Research Project
Identification of biological interrelations in the human genome by methods of language model interpretation

Data Science and Business Analytics

Moscow 2025

Исследовательский проект
Идентификация биологических взаимосвязей в геноме человека методами интерпретации языковых моделей

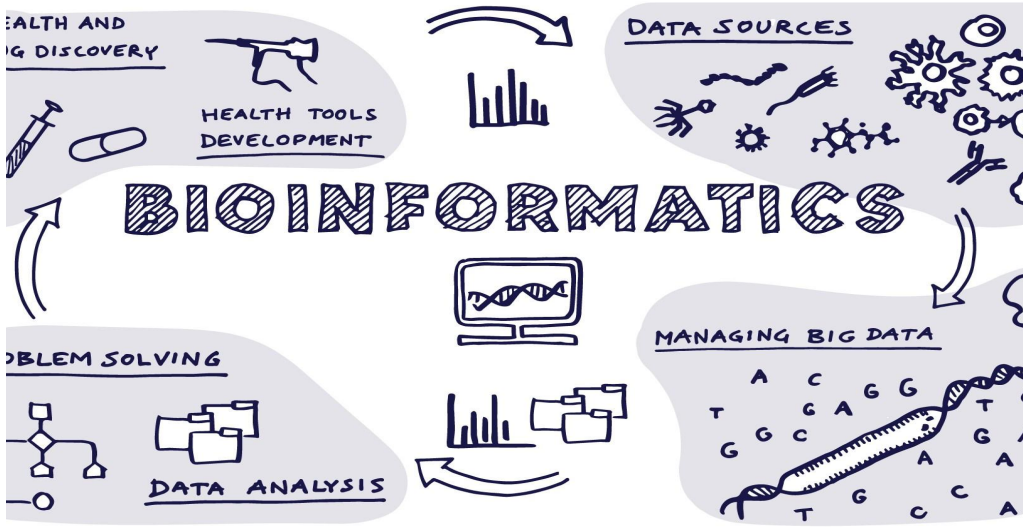
Author:
Bokhyan Roman Beniaminovich, БПАД 222

Project Supervisor:
Borevsky Andrey Olegovich
Assistant
Faculty of Computer Science, HSE University

Faculty of Computer Science

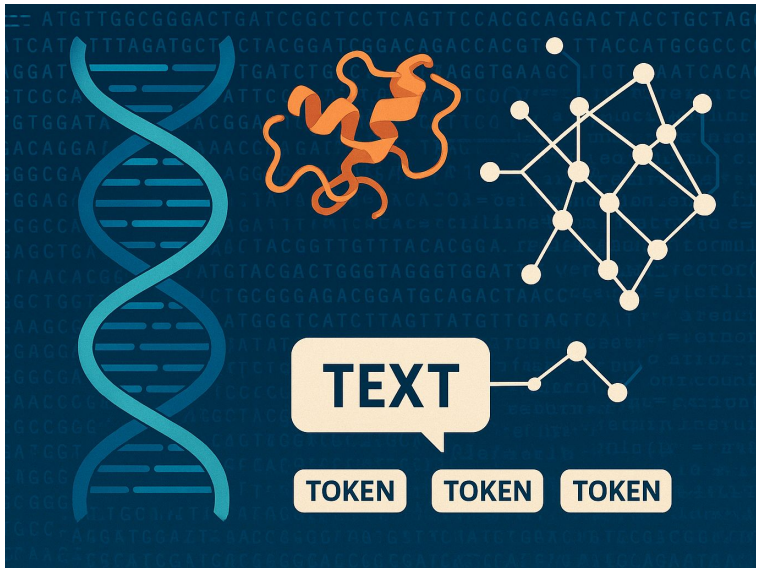
Stating the problem

- Traditional techniques for DNA analysis are rapidly losing relevance.
- The human genome is an immense repository of information, brimming with answers that demand exploration.
- What outcomes arise when these questions are investigated using large language models?



It is not just a coursework

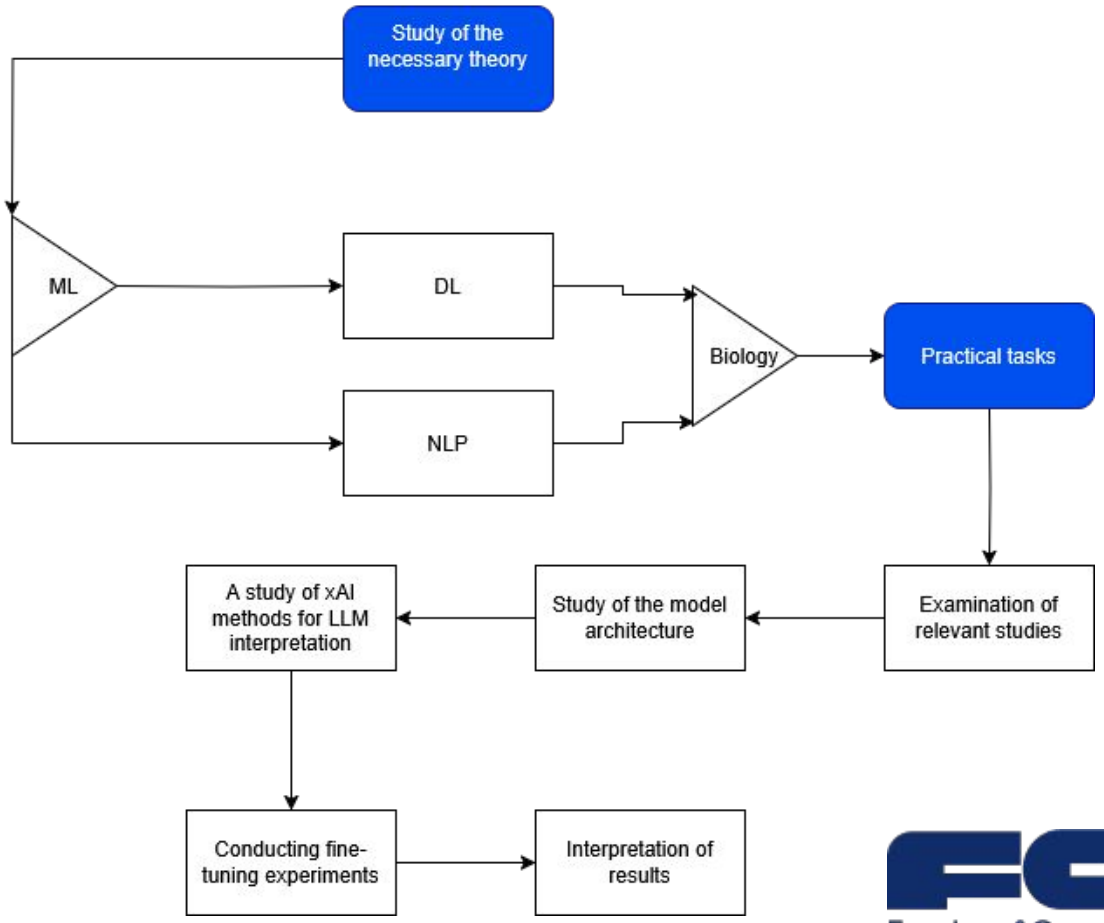
- Part of the HSE Bioinformatics Lab's study on LLM interpretation of complex, rare DNA structures
- Applied a novel LLM architecture to predict DNA secondary structures
- Conducted experiments and leveraged xAI techniques to assess model performance
- Compared model outputs against established biological insights
- Delivered a methodological breakthrough to guide future bioinformatics research



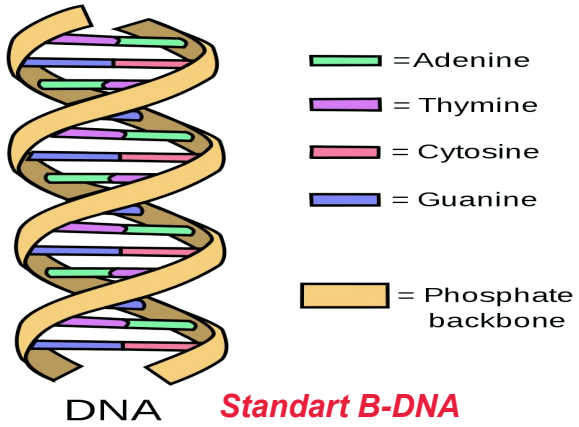
Formal objective

Development of a large language model for predicting double helix structures and interpretation of its performance

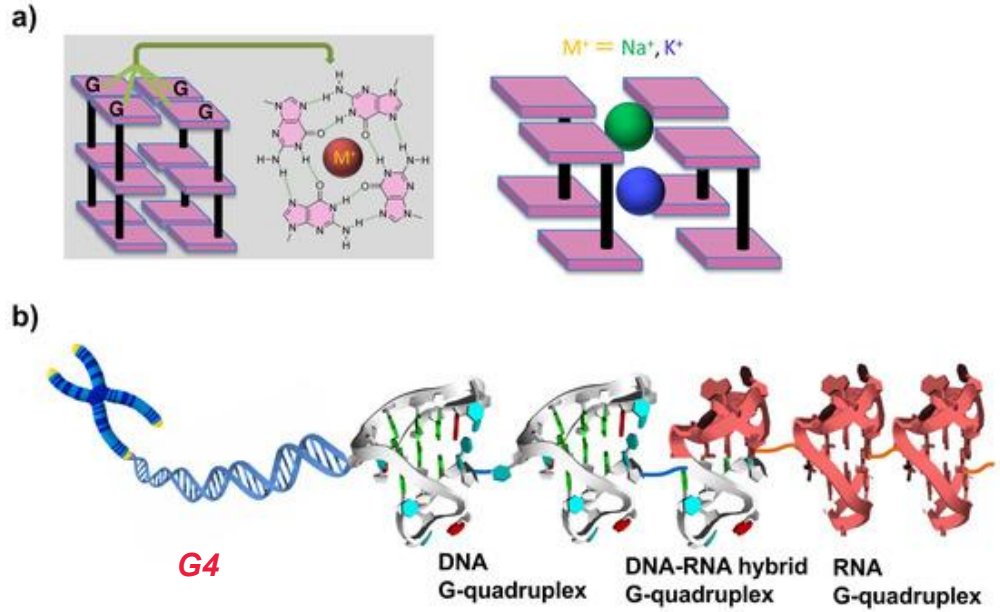
Tasks for the implementation of the research

