved best results in 5.1. Fine-tuning was done for 10 epochs, models were evaluated based on F1 score.

Table 5.2 shows the results of experiments.

Table 5.2: LoRA finetuning results. Metrics are computed for nucleotide prediction task on validation set.

	GenaLM			DNABERT − 2		
	F1	MCC	Precision	F1	MCC	Precision
max_len = 5, thr = 0.66	0.7516	0.7485	0.7279	-	-	-
max_len = 5, thr = 0.7	-	-	-	0.7844	0.7866	0.6695

Metrics appear to be much lower, though it is predictable for LoRA. After receiving the results, we did not concentrate on LoRA fine-tuning for too long as the main goal of fine-tuning was achieving better results on single-nucleotide predictions. Moreover, we did not win much in time - the time required for fine-tuning stayed practically the same as for standard fine-tuning.

5.6 Final metrics

We took the best obtained models and evaluated them on the test set. The final metrics are shown in Table 5.3

Table 5.3: Final metrics of nucleotide-wise predictions on the test set.

	F1	MCC	Precision
Gena-LM	0.8626	0.8551	0.9024
DNABERT-2	0.8908	0.8878	0.8399

For DNABERT-2 the parameters were max_len=5, label threshold=0.7.

For Gena-LM max_len=5 and it was trained in 3 stages:

1. On sequences of length 510 bp with threshold=0.3
2. On sequences of length 110 bp with threshold=0.66
3. On sequences of length 70 bp with threshold=0.7.

Although on validation set DNABERT-2 showed worse results than Gena-LM, on test set it yielded better results (F1=0.89 for DNABERT-2 and F1=0.8626 for Gena-LM).

As it is seen from Table 5.3, our methods led to large improvements of previous-best metrics of Z-DNABERT with F1=0.83.

With those best models that we obtained, we proceed to interpret their results.

5.7 Z-DNA model interpretation

In the article "Z-Flipon Variants reveal the many roles of Z-DNA and Z-RNA in health and disease" [9] a list of Z-DNABERT attention rank is provided. The highest attention rank on Kouzine et al dataset is obtained by $GCGCGC$, $GTGTGT$, $CGCGCG$, $ACACAC$ and $TGTGTG$ k-mers in the order of their mentioning. The importance of these k-mers is biologically-driven as Z-DNA is mostly found in alternation purine-pyrimidine sequences, e.g. $(CG)_n$ and $(TG)_n$. All the k-mers marked important by Z-DNABERT are examples of purine-pyrimidine alternation and are truly important in Z-DNA formation.

5.7.1 Analyzing k-mers inside Z-DNA sequences

Most of the current interpretation methods are quite long for fast model benchmarking. That is why we also used the following idea to measure the importance of tokens in Z-DNA prediction.

For the fine-tuned model, we iterated through the test dataset and looked at the classes assigned to tokens and their true labels. We counted a value equal to $\frac{\mathbf{TP}}{\mathbf{Total}}$ where **TP** is the number of occurrences of a certain token as a True Positive one and **Total** is the total number of times this token was met in a test dataset.

NB: this method will not catch k-mers associated with Z-DNA if they are located outside of the Z-DNA-region in the sequence.

The head of the interpretation file for the leading models is provided below.

Table 5.4: Interpretation of token importance in TP predictions via TP ratio. Scores are rounded to 4 decimal places.

	Gena-LM			DNABERT-2		
№	Token	Score	Count	Token	Score	Count
1	GCGCG	0.8230	339	CACGC	0.7176	255
2	GCGC	0.6916	3201	GCGC	0.6396	2470
3	GCACG	0.6564	131	CGC	0.4864	6658
4	ACGCG	0.6036	217	GCGTG	0.4532	759
5	GCGTG	0.5462	108	GCGCA	0.4496	745
6	TGCGC	0.5042	827	GGCGC	0.4424	556
7	ACGC	0.4842	2121	CGCA	0.4280	3549

The importance of tokens is biologically correct and it proves that Z-DNA is mainly formed

in areas with purine-pyrimidine alternation.

5.7.2 Analyzing k-mers via Intergated Gradients

For the interpretation we wrote a script using TokenClassificationExplainer from transformers_interpret library. Its idea is similar to the script for promoter prediction interpretation but there are some differences regarding token classification. The gist is in initializing TokenClassificationExplainer with our model and tokenizer, then iterating though sequences from test dataset and counting word attributions for them. For each token that is positive and is truly labeled by our model we choose top 10 tokens by their attribution scores and store counts and sums for each token. Eventually we get a mean score for each token that shows how much this token is associated with giving a true positive prediction on Z-DNA nucleotide classification task.

In Table 5.5 the results of IG interpretation is listed for both our models.

