Enhancing Genomic Foundation Model Robustness through Iterative Black-Box Adversarial Training

Adversarial Attacks on Genomic Foundation Models

Jeyashree Krishnan Ajay Mandyam Rangarajan

RWTH Aachen University

September 2025

Apart Research CBRN AI Risks Research Sprint

Outline

What are Genomic Foundation Models?

Definition: Large-scale neural networks pre-trained on massive genomic datasets to learn general representations of DNA sequences.

Key Characteristics:

- Pre-trained on millions of genomic sequences
- Learn general patterns in DNA structure and function
- Can be fine-tuned for specific tasks (classification, prediction)
- Transfer learning capabilities across different genomic tasks

Examples:

- DNABERT-2: BERT-based architecture for genomic sequences
- Nucleotide Transformer: Transformer-based model
- DNABERT: Original BERT adaptation for DNA

Why GFMs Matter

Applications:

- Promoter prediction
- Enhancer identification
- Splice site detection
- Transcription factor binding prediction
- Disease variant classification

Clinical Impact:

- Automated genomic analysis
- Personalized medicine
- Drug discovery
- Diagnostic tools

Critical Question

How robust are these models to adversarial attacks in clinical settings?

DNABERT-2: Tokenizer

BPE (Byte Pair Encoding) Tokenization:

- Breaks DNA sequences into meaningful sub-sequences
- Learns frequent k-mer patterns during pre-training
- Handles variable-length sequences efficiently
- Captures biological motifs and patterns

Example:

$$\texttt{ATCGATCG} \rightarrow \texttt{[ATCG] [ATCG]} \text{ or } \texttt{[ATC] [GAT] [CG]}$$

Advantages over character-level:

- Better representation of biological motifs
- Reduced sequence length
- Improved computational efficiency

DNABERT-2: Classification Head

Architecture:

- 12 transformer layers, 768 hidden dimensions
- 117M total parameters
- Frozen encoder (pre-trained representations)
- Trainable classification head (296K parameters)

Training Strategy:

- Pre-trained on large-scale genomic data
- Fine-tune only classification head for specific tasks
- Preserves learned genomic representations
- Efficient transfer learning approach

For Binary Classification:

$$\texttt{[CLS] token} \rightarrow \texttt{Linear layer} \rightarrow \texttt{Sigmoid} \rightarrow \texttt{Probability}$$

Targeted vs Untargeted Attacks

Untargeted Attack:

- Goal: Change model prediction to any incorrect class
- Example: Class A $\rightarrow$ Not Class A
- Easier to achieve, less specific

Targeted Attack:

- Goal: Change model prediction to a specific incorrect class
- Example: Class A $\rightarrow$ Specific Class B
- Harder to achieve, more specific

Binary Classification Special Case

For binary classification: **Targeted = Untargeted**

- Only two classes: Class A and Class B
- Changing from A to B is both targeted and untargeted
- No distinction between attack types

Universal Adversarial Attacks

Definition: Single perturbation that fools the model on multiple inputs

Example:

- Find nucleotide substitution pattern
- Apply same pattern to different sequences
- Model misclassifies multiple sequences

Threat Level:

- High: One attack affects many samples
- Practical for real-world deployment
- Difficult to defend against

Genomic Context:

- Motif-based universal attacks
- GC content manipulation
- Regulatory element disruption

Nucleotide Substitutions

Method: Replace individual nucleotides (A, T, C, G) in the sequence

Example:

Original:	`ATCGATCGATCG`
Adversarial:	`ATCGTTCGATCG`
Change:	Position 5: A $\rightarrow$ T

Characteristics:

- Minimal changes (2-3 nucleotides)
- High attack success rate
- Biologically plausible
- Easy to implement

Our Results: 20-50% attack success with only 2 average nucleotide edits

Codon Replacements

Method: Replace entire codons (3-nucleotide sequences) while preserving amino acid meaning

Example:

Original:	`ATG` (Methionine)
Adversarial:	`ATA` (Methionine)
Change:	Synonymous codon substitution

Advantages:

- Preserves protein function
- More biologically realistic
- Harder to detect
- Maintains amino acid sequence

Challenge: Requires understanding of genetic code and codon usage

Other Attack Methods

Insertions/Deletions:

- Add or remove nucleotides
- Can cause frameshift mutations
- More dramatic changes

Inversions:

- Reverse sequence segments
- Maintain nucleotide composition
- Disrupt regulatory motifs

Duplications:

- Repeat sequence segments
- Common in genomic evolution
- Can affect gene expression

Our Focus: Nucleotide substitutions for simplicity and effectiveness

Genetic Algorithm for Adversarial Generation

Implementation: Using DEAP (Distributed Evolutionary Algorithms in Python)

Algorithm Components:

- Population size: 50-100 individuals per generation
- Mutation rate: 0.3 (30% of nucleotides can be mutated)
- Crossover rate: 0.9 (90% probability of genetic crossover)
- Max generations: 100-200 generations
- Convergence threshold: 20 generations without improvement

Fitness Function:

$$\texttt{Fitness} = \alpha \cdot \texttt{confidence\_drop} - \beta \cdot \texttt{num\_perturbations} - \gamma \cdot \texttt{biological\_violations}$$

Selection Strategy:

- Tournament selection with size 3
- Elitism: Keep best individuals

Genetic Algorithm Process

Step 1: Initialization

- Create random population of sequences
- Apply small random mutations to original sequence

Step 2: Evaluation

- Test each sequence against target model
- Calculate fitness based on attack success and biological constraints

Step 3: Selection and Reproduction

- Select parents using tournament selection
- Create offspring through crossover and mutation
- Apply biological constraints during mutation

Step 4: Termination

- Stop when attack succeeds or max generations reached
- Return best adversarial example found

Why Biological Plausibility Matters

Clinical Relevance:

- Adversarial examples must be realistic threats
- Unrealistic attacks don't represent real-world risks
- Clinical applications require biologically valid sequences

Research Validity:

- Results must be interpretable by biologists
- Attack methods should reflect natural variation
- Findings should inform real-world security measures

Key Insight

Adversarial examples that violate biological constraints are not meaningful threats in genomic applications.

Biological Constraints

GC Content Preservation:

- Maintain original GC percentage ($\pm 5\%$)
- Affects DNA stability and structure
- Important for regulatory function

Transition Preferences:

- Prefer A$\leftrightarrow$G, C$\leftrightarrow$T mutations
- More common in natural evolution
- Biologically realistic substitution patterns

Motif Preservation:

- Maintain known regulatory motifs
- Preserve transcription factor binding sites
- Keep functional sequence elements

Stop Codon Avoidance:

- Prevent premature stop codons

Promoter Dataset Overview

Dataset: Binary promoter classification task

Size:

- Training: 47,356 sequences
- Validation: 5,920 sequences
- Test: 5,920 sequences

Sequence Properties:

- Length: 300 base pairs
- Classes: Promoter vs Non-promoter
- Includes TATA and non-TATA promoters

Why This Dataset?

- Well-established benchmark
- Binary classification (simplifies attack analysis)
- Clinically relevant (gene regulation)
- Challenging task requiring sequence understanding

Promoter Dataset: Biological Context

Promoters:

- Regulatory DNA sequences
- Control gene expression
- Located upstream of genes
- Contain transcription factor binding sites

Classification Challenge:

- Distinguish functional promoters from random sequences
- Requires understanding of regulatory motifs
- Tests model's ability to recognize biological patterns

Clinical Relevance:

- Disease-associated promoter mutations
- Drug target identification
- Personalized medicine applications

Iterative Adversarial Training Algorithm

Algorithm 1 Iterative Adversarial Training

1: **Input:** Original dataset D_{orig}, number of iterations T
2: Train initial model M_0 on D_{orig}
3: **for** $i = 1$ to T **do**
4: Generate adversarial examples A_i against model M_{i-1}
5: Combine datasets: $D_i = D_{i-1} \cup A_i$
6: Train model M_i on D_i (full dataset + adversarial examples)
7: Save model M_i
8: **end for**
9: **Output:** Robust model M_T

Key Points:

- Train on full dataset + new adversarial examples
- Frozen encoder, trainable classification head
- Iterative improvement of robustness

Training Process Details

Stage 1: Initial Training

- Train on original 47K promoter sequences
- Achieve baseline performance (93.73% accuracy)
- Establish vulnerability baseline