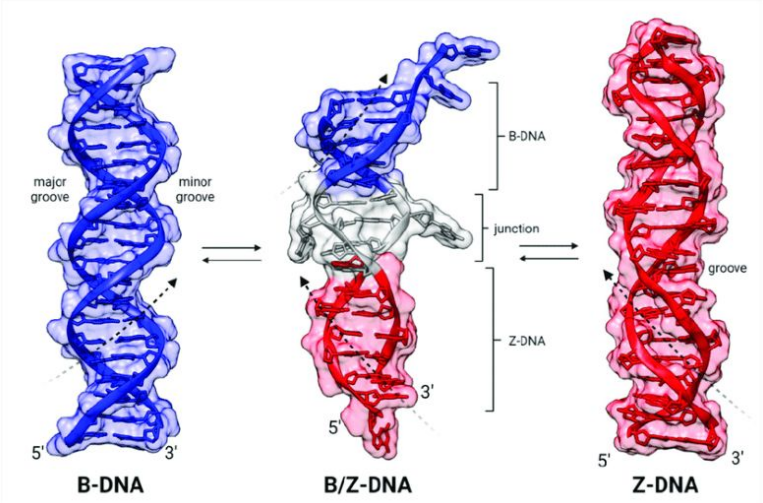


1. **Standard B-DNA:** Right-handed double helix built from A, T, C, and G bases
2. **Z-DNA:** A left-handed helix that can form in alternating CG regions, often transient during active transcription.
3. **G-quadruplexes (G4):** Stacks of four guanines held together by metal ions (e.g., Na^+ or K^+), found in promoters and regulatory regions
4. **Promoter:** DNA segment where transcription begins.

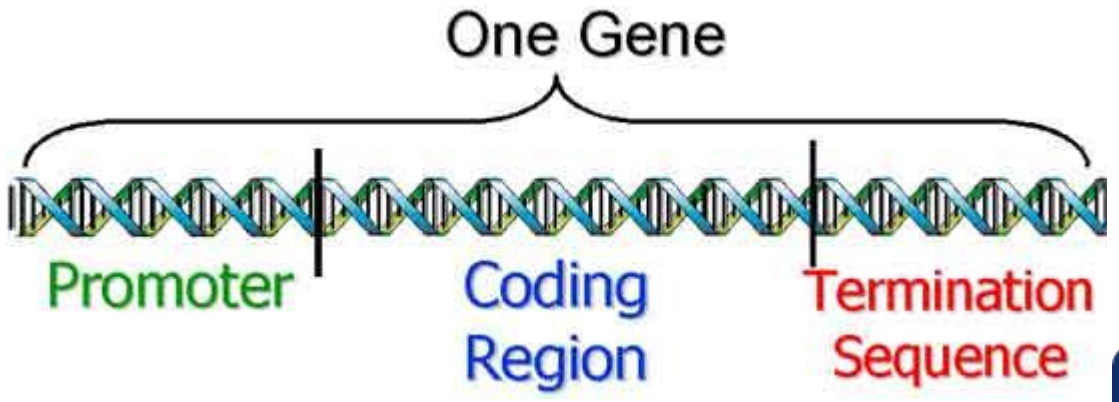


Standart B-DNA





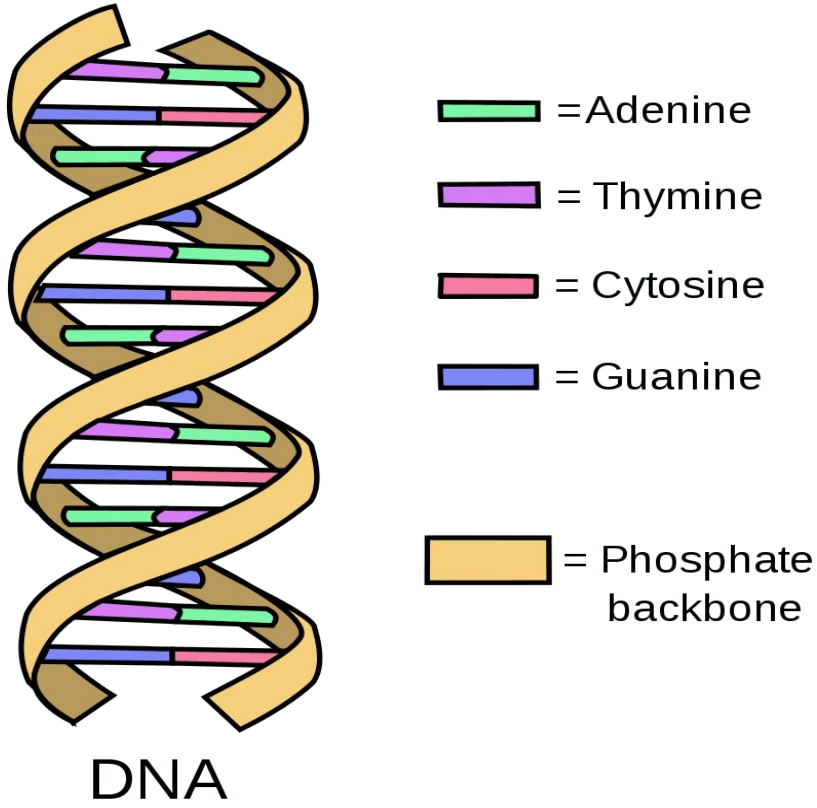
Z-DNA



Promoter

Deoxyribonucleic acid (DNA)

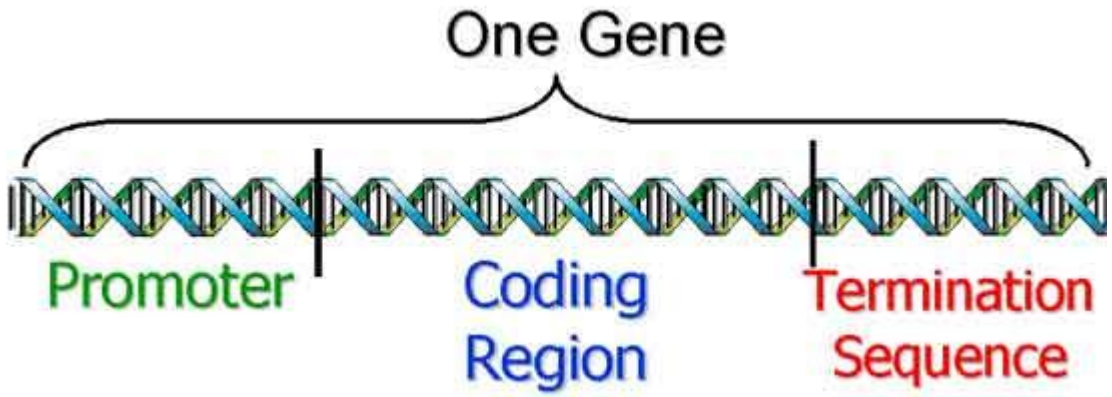
- A macromolecule that carries the genetic information necessary for the development, function and reproduction of all known organisms and most viruses.
- A sequence of length 3 billion
- Each "line of code" in DNA is a nucleotide, and sets of four "letters" (A, T, G, C) form "operators" - a similar pair of characters in the code. The two DNA strands are twisted together as two version control branches that are always synchronised (A is always "paired" with T, G with C).
- DNA is the human code, it describes the whole person.



DNA

Promoter

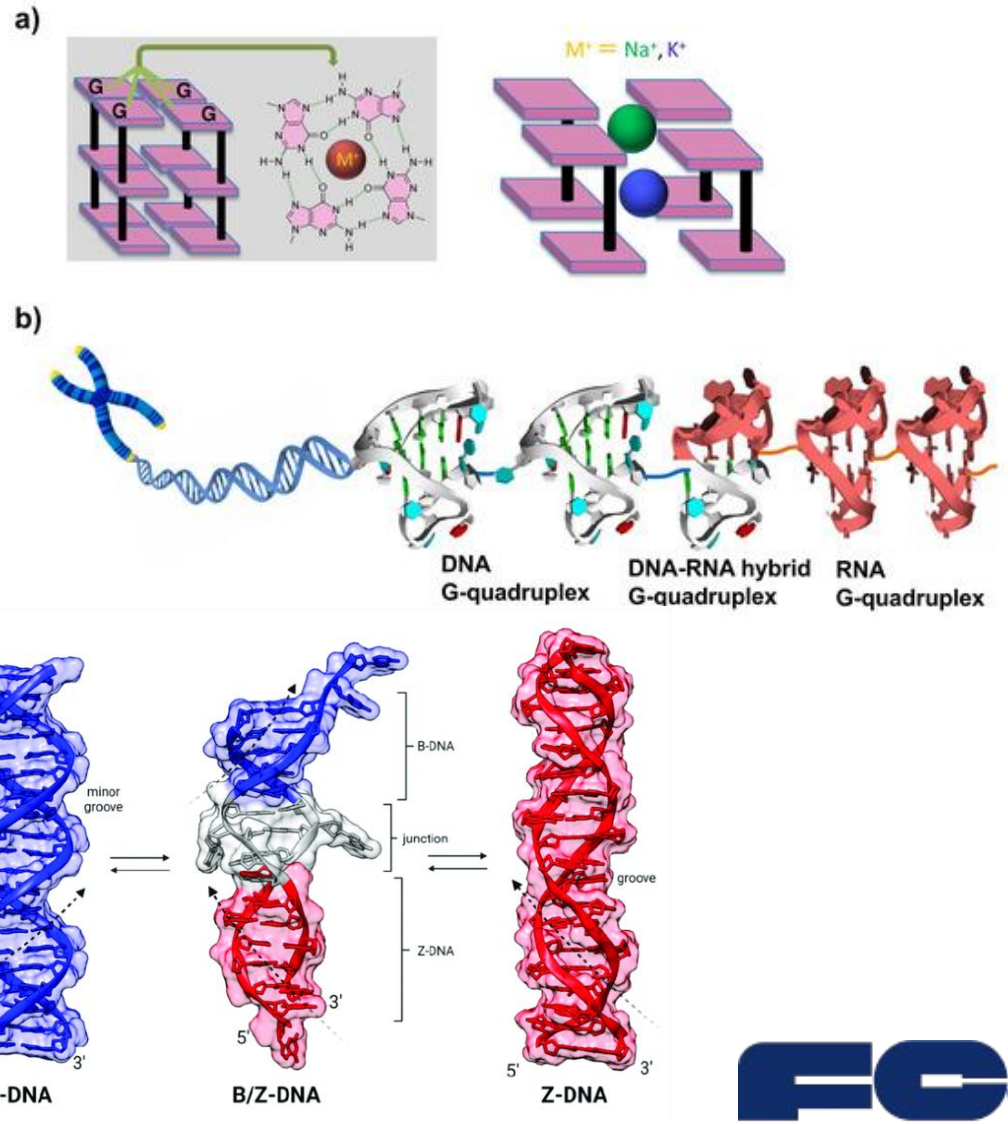
- It is a section of DNA to which RNA polymerase(reads the part of DNA to make it analysable) and transcription factors attach when transcription is initiated, providing the start of RNA synthesis. In other words it is a sign that coding region is close.
- Without a promoter, the "compiler" (transcription complex) will not understand where in the source the necessary piece is located and from which place it is necessary to "start" copying.