Table 5.5: Interpretation of token importance using IG.

	Gena-LM			DNABERT-2		
№	Token	Score	Count	Token	Score	Count
1	AAATC	0.9576	1	CTGTT	0.6490	4
2	GCGTG	0.8032	28	GCGTG	0.5938	51
3	ACGCG	0.6589	31	CGTG	0.5206	289
4	GCGCG	0.6474	99	TACTA	0.5168	1
5	GCGC	0.5584	43	GCGC	0.5006	319
6	GCATG	0.5248	30	GCGCA	0.4859	76
7	GCG	0.5186	752	CGC	0.4643	761

The results are similar to those from TP ratio method. However, the IG method also captured some tokens with low occurence count (*AAATC, CTGTT, TACTA*) which were not seen before.

5.7.3 LoRA token interpretation results

Although LoRA fine-tuning metrics were worse than those from standard fine-tuning and this PEFT method did not win us much time, we still wondered if the significant tokens would differ.

We took the best LoRA models' checkpoints and ran them through our Integrated Gradients pipeline. Resulting significant tokens are shown in Table 5.6.

Table 5.6: Interpretation of token importance using IG for LoRA fine-tuned models.

	Gena-LM			DNABERT-2		
№	Token	Score	Count	Token	Score	Count
1	GCGCG	0.6082	411	CACGC	0.7215	255
2	GCGC	0.4544	4841	GCGC	0.6400	2470
3	GCGTG	0.3432	201	CGC	0.4860	6658
4	GCACG	0.3206	237	GCGTG	0.4505	759
5	ACGCG	0.3176	340	GGCGC	0.4460	556

Alas, there were no insights in interpretation. Important tokens stayed practically the same. It can be concluded that this method is suitable for token interpretation but inappropriate for single-nucleotide prediction as the quality is too low.

5.8 Interpretation ranking

For now for each model we have two different files with interpretation results from Integrated Gradients and TP ratio methods. A different research group developed a script for ranking tokens based on results of several interpretation methods. Their ranking approach is based on computing percentage deviation for each interpretation method from it's mean. Then, these deviation scores are averaged for each token across all interpretation methods and all models (if several are present).

We apply this approach to get token scoring for each model and unified token ranks for both models.

In Table 5.7 token ranking results for each model are shown.

Table 5.7: Ranked token importance for both models based on TP ratio and IG methods. Mean deviation value is truncated to integers.

	Gena-LM		DNABERT-2	
№	Token	Mean_dev	Token	Mean_dev
1	GCGCG	1239	CACGC	722
2	GCGC	1032	GCGC	678
3	GCACG	988	GCGTG	530
4	ACGCG	968	CGC	519
5	GCGTG	962	GCGCA	491
6	TGCGC	765	GGCGC	463
7	ACGC	710	CGTG	458
8	GCG	673	CGCA	452
9	GTGC	660	CCCGC	408
10	GTGTG	602	TACGC	277

The resulting most high-ranked tokens are really similar across models. It is also worth noting that for Gena-LM model the mean deviation value is generally higher than for DNABERT-2.

Table 5.8 shows unified token ranking results for both models. For tokens that were present only in one model's tokenization and not the other's for each method a score of 0.0 was set.

Table 5.8: Ranked token importance across both models based on TP ratio and IG methods. Mean deviation value is truncated to integers.

№	Token	Mean_dev
1	GCGC	1454.8149
2	GCGTG	1343.7900
3	CCCGC	818.2007
4	GCGCC	815.1126
5	GTGTG	747.6250
6	CGC	675.4476
7	GGCGC	630.4143
8	GCATG	568.2176
9	TACGC	530.5543
10	GGGC	518.6210

As seen from Table 5.8, resulting rank tokens are consistent with previously discovered k-mers that are associated with Z-DNA formation, purine-pyrimidine alternating sequences. In our resulting rank, $(CG)_n$ sequences are more common. Moreover, they are in line with Z-DNABERT's attention ranked tokens, accounting for limited token length in our models.

The full list of ranked tokens is available on our GitHub.

6 Conclusion

In our current research we achieved the following results:

1. We evaluated DNABERT-2 and Gena-LM models on the task of core promoter detection, confirming the models' ability to train and capture meaningful biological relations via xAI methods

2. We fine-tuned DNABERT-2 and Gena-LM models on the task of Z-DNA prediction. This required alternating the dataset and models' classification heads, as previously this task was not done with non-overlapping tokenization. We also came up with a way to transfer nucleotide labels to token labels and token predictions to nucleotide predictions in order to compare our models to previous nucleotide-wise approaches. Different fine-tuning experiments were held.

3. We were able to outperform the existing benchmark of Z-DNABERT (F1 = 0.83) and improved it significantly (F1=0.89 for DNABERT-2, F1=0.86 for Gena-LM).