Secondary DNA structures

- DNA double helix formed by complementary base pairing and base stacking
  - Includes the common right-handed B-helix
  - Alternative conformations: Z-helix (Z-DNA) and G-quadruplexes (G4)
- **Z-DNA (Z-helix)**
  - Left-handed double helix forming in (CG)n regions with alternating bases
  - Can transiently switch between B-form and Z-form
  - Involved in transcription regulation, RNA-editing protein interactions, and innate immune responses
  - Linked to Alzheimer's disease and being explored as a potential anticancer target
- **G-quadruplex (G4) structures**

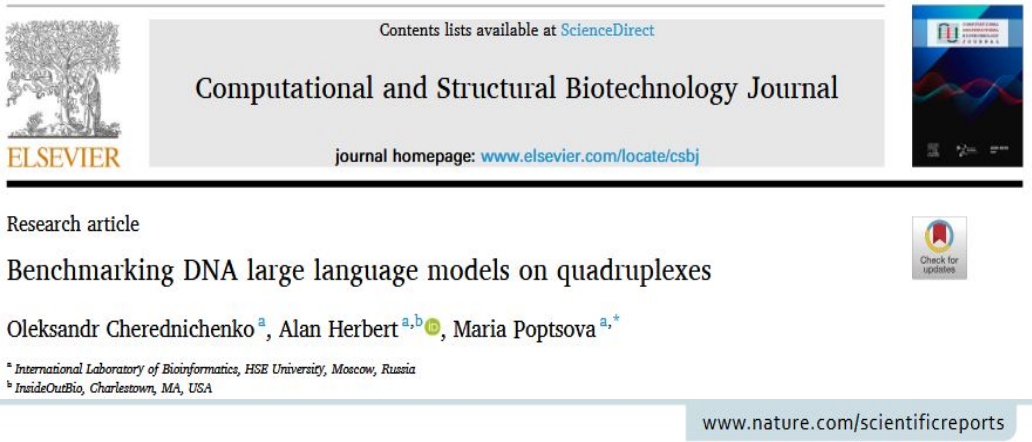
  Consist of four guanines stacking into a quadruplex helix
  - Frequently found in gene promoters, including oncogenes
  - Influence transcription factor binding to repress or activate gene expression
  - Stabilizing G4s is pursued as a strategy for novel drug development

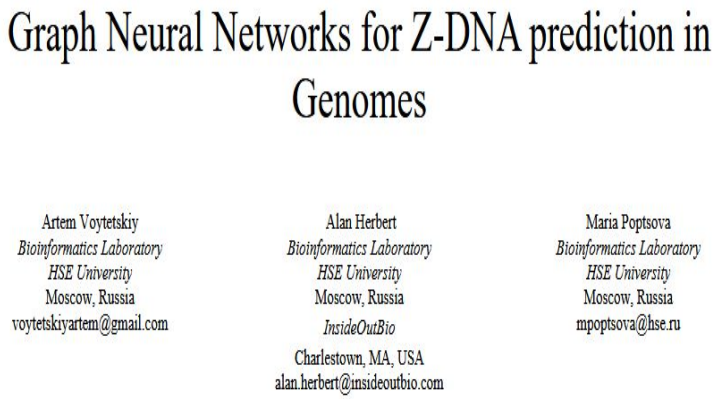


Faculty of Computer Science

Relevant papers

1. DeepZ (Nazar Beknazarov, Seungmin Jin, Maria Poptsova, 2020) – RNN F1 = 0.40
2. GraphZ (Artem Voytetskiy, Alan Herbert, Maria Poptsova, 2022) – GNN F1 = 0.124
3. Z-DNABERT (Dmitry Umerenkov et al, 2023) – LLM Bert F1 = 0.83
4. Benchmarking DNA LLMs on G4 (Oleksandr Cherednichenko, Alan Herbert, Maria Poptsova, 2025) - LLM HyenaDNA F1 = 0.75





Faculty of Computer Science

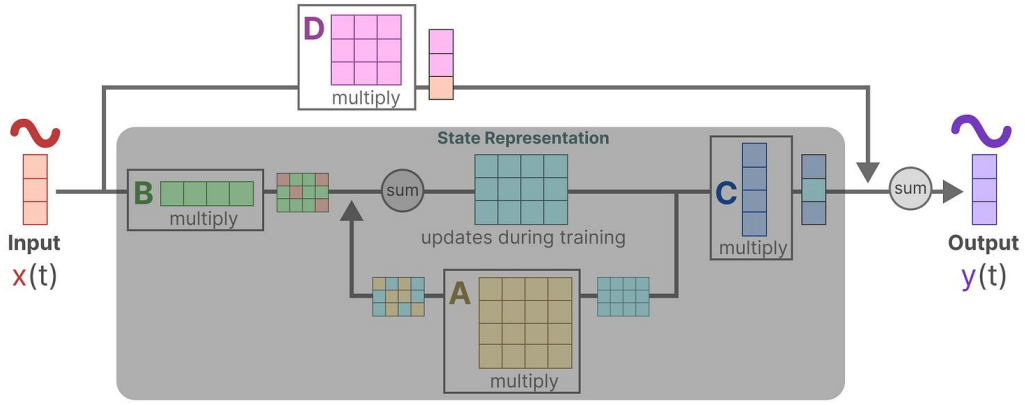
HumanGenome38 Kouzine et al

- Mapped single-stranded DNA in living cells using Kouzine et al. data.
- Overlapped these regions with known G4 sites many matched, confirming G4 formation in cells.
- Identified Z-DNA by locating unpaired T bases in Z-DNA prone sequences, marking B↔Z transitions.
- The data covers ~ 41k regions.

Genome Understanding Evaluation (GUE)

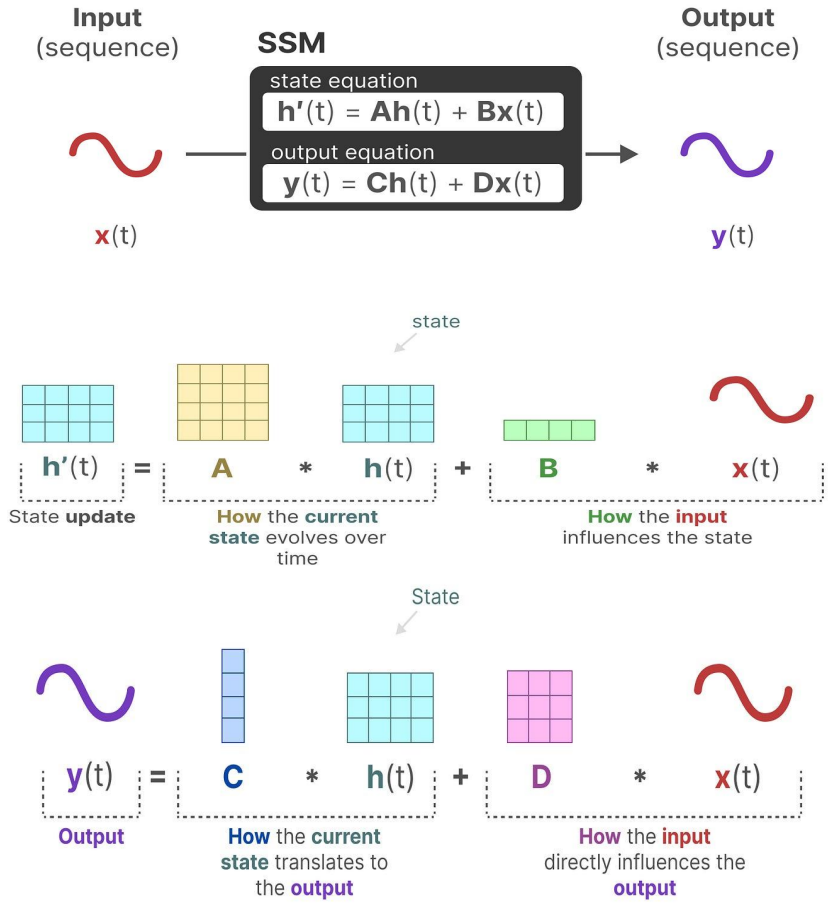
- GUE for promoter analysis.
- A genome classification benchmark
- Consists of ~60k regions

State Space Models



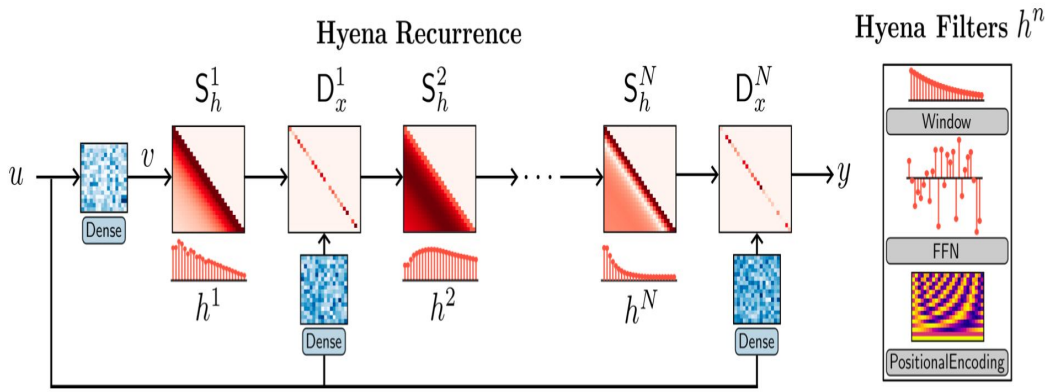
State Space Model

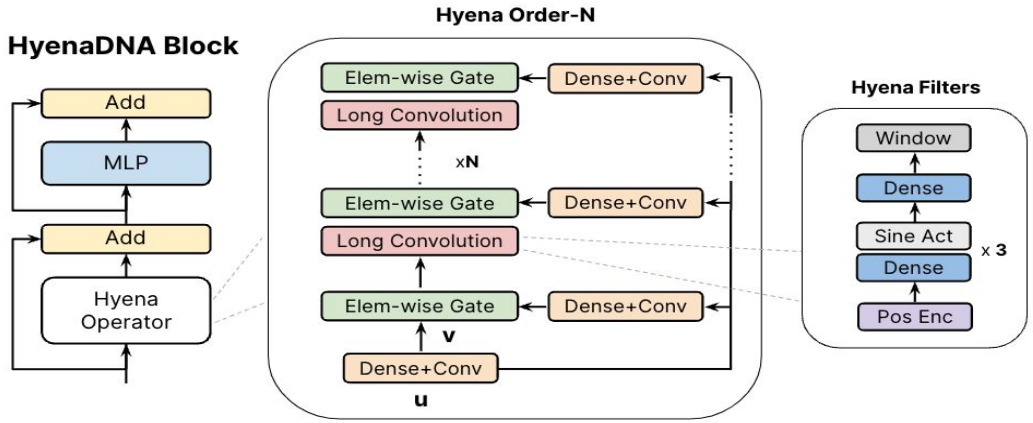
An SSM is like a smart memory box that gets updated with each new input in a fast, rolling way. No need to compare every past item to every other. Unlike self-attention, which looks at all pairs of positions, SSM simply carries forward a single state that already captures everything it's seen, allowing it to handle extremely long sequences with far less computation.



Faculty of Computer Science

Hyena Operator

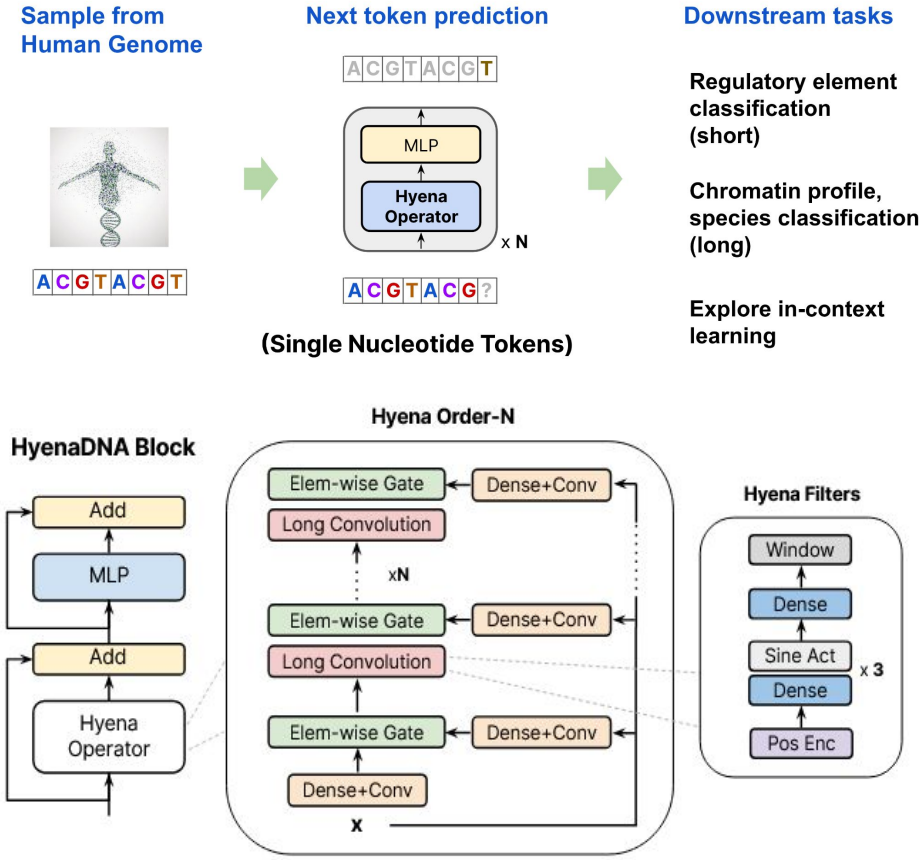




Hyena swaps the O(L²) attention step for implicit long convolutions plus element-wise gating. It projects inputs into multiple streams, uses a small MLP to generate convolution filters on the fly, and applies FFT to compute convolutions in O(L log L) time. Gating then steers the information flow. Because filters are generated rather than stored, the parameter count stays fixed enabling efficient modeling of contexts up to ~1 million tokens without losing transformer-level performance.

HyenaDNA

- HyenaDNA, a decoder based foundation model for DNA analysis capable of processing sequences up to 1 million in length at the single character level, was chosen as the model.
- Model pre-trained on the entire human reference genome
- By combining these two new approaches, the authors have presented a variant of LLM that works for sub-quadratically complexity and having much less parameters.

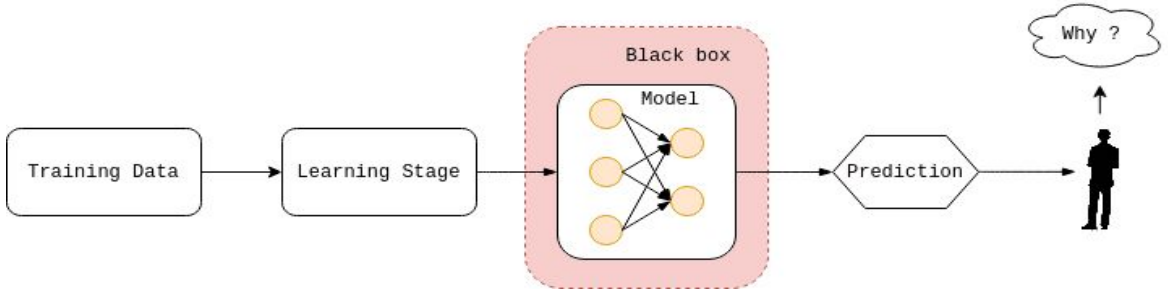


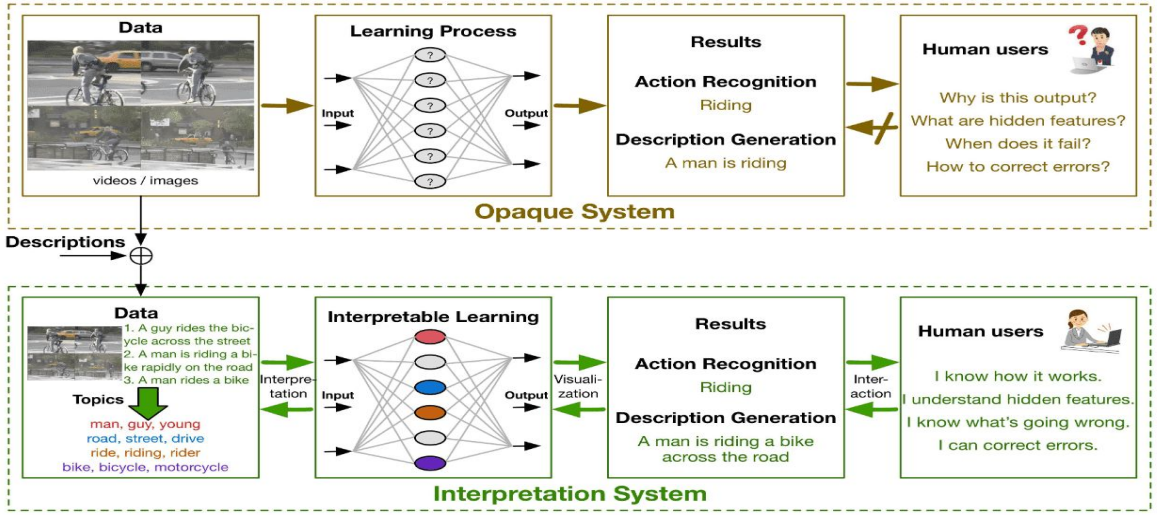
Explainable AI (xAI) or Interpretation

- Interpretability describes how clearly we can see why an algorithm made a particular decision.
- Aims to open the “black box” of complex models so we can see why they make certain decisions.
- Compute how the output changes when you tweak each input slightly, highlighting key features.
- Builds user confidence by showing “what the model looked at” when making a prediction.

$$\mathrm{IG}_i(x) \approx (x_i - x_i') \frac{1}{m} \sum_{k=1}^{m} \frac{\partial F\left(x' + \frac{k}{m}(x - x')\right)}{\partial x_i}$$

$$\mathrm{SmoothGrad}(x) = \frac{1}{n} \sum_{k=1}^{n} \frac{\partial F\left(x^{(k)}\right)}{\partial x}$$





Integrated Gradients

- Assigns each input feature a contribution score by accumulating the model's gradients along a straight-line path from a baseline input x' (often all zeros) to the actual input x.
- In practice, this integral is approximated by summing over m steps.

$$\mathrm{IG}_i(x) \approx (x_i - x_i') \frac{1}{m} \sum_{k=1}^{m} \frac{\partial F\left(x' + \frac{k}{m}(x - x')\right)}{\partial x_i}$$

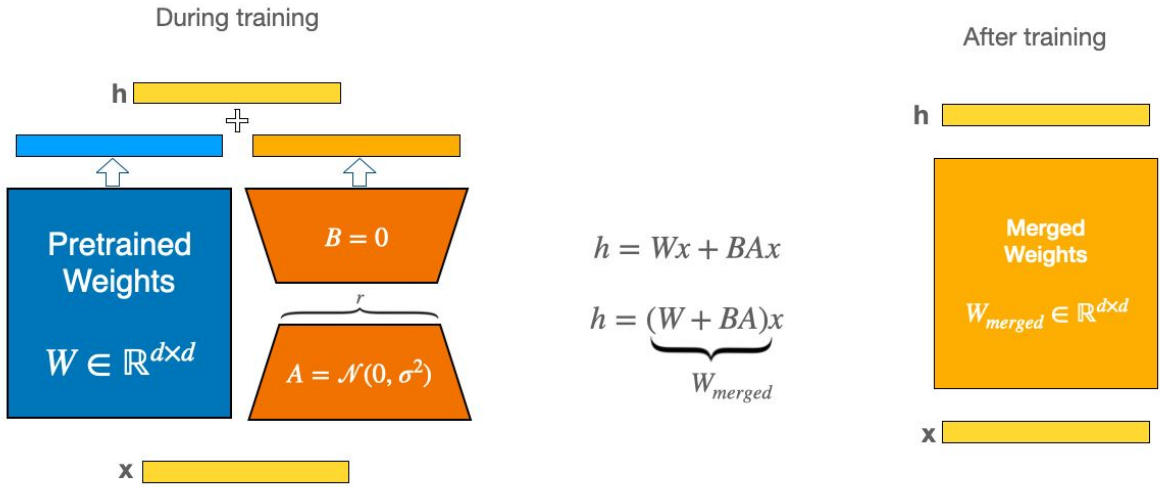
SmoothGrad

- Reduces noise in gradient-based saliency maps by averaging over many noisy samples of the input. Given n noisy copies x^k = x + Normal, the final attribution is calculated by the formula
- By adding Gaussian noise to embeddings or to the raw input, and then averaging the resulting gradient maps, SmoothGrad highlights consistent attribution patterns and smooths out unexpected peaks.