4. We applied two interpretation pipelines to our models' predictions. Based on the results, we were able to outline key tokens associated with Z-flipons using our colleagues' ranking script. The resulting k-mers correspond to known biological sequences.

Currently we are preparing a publication on the topic. In our further work more experiments with token labeling, hyperparameters, architectures and fine-tuning will be held. We will also apply more xAI methods to obtain more robust interpretation results.

One can find our results in GitHub repository [14].

References

[1] Wang AJ., G. Quigley, and F et al Kolpak. "Molecular structure of a left-handed double helical DNA fragment at atomic resolution". In: *Nature* 282 (1979), pp. 680–686.

[2] N. Beknazarov, S. Jin, and M. Poptsova. "Deep learning approach for predicting functional Z-DNA regions using omics data". In: *Scientific Reports* 10.19134 (2020).

[3] Nazar Beknazarov, Dmitry Konovalov, Alan Herbert, and Maria Poptsova. "Z-DNA formation in promoters conserved between human and mouse are associated with increased transcription reinitiation rates". In: *Scientific Reports* 14.17786 (2024).

[4] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. "LoRA: Low-rank adaptation of large language models." In: *ICLR* 1.2 (2022), p. 3.

[5] Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. "DNABERT: pre-trained Bidirectional Encoder Representations from Transformers model for DNA-language in genome". In: *Bioinformatics* 37.15 (2021), pp. 2112–2120.

[6] F. Kouzine, D. Wojtowicz, L. Baranello, A. Yamane, S. Nelson, W. Resch, K.-R. Kieffer-Kwon, C. J. Benham, R. Casellas, T. M. Przytycka, and D. Levens. "Permanganate/S1 nuclease footprinting reveals Non-B DNA structures with regulatory potential across a mammalian genome". In: *Cell Systems* 4.3 (2017).

[7] Ofir Press, Noah A Smith, and Mike Lewis. "Train short, test long: Attention with linear biases enables input length extrapolation". In: *arXiv preprint arXiv:2108.12409* (2021).

[8] Mukund Sundararajan, Ankur Taly, and Qiqi Yan. "Axiomatic attribution for deep networks". In: *International conference on machine learning*. PMLR. 2017, pp. 3319–3328.

[9] Dmitry Umerenkov, Alan Herbert, Dmitrii Konovalov, Anna Danilova, Nazar Beknazarov, Vladimir Kokh, Aleksandr Fedorov, and Maria Poptsova. "Z-Flipon Variants reveal the many roles of Z-DNA and Z-RNA in health and disease". In: *bioRxiv* (2023). DOI: 10.1101/2023.01.12.523822. eprint: https://www.biorxiv.org/content/early/2023/01/13/2023.01.12.523822.full.pdf. URL: https://www.biorxiv.org/content/early/2023/01/13/2023.01.12.523822.

[10] P Vasudevaraju, RM Garruto, K Sambamurti, KSJ Rao, et al. "Role of DNA dynamics in Alzheimer's disease". In: *Brain research reviews* 58.1 (2008), pp. 136–148.

[11] Artem Voytetskiy, Alan Herbert, and Maria Poptsova. "Graph Neural Networks for Z-DNA prediction in Genomes". In: *IEEE* 10.10.1101/2022.08.23.504929 (2022).

[12] Guliang Wang and Karen M. Vasquez. "Z-DNA, an active element in the genome". In: *FBL* 12.7 (2007), pp. 4424–4438.

[13] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. *HuggingFace's Transformers: State-of-the-art Natural Language Processing*. 2020. arXiv: 1910.03771 [cs.CL]. URL: https://arxiv.org/abs/1910.03771.

[14] Irina Skvortsova Yana Libina. *Identifying Biological Relations in Human Genome Through Interpretation of Language Models*. URL:https://github.com/yalibina/ZDNA_LLM. 2025.

[15] Ruolei Zeng, Zihan Li, Jialu Li, and Qingchuan Zhang. "DNA promoter task-oriented dictionary mining and prediction model based on natural language technology." In: *Nature* 15.153 (2025).

[16] Ting Zhang, Chaoran Yin, Aleksandr Fedorov, Liangjun Qiao, Hongliang Bao, Nazar Beknazarov, Shiyu Wang, Avishekh Gautam, Riley M Williams, Jeremy Chase Crawford, et al. "ADAR1 masks the cancer immunotherapeutic promise of ZBP1-driven necroptosis". In: *Nature* 606.7914 (2022), pp. 594–602.

[17] Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana Davuluri, and Han Liu. "DNABERT-2: Efficient Foundation Model and Benchmark For Multi-Species Genomes". In: *arXiv:2306.15006* (2023).