$$\mathrm{SmoothGrad}(x) = \frac{1}{n} \sum_{k=1}^{n} \frac{\partial F\left(x^{(k)}\right)}{\partial x}$$

Low-Rank Adaptation (LoRA)

- LoRA is a common PEFT method for fine-tune large models cheaply by adding and learning only low-rank adapters.
- LoRA keeps the original pretrained weight matrix W frozen and learns two small low-rank matrices A and B.
- During fine-tuning, only A and B are updated, drastically reducing the number of trainable parameters, and at inference time the low-rank update is merged into W for standard usage.

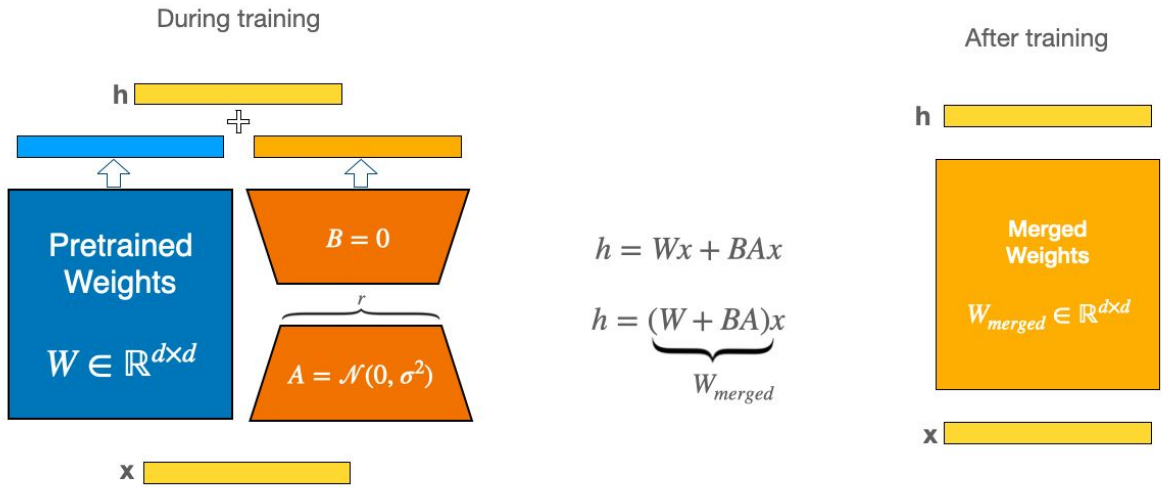


Low-Rank Adaptation (LoRA)

- LoRA makes fine-tuning large pretrained models faster and lighter by keeping the original weight matrix frozen and learning only a small, low-rank update.
- Express the update ΔW to the frozen weight matrix W as the product of two much smaller matrices A B.
- A and B capture the task-specific changes without touching W itself.
- Then form the adapted weight matrix W ′ by adding the update: W ′ = W + ΔW. W stays exactly as it was pretrained, and ΔW carries all the fine-tuning.
- When the layer sees an input x ∈ R^k, it computes

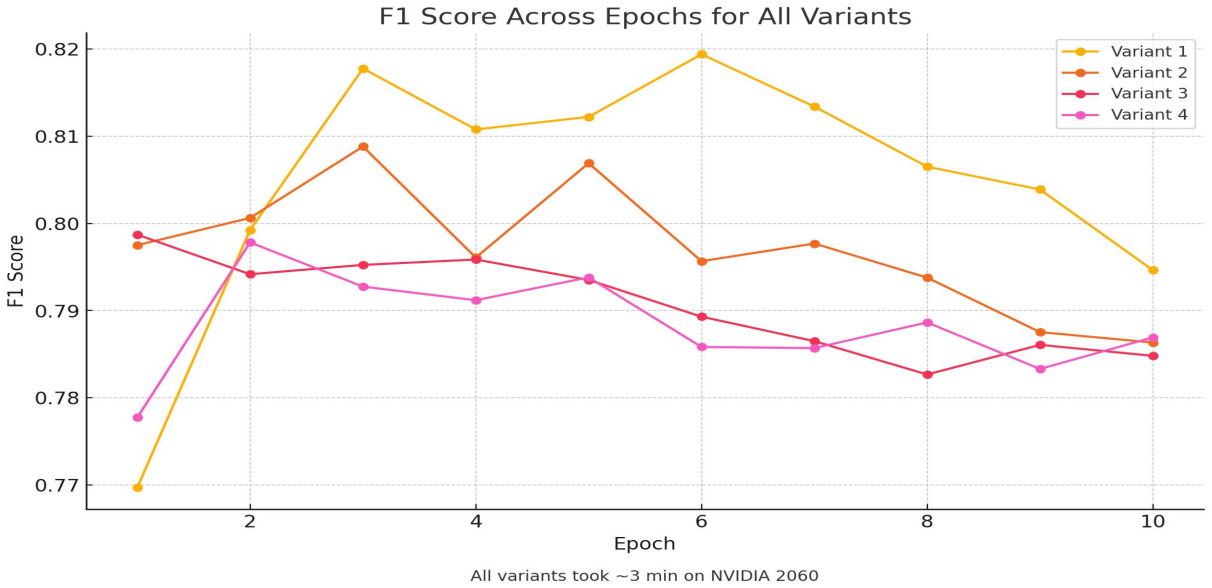
  y = W ′ x = W x + A (Bx).

- Since r (rank A and B) is much smaller than d and k, training costs drop from O(dk) to O((d + k)*r)



Standard fine-tuning

- The data from GUE was stored as follows: sequences of length 70, labeled 1 if the sequence is a promoter and 0 otherwise, so it is a sequence classification task.
- As a result, the best performance ended up being the settings of the first experiment. F1 = 0.82



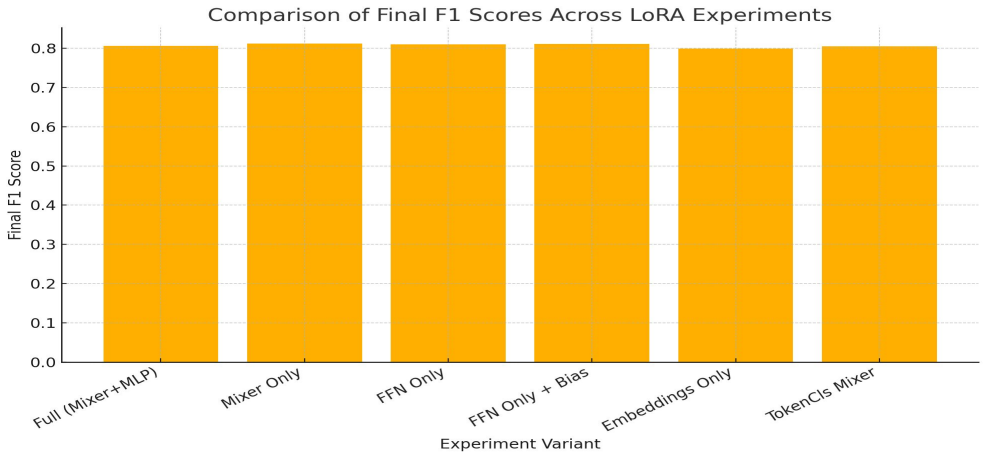
Metric	Value
Validation loss	0.4013
Accuracy	0.8167
Precision	0.8024
Recall	0.8416
F1 score	0.8216
MSS	0.6342
Evaluation runtime (s)	0.8965
Samples per second	6603.3390
Steps per second	103.7350
Epoch	10.0

Experiment	Learning Rate	Train / Eval Batch Size	LR Scheduler
1	6×10^{-4}	64 / 64	linear
2	5×10^{-4}	32 / 32	linear
3	2×10^{-5}	64 / 64	cosine
4	5×10^{-4}	8 / 8	linear

Hyperparameter settings for the four HyenaDNA training runs. All experiments used 10 epochs, warmup ratio 0.1, weight decay 0.01, the adamw_torch optimizer, and FP16 precision

LoRA

- As a result, LoRA did not bring any advantage in the task of promoters on the HyenaDNA model, time required is the same and F1 = 0.81



Experiment	r	α	Dropout	% Trainable
Full (Mixer+MLP)	8	32	0.1	2.79%
Mixer Only	8	32	0.1	7.04%
FFN Only	8	32	0.1	5.38%
FFN Only + All Bias	16	16	0.1	9.54%
Embeddings Only	16	32	0.1	0.58%
TokenCls (Mixer)	16	32	0.1	6.33%

LoRA configuration for each experiment: rank r, α, dropout, and percentage of trainable parameters.

Interpretation

Algorithm 2 SequenceClassificationExplainer Attribution Aggregation

```
procedure ExtractTopKmers(model, tokenizer, dataset, k)
    TP ← []                                              ▷ true positives
    for all example in dataset.dev do
        seq ← example.sequence
        inputs ← tokenizer(seq) on device
        logits ← model(inputs).logits
        pred, true ← arg max(logits), example.label
        if pred = 1 and true = 1 then
            TP.append(seq)
        end if
    end for
    scores ←                                             ▷ map k-mer to list of scores
    for all seq in TP do
        attrs ← explainer(seq, n_steps = 50)
        (tokens, vals) ← filterSpecials(attrs)
        for i = 1 to |tokens| − k + 1 do
            kmer ← tokens[i : i + k] as string
            s ← mean(vals[i : i + k])
            scores[kmer].append(s)
        end for
    end for
    avgScores ← (kmer, mean(scores[kmer]))
    sorted ← sort_desc(avgScores)
    return sorted
end procedure
```

Algorithm 3 SmoothGrad-Based K-mer Attribution

```
procedure RankKmersSmoothGrad(model, tokenizer, dataset, k)
    move model to GPU and set to eval mode
    TP ← ∅                                               ▷ collect true positives
    for all example in dataset.dev do
        seq ← example.sequence
        enc ← tokenizer(seq) on device
        pred ← arg max(model(enc).logits)
        if pred = 1 and example.label= 1 then
            add seq to TP
        end if
    end for
    scores ← {}                                          ▷ map each k-mer to list of attributions
    for all seq ∈ TP do
        enc ← tokenizer(seq) on device
        ids ← enc.input_ids, mask ← enc.attention_mask
        pred ← arg max(model(enc).logits)
        E ← embedding_layer(ids)
        B ← 0 tensor shaped like E
        A ← NoiseTunnel(IntegratedGradients)(E, B, target = pred, nt_samples =
    50, stdevs = 0.02, args = (mask, ))
        attr ← ∑A over embedding dim
        tokens ← tokenizer.convert_ids_to_tokens(ids)
        remove special tokens from tokens and attr
        for i = 1 to |tokens| − k + 1 do
            kmer ← join(tokens[i : i + k])
            s ← mean(attr[i : i + k])
            append s to scores[kmer]
        end for
    end for
    avgScores ← {(k, m = mean(scores[k]))}
    return sorted avgScores in descending order
end procedure
```

Interpretation

- For interpretation I used weights of the model with the best F1 (0.82)
- In summary, the two methods of interpretation and their ranking showed that G and C combinations are in the top, which is consistent with the biological promoter theory, and most importantly shows that the model was able to identify the correct dependencies.

k-mer	SmoothGrad Attribution
GCTGG	0.09124
CTGGG	0.08750
GGTGG	0.08517
CGCGG	0.08249
TGGGG	0.08248
GCGGG	0.08084
GTTGG	0.08069
TGGGA	0.08046
GCGCG	0.08040
GGAAG	0.08014

Top 10 5-mers by average SmoothGrad attribution score

k-mer	SeqClassExplainer Attribution
GGCTG	0.06518
GCTGG	0.06350
GCTGC	0.06171
TTTCC	0.06139
CTGCT	0.06096
CTTCC	0.05841
TTCCG	0.05808
TTCCT	0.05740
TGCTG	0.05660
GGGAG	0.05548

Top 10 5-mers by average IG attribution score

k-mer	Mean Dev (%)
GCTGG	138.50
GGCTG	125.49
GCTGC	110.13
CTGGG	106.59
GGGAG	104.54
TGCTG	103.48
GGTGG	95.47
TGGGG	88.36
TGGGA	87.88
GGAGG	86.42

Consensus ranking of top 5-mers by average percent deviation from mean attributions across Integrated Gradients and SmoothGrad

Model's Details

- Positive examples are one-quarter as frequent as negatives, so all train/val/test splits use stratified sampling to maintain the 1:3 class ratio.
- The pretrained HyenaDNA model lacks a token-classification head, so a new class, `HyenaDNAForTokenClassification`, was created extending `HyenaDNAPreTrainedModel`
- Read `num_labels` from either keyword arguments or configuration.
- Instantiate the HyenaDNA backbone via `HyenaDNAModel(config)`. Attach a linear classifier of shape (d_model × num_labels) without bias.
- Set up cross-entropy loss with class weights (tuned experimentally) and `ignore_index = -100` to skip special tokens.
- Methods `get_input_embeddings` / `set_input_embeddings` expose the embedding layer for external modifications.
- Returns a `TokenClassifierOutput` containing `loss`, `logits`, and optional hidden states, or a `(loss, logits)` tuple if `return_dict=False`

Algorithm 1 HyenaDNA Token-Level Classification

Init(config, num_labels)
  $L \leftarrow$ HyenaDNAModel(config)
  $C \leftarrow$ Linear(d_model, num_labels)
  $\mathcal{L} \leftarrow$ CrossEntropyLoss(weights, ignore_index)
  post_init()
procedure FORWARD(input_ids, inputs_embeds, labels, return_dict)
  $O \leftarrow L$(input_ids, inputs_embeds, return_dict)
  $H \leftarrow O$.last_hidden_state
  logits $\leftarrow C(H)$
  if labels provided **then**
    $\ell \leftarrow \mathcal{L}$(logits.*reshape*$(-1, \_)$, labels.*reshape*$(-1)$)
  else
    $\ell \leftarrow$ None
  end if
  if return_dict **then**
    return {loss=ℓ, logits, hidden_states $= O$.hidden_states} **else**
    return loss? $(\ell$, logits$)$: (logits) **end if**

Details

- Positive examples are one-quarter as frequent as negatives, so all train/val/test splits use stratified sampling to maintain the 1:3 class ratio.
- The pretrained HyenaDNA model lacks a token-classification head, so a new class, `HyenaDNAForTokenClassification`, was created extending `HyenaDNAPreTrainedModel`

Algorithm 1 HyenaDNA Token-Level Classification

Init(config, num_labels)
  $L \leftarrow$ HyenaDNAModel(config)
  $C \leftarrow$ Linear(d_model, num_labels)
  $\mathcal{L} \leftarrow$ CrossEntropyLoss(weights, ignore_index)
  post_init()
procedure FORWARD(input_ids, inputs_embeds, labels, return_dict)
  $O \leftarrow L$(input_ids, inputs_embeds, return_dict)
  $H \leftarrow O$.last_hidden_state
  logits $\leftarrow C(H)$
  if labels provided **then**
    $\ell \leftarrow \mathcal{L}$(logits.*reshape*$(-1, \_)$, labels.*reshape*$(-1)$)
  else
    $\ell \leftarrow$ None
  end if
  if return_dict **then**
    return {loss=ℓ, logits, hidden_states $= O$.hidden_states} **else**
    return loss? $(\ell,$ logits$)$: (logits) **end if**

Standart Fine-Tuning: Best Approach

- Through extensive Z-DNA experiments, the most effective setup was freezing both the model's head and body, yielding an F1 0.64 , and an MCC 0.62.
- The best performance was obtained with a custom two-phase training regimen under the following settings: class-weighted cross-entropy loss with weights $w- = 0.7$ (negative) and $w+ = 2.0$; sequence length $L = 100$; batch size 64; no data augmentation; standard HyenaDNA tokenization; stratified 60/20/20 train/val/test split.
- Phase 1: Frozen Backbone. Only the classification head was trained for 3 epochs at learning rate $\eta = 10-3$, while all Hyena backbone parameters remained frozen. This allowed the newly-initialized head to specialize rapidly without perturbing pretrained features.
- Phase 2: Full Fine-Tuning. The entire model was unfrozen and trained for 6 further epochs with discriminative learning rates: $\eta backbone = 10-5$ and $\eta head = 5 \times 10-4$. This produced the final F1 score of 0.64 on the evaluation set.

Standart Fine-Tuning: Another Approaches

Configuration	Variants / Settings	Purpose / Hypothesis
Sequence length	100, 256, 512, 1024	Effect of context-window size on accuracy
Batch size	8, 16, 32, 64, 128	Throughput vs. generalization trade-off
Loss function	Weighted CrossEntropy, Focal Loss	Handling class imbalance
Data augmentation	Random cropping (±20), overlapping windows	Robustness to fragment positioning
Reverse-complement augm.	On-the-fly reverse-complement on 50% of batches	Enforce strand-invariance
Freezing strategy	Freeze backbone → train head only, then full fine-tune	Stability of pre-trained features
Model head variant	Custom linear vs. Electra-small [5]	Impact of classification head capacity
Learning rate schedule	Linear warmup, cosine decay, constant + ReduceLROnPlateau	Optimization stability and convergence speed
Tokenization setup	With vs. without special tokens	Influence of token markers on sequence understanding
Data split	Stratified split vs. Stratified K-Fold cross-validation	Variance and bias in validation metrics
Low-batch-size regime	batch_size=1, accum_steps=4, bf16, checkpointing, 10% warmup	Simulate low-memory training with effective batch size=4

Experiment	F1	ROC-AUC	MCC	Precision	Recall
Batch=1, accum=4, 5ep, Len=100	0.6283	0.7836	0.6136	0.6846	0.5805
Batch=128, 10ep, Len=100, CE(0.7/2)	0.6378	0.8021	0.6205	0.6562	0.6204
Batch=64, 12ep, Len=100, constant + warmup	0.6319	0.8055	0.6135	0.6347	0.6291
Batch=8, 12ep, Len=100	0.6385	0.8046	0.6209	0.6516	0.6259
5-fold CV, 12ep, Len=100, Batch=64	0.6378	0.8012	0.6207	0.6586	0.6186
Augmented data (80–120bp), 12ep, Batch=64	0.6437	0.7973	0.6282	0.6829	0.6086

Comparison of evaluation metrics across various training configurations near the optimal solution

LoRA

- The LoRa situation on the Z-DNA task is very similar to the outcome of the promoters'task.
- LoRa once again did not provide any significant speedup, and its performance metrics were almost the same as with standard fine-tuning

Variant	Trainable %	F1	ROC-AUC	MCC
Full (Mixer + FFN), 3ep	3.86%	0.6363	0.8057	0.6184
Full (Mixer + FFN), 10ep	3.86%	0.6181	0.8014	0.5989
Mixer only (in/out projections), 10ep	1.49%	0.6291	0.8049	0.6106
FFN only (fc1/fc2), 10ep	4.54%	0.6257	0.8022	0.6071
Short-filter only, 10ep	5.38%	0.6279	0.8015	0.6096
FFN + all biases, 10ep	9.54%	0.6301	0.8036	0.6118
Embeddings only, 10ep	0.58%	0.5830	0.7831	0.5620

Interpretation

Algorithm 4 HyenaDNA Token-Level Z-DNA k-mer Extraction IG

procedure ExtractZDNAKmers(model, dataset, K)
  Initialize empty maps *kmerCounts* and *kmerScoreSums*
  for all sample in dataset **do**
    $seq \leftarrow$ sample.sequence
    $ids,\ mask \leftarrow$ sample.input_ids, sample.attention_mask
    $labels \leftarrow$ sample.labels
    $embeds \leftarrow$ model.get_input_embeddings()(ids)
    $logits \leftarrow$ model($embeds,\ mask$).logits
    $preds \leftarrow \arg\max(logits,\ \text{axis} = -1)$
    $tpMask \leftarrow (preds = 1) \wedge (labels = 1)$ ▷ float mask
    if $\sum tpMask = 0$ **then**
      continue
    end if
    Define $forwardTP(e, m) = \sum(\text{model}(e, m).\texttt{logits}[..., 1] \times tpMask)$
    ig.forward_func $\leftarrow forwardTP$
    $(attrs, \delta) \leftarrow$ IntegratedGradients.attribute($e = embeds$, baselines = 0, additional_forward_args = $(mask)$, $n\_steps = 50$, return_convergence_delta = $True$)
    $scores \leftarrow \sum(attrs,\ \text{axis} = -1)$
    for $i = 1$ **to** $|seq| - K + 1$ **do**
      if all $labels[i : i + K] = 1$ and $preds[i : i + K] = 1$ **then**
        $kmer \leftarrow seq[i : i + K]$
        $kmerCounts[kmer]$ += 1
        $kmerScoreSums[kmer]$ += $\sum(scores[i : i + K])$
      end if
    end for
  end for
  Build table of $\{kmer,\ Count,\ AvgScore = \frac{kmerScoreSums[kmer]}{kmerCounts[kmer]}\}$
  return table sorted by *AvgScore* descending
end procedure

Algorithm 5 SmoothGrad-Saliency Z-DNA k-mer Extraction

procedure ExtractZDNAKmersSmoothGrad(model, dataset, K)
  Initialize empty maps *kmerCounts* and *kmerScoreSums*
  for all sample in dataset **do**
    $seq \leftarrow$ sample.sequence
    $ids,\ mask \leftarrow$ sample.input_ids, sample.attention_mask
    $labels \leftarrow$ sample.labels
    $embeds \leftarrow$ model.get_input_embeddings()(ids)
    $logits \leftarrow$ model($embeds,\ mask$).logits
    $preds \leftarrow \arg\max(logits,\ \text{axis} = -1)$
    $tpMask \leftarrow (preds = 1) \wedge (labels = 1)$ ▷ float mask
    if $\sum tpMask = 0$ **then**
      continue
    end if
    Define $forwardTP(e, m) = \sum(\text{model}(e, m).\texttt{logits}[..., 1] \times tpMask)$
    saliency.forward_func $\leftarrow forwardTP$
    nt $\leftarrow$ NoiseTunnel(saliency)
    $attrs \leftarrow$ nt.attribute(inputs = $embeds$, nt_type = 'smoothgrad', nt_samples = 50, nt_samples_batch_size = 10, stdevs = 0.1, additional_forward_args = $(mask)$)
    $scores \leftarrow \sum(attrs,\ \text{axis} = -1)$
    for $i = 1$ **to** $|seq| - K + 1$ **do**
      if all $labels[i : i + K] = 1$ and $preds[i : i + K] = 1$ **then**
        $kmer \leftarrow seq[i : i + K]$
        $kmerCounts[kmer]$ += 1
        $kmerScoreSums[kmer]$ += $\sum(scores[i : i + K])$
      end if
    end for
  end for
  Build table of $\{kmer,\ Count,\ AvgScore = \frac{kmerScoreSums[kmer]}{kmerCounts[kmer]}\}$
  return table sorted by *AvgScore* descending
end procedure

Interpretation

- Count k-mer True-Positive Importance algorithm is not a gradient-based interpretation method, but rather an idea that can be used to augment complete solutions.
- To get a quick, model-agnostic importance score, I simply computed, for each token, the fractionTP occurrences / Total occurrences over the set (also TP only).
- Note that “TP / Total” score only captures tokens inside annotated Z-DNA regions

Algorithm 6 Count k-mer True-Positive Importance

```
procedure ComputeKmerImportance(model, dataset, K)
    Initialize empty maps tpCounts and occCounts
    for all sample index in 0, ..., |dataset| − 1 do
        seq ← dataset[index].seq
        labels ← dataset[index].labels
        Prepare inputs, mask from sample and move to device
        with no_grad(): logits ← model(inputs, mask).logits
        preds ← arg max(logits, axis = −1)
        L ← |seq|
        for i = 0 to L − K do
            kmer ← seq[i : i + K]
            occCounts[kmer] += 1
            windowLabels ← labels[i : i + K]
            windowPreds ← preds[i : i + K]
            if all entries of windowLabels and windowPreds equal 1 then
                tpCounts[kmer] += 1
            end if
        end for
    end for
    Initialize empty list records
    for all (kmer, total) in occCounts do
        tp ← tpCounts.get(kmer, 0)
        importance ← tp/total
        Append to records {kmer, tp, total, importance}
    end for
    return records sorted by importance descending
end procedure
```

Faculty of Computer Science

Interpretation

- For interpretation I used weights of the model with the best F1 (0.64)
- In summary, interpretation analyses of IG, SmoothGrad and TP k-mer counting have all independently identified GC-rich 5-mers such as CGCGC, GCGCG and CGCGT as the most influential motifs for Z-DNA prediction. This result is consistent with the biological theory of Z-DNA, which states that the alternation of such nucleotides occurs in regions of the left-handed helix.

5-mer	TP Count	Occurrences	Importance
CGCAC	844	2895	0.2915
GTGCG	769	2992	0.2570
CGCGT	427	1676	0.2548
GCGCA	756	3006	0.2515
CGTGT	532	2140	0.2486
CGCGG	1069	4390	0.2435
GCGCG	1622	6925	0.2342
CGCGC	1597	6897	0.2315
CGTGC	604	2612	0.2312
GCACG	598	2669	0.2241

Top 10 5-mers ranked by True-Positive importance: the fraction of times each 5-mer was predicted correctly among its total occurrences.

5-mer	Count	Avg SmoothGrad Score
ATACG	60	534.45
GCGCG	1622	529.61
CGCGC	1597	523.68
TACGT	61	456.88
ACGCG	200	454.00
CGCGT	427	451.16
ATGTA	38	420.26
GCGTG	763	419.04
CGTAT	48	413.98
GTGCG	769	412.44

Top 10 5-mers by average SmoothGrad attribution score

5-mer	Count	Avg IG Score
CGCGC	1597	23.67
GCGCG	1622	21.97
CGTGC	604	21.69
CGCGT	427	21.60
CGTGT	532	20.11
CGCAC	844	19.50
GCGTG	763	19.42
AATGC	1	17.07
GTGCG	769	16.96
AAGAC	1	16.88

Top 10 5-mers by average IG attribution score

5-mer	Mean Dev. (%)
CGCGC	542.54
GCGCG	511.67
CGCGT	477.10
CGTGC	461.29
CGTGT	426.80
GCGTG	423.48
CGCAC	417.34
ACGCG	380.64
GTGCG	373.61
GCGCA	351.54

Consensus ranking of top 5-mers by average percent deviation from mean attributions across Integrated Gradients and SmoothGrad

Z-DNA conclusion

- Outperformed DeepZ and GraphZ even without using omics features.
- Did not exceed Z-DNABERT's F1 score of 0.83, confirming that Transformer models still lead in raw accuracy.
- HyenaDNA trains dramatically faster than Transformer-based approaches
- Uses strict token-level labeling for Z-DNA regions, offering a more rigorous benchmark than previous studies.
- Despite lower overall accuracy, the top-ranked 5-mer from attribution matches known biologically validated motifs, showing the model has learned meaningful signals.

Parameter	Value
Loss weighting	CE (negative: 0.7, positive: 2)
Stage 1 epochs	3
Stage 1 learning rate	1×10^{-3}
Stage 1 backbone	frozen
Stage 2 epochs	6
Stage 2 learning rate (backbone)	1×10^{-5}
Stage 2 learning rate (head)	5×10^{-4}
Evaluation loss	0.1525
F1 score	0.6401
ROC–AUC	0.8046
MCC	0.6226
Precision	0.6550
Recall	0.6257

Two-stage fine-tuning setup and resulting test metrics for the best model.

Standard fine-tuning

- After running numerous experiments, it became clear that the model, regardless of hyperparameter settings, quickly reaches a performance plateau indicating its limited capacity for further improvement on this task.
- The best result, however, was achieved with standard training using a batch size of 32, sequence length of 100, 10 percent warmup, class-weighted cross-entropy loss (CE = 1 and 8), and 12 epochs, yielding an F1 0.5788.

Experiment	CE	Batch	Ep.	Scheduler	F1	ROC-AUC
Two-stage	0.7/2	64	3 | 6	Linear warmup	0.5779	**0.8206**
Single-stage	1/8	64	12	ReduceLROnPlateau	0.5772	0.8178
Single-stage	1/8	**32**	12	Linear warmup	**0.5788**	0.8177
Single-stage	0.7/2	64	12	Linear warmup	0.5761	0.8166

Comparison of G4-classification setups.

LoRA and results for G4

- In the end LoRa showed the same trend as in the previous experiments, no significant code acceleration, but metrics almost like a regular fine-tune. LoRa once again proved unnecessary for the HyenaDNA architecture.
- G4 prediction proved to be the weakest link in this study, with metrics too low to warrant deeper interpretation. By comparison, a recent benchmark of HyenaDNA achieved an F1 0.75 roughly 20 points higher when training from scratch.
- Although differences in pre-training strategies make direct comparison unfair, it is still instructive to note that Transformer-based models adapt far more effectively to G4 classification.
- In other words, for DNA G4 secondary-structure prediction, HyenaDNA falls short of its alternatives.

Experiment	Trainable %	Epochs	F1	ROC-AUC	MCC	Prec.	Rec.
Full (Mixer + MLP)	7.04	10	**0.5767**	**0.8242**	**0.5438**	**0.4824**	**0.7168**
Mixer only	2.79	10	0.5756	0.8237	0.5426	0.4811	0.7162

Comparison of LoRA-finetuned HyenaDNA variants on the G4 token-classification task. The "Full" variant adapts both mixer and MLP layers, while "Mixer only" adapts mixer layers alone. Metrics taken at the final epoch (10). Best values in bold.

HyenaDNA results

- HyenaDNA, fine-tuned on promoter data, identified promoter segments.
- Achieved good performance across all promoter categories with significantly lower training time.
- Outperformed both DeepZ and GraphZ in quality metrics, despite using only sequence data (no omics features).
- Did not surpass Z-DNABERT's F1 of 0.83, but training and inference were substantially faster.
- Attribution analyses ranked the known 5-mer motif highest, confirming correct biological signal learning.
- HyenaDNA's fine-tuned models for G4 scored lowest among the three tasks
- Across tasks, HyenaDNA often approached good results while reducing training time dramatically comparing to transformers.
- Demonstrated strong signal localization (especially in Z-DNA), even in cases of lower raw accuracy.

Task	Accuracy	F1	MCC
Promoter	0.8167	0.8216	0.6342
Z-DNA	0.9664	0.6401	0.6226
G4	0.9172	0.5788	0.5443

Resulting best metrics for each task

Goals

- Continue exploring HyenaDNA's architecture beyond classical fine-tuning, including training from scratch and integrating multi-omics data.
- Continue work with the International Laboratory of Bioinformatics to refine LLM approaches for DNA secondary-structure prediction.
- Expand pretraining objectives and data regimes to better tailor HyenaDNA to genomics-specific patterns.
- Aim to publish refined methodologies and benchmark results in a top-tier bioinformatics journal.


HSE UNIVERSITY
EST 1992

References

1. Pierce, Benjamin A. *Genetics: A Conceptual Approach*. W. H. Freeman, 2017. ISBN 978-1-319-72085-3.
2. Bartas, Martin; Slychko, Kristyna; Cerven, Jiri; Pecinka, Petr; Arndt-Jovin, Donna J.; Jovin, Thomas M. “Extensive Bioinformatics Analyses Reveal a Phylogenetically Conserved Winged Helix (WH) Domain (Zτ) of Topoisomerase IIα, Elucidating Its Very High Affinity for Left-Handed Z-DNA and Suggesting Novel Putative Functions.” *International Journal of Molecular Sciences*, 2023.
3. Cherednichenko, O.; Herbert, A.; Poptsova, M. “Benchmarking DNA Large Language Models on Quadruplexes.” *Computational and Structural Biotechnology Journal*, Vol. 27, 2025, pp. 992–1000. doi:10.1016/j.csbj.2025.03.007.
4. Clark, Kevin; Luong, Minh-Thang; Le, Quoc V.; Manning, Christopher D. “ELECTRA: Pre-training Text Encoders as Discriminators Rather Than Generators.” *arXiv preprint arXiv:2003.10555*, 2020. Proceedings of ICLR 2020. doi:10.48550/arXiv.2003.10555.
5. Fishman, Veniamin; Kuratov, Yuri; Shmelev, Aleksei; Petrov, Maxim; Penzar, Dmitry; Shepelin, Denis; Chekanov, Nikolay; Kardymon, Olga; Burtsev, Mikhail. “GENA-LM: A Family of Open-Source Foundational DNA Language Models for Long Sequences.” *Nucleic Acids Research*, 2025.
   Gu, Albert; Goel, Karan; Ré, Christopher. “Efficiently Modeling Long Sequences with Structured State Spaces.” *arXiv preprint arXiv:2111.00396*, 2021.
6. Hu, Edward J.; Shen, Yelong; Wallis, Phillip; Allen-Zhu, Zeyuan; Li, Yuanzhi; Wang, Shean; Wang, Lu; Chen, Weizhu. “LoRA: Low-Rank Adaptation of Large Language Models.” *arXiv preprint arXiv:2106.09685*, 2021. https://doi.org/10.48550/arXiv.2106.09685.
7. Kouzine, F.; Wojtowicz, D.; Baranello, L.; Yamane, A.; Nelson, S.; Resch, W.; Kieffer-Kwon, K.-R.; Benham, C. J.; Casellas, R.; Przytycka, T. M.; Levens, D. “Permanganate/S1 Nuclease Footprinting Reveals Non-B DNA Structures with Regulatory Potential Across a Mammalian Genome.” *Cell Systems*, Vol. 4, No. 3, 2017.
8. Beknazarov, Nazar; Konovalov, Dmitrii; Herbert, Alan; Poptsova, Maria. “Z-DNA Formation in Promoters Conserved Between Human and Mouse Are Associated with Increased Transcription Reinitiation Rates.” *Scientific Reports*, Vol. 14, 2024, Article 17786.
10. Beknazarov, Nazar; Jin, Sheng; Poptsova, Maria. “Deep Learning Approach for Predicting Functional Z-DNA Regions Using Omics Data.” *Scientific Reports*, vol. 10, Article 19134, 2020. https://doi.org/10.1038/s41598-020-76203-1.
11. Kosiol, Natalie; Juranek, Silke; Brossart, Peter; Heine, Andreas; Paeschke, Klaus. “G-Quadruplexes: A Promising Target for Cancer Therapy.” *Molecular Cancer*, vol. 20, no. 1, Article 40, 25 Feb. 2021. https://doi.org/10.1186/s12943-021-01328-4. PMID: 33632214; PMCID: PMC7905668.
12. Nguyen, Eric; Poli, Michael; Faizi, Marjan; Thomas, Armin; Birch-Sykes, Callum; Wornow, Michael; Patel, Aman; Rabideau, Clayton; Massaroli, Stefano; Ermon, Stefano; Bengio, Yoshua; Baccus, Stephen A.; Ré, Christopher. “HyenaDNA: Long-Range Genomic Sequence Modeling at Single Nucleotide Resolution.” *arXiv preprint arXiv:2306.15794 [cs.LG]*, 2023.
13. Poli, Michael; Massaroli, Stefano; Nguyen, Eric; Fu, Daniel Y.; Dao, Tri; Baccus, Stephen; Bengio, Yoshua; Ermon, Stefano; Ré, Christopher. “Hyena Hierarchy: Towards Larger Convolutional Language Models.” In: Krause, Andreas; Brunskill, Emma; Cho, Kyunghyun; Engelhardt, Barbara; Sabato, Sivan; Scarlett, Jonathan, eds. *Proceedings of the 40th International Conference on Machine Learning*. Vol. 202, Proceedings of Machine Learning Research, PMLR, 2023, pp. 28043–28078. https://proceedings.mlr.press/v202/poli23a.html.
14. Schiff, Yair; Kao, Chia-Hsiang; Gokaslan, Aaron; Dao, Tri; Gu, Albert; Kuleshov, Volodymyr. “Caduceus: Bi-Directional Equivariant Long-Range DNA Sequence Modeling.” *arXiv preprint arXiv:2403.03234v2 [q-bio.GN]*, 2024.
15. Smilkov, Daniel; Thorat, Nikhil; Kim, Been; Viégas, Fernanda; Wattenberg, Martin. “SmoothGrad: Removing Noise by Adding Noise.” *arXiv preprint arXiv:1706.03825*, 2017. https://doi.org/10.48550/arXiv.1706.03825.
16. Sundararajan, Mukund; Taly, Ankur; Yan, Qiqi. “Axiomatic Attribution for Deep Networks.” In: *Proceedings of the International Conference on Machine Learning*, PMLR, 2017, pp. 3319–3328.
17. Umerenkov, Dmitry; Herbert, Alan; Konovalov, Dmitrii; Danilova, Anna; Beknazarov, Nazar; Kokh, Vladimir; Fedorov, Aleksandr; Poptsova, Maria. “Z-Flipon Variants Reveal the Many Roles of Z-DNA and Z-RNA in Health and Disease.” *bioRxiv*, 2023. https://doi.org/10.1101/2023.01.12.523822.
18. Varshney, Jonathan; Spiegel, David; Zyner, Kristopher; Tannahill, David; Balasubramanian, Shankar. “The Regulation and Functions of DNA and RNA G-Quadruplexes.” *Nature Reviews Molecular Cell Biology*, vol. 21, pp. 459–474, 2020. https://doi.org/10.1038/s41580-020-0236-x.
19. Vasudevaraju, P.; Bharathi; Garruto, R. M.; Sambamurti, K.; Rao, K. S. J. “Role of DNA Dynamics in Alzheimer’s Disease.” *Brain Research Reviews*, vol. 58, no. 1, pp. 136–148, 2008. https://doi.org/10.1016/j.brainresrev.2008.01.001.

References

20. Vaswani, Ashish; Shazeer, Noam; Parmar, Niki; Uszkoreit, Jakob; Jones, Llion; Gomez, Aidan N.; Kaiser, Łukasz; Polosukhin, Illia. “Attention Is All You Need.” In *Advances in Neural Information Processing Systems*, vol. 30, 2017, pp. 5998–6008.
21. Voytetskiy, Artem; Herbert, Alan; Poptsova, Maria. “Graph Neural Networks for Z-DNA Prediction in Genomes.” *IEEE*, 2022. DOI: 10.1101/2022.08.23.504929.
22. Wang, A. J.; Quigley, G.; Kolpak, F.; et al. “Molecular Structure of a Left-Handed Double Helical DNA Fragment at Atomic Resolution.” *Nature*, vol. 282, 1979, pp. 680–686. https://doi.org/10.1038/282680a0
23. Xu, Yan; Komiyama, Makoto. “G-Quadruplexes in Human Telomere: Structures, Properties, and Applications.” *Molecules*, vol. 29, no. 1, 2024. ISSN 1420-3049. doi: 10.3390/molecules29010174. URL: https://www.mdpi.com/1420-3049/29/1/174
24. Zeng, Rong; Li, Zhi; Li, Jia; Zhang, Qing. “DNA Promoter Task-Oriented Dictionary Mining and Prediction Model Based on Natural Language Technology.” *Scientific Reports*, vol. 15, no. 1, Jan. 2025, p. 153. doi: 10.1038/s41598-024-84105-9
25. Zhang, Ting; Yin, Chaoran; Fedorov, Aleksandr; Qiao, Liangjun; Bao, Hongliang; Beknazarov, Nazar; Wang, Shiyu; Gautam, Avishekh; Williams, Riley M.; Crawford, Jeremy Chase; et al. “Z-DNA Secondary Structure Conforms to a Unique Conformation In Vivo and Affects Transcription and Chromatin Structure.” *Nature*, vol. 606, no. 7914, 2022, pp. 594–602. doi: 10.1038/s41586-022-04753-7
26. Zhou, Zhihan; Ji, Yanrong; Li, Weijian; Dutta, Pratik; Davuluri, Ramana; Liu, Han. “DNABERT-2: Efficient Foundation Model and Benchmark for Multi-Species Genome.” *arXiv preprint arXiv:2306.15006 [q-bio.GN]*, 2023.