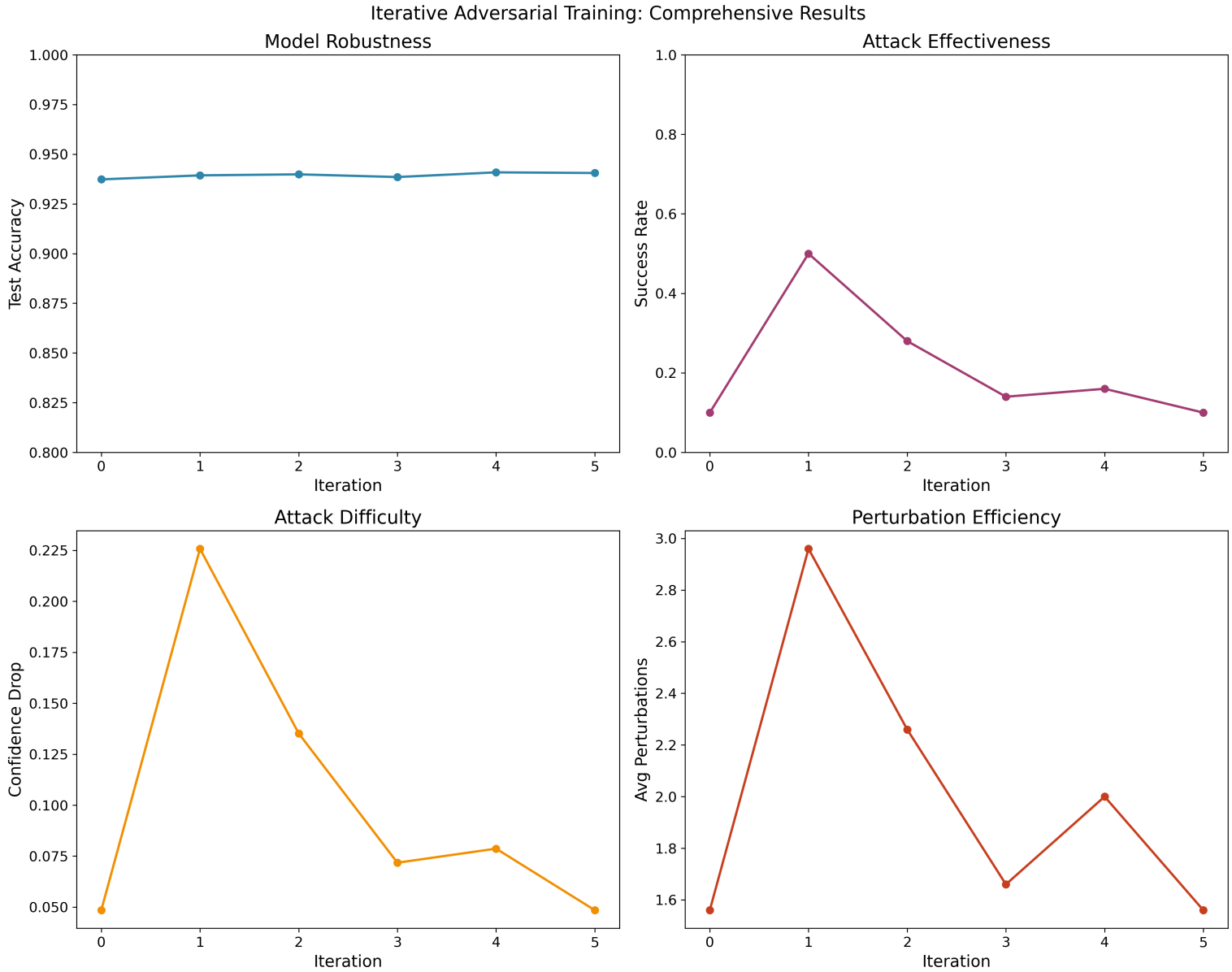
Stage 2: Iterative Adversarial Training

- Generate 25-60 adversarial examples per iteration
- Add to full original dataset (0.13% of training data)
- Retrain classification head on combined dataset
- Repeat for 5 iterations

Expected Pattern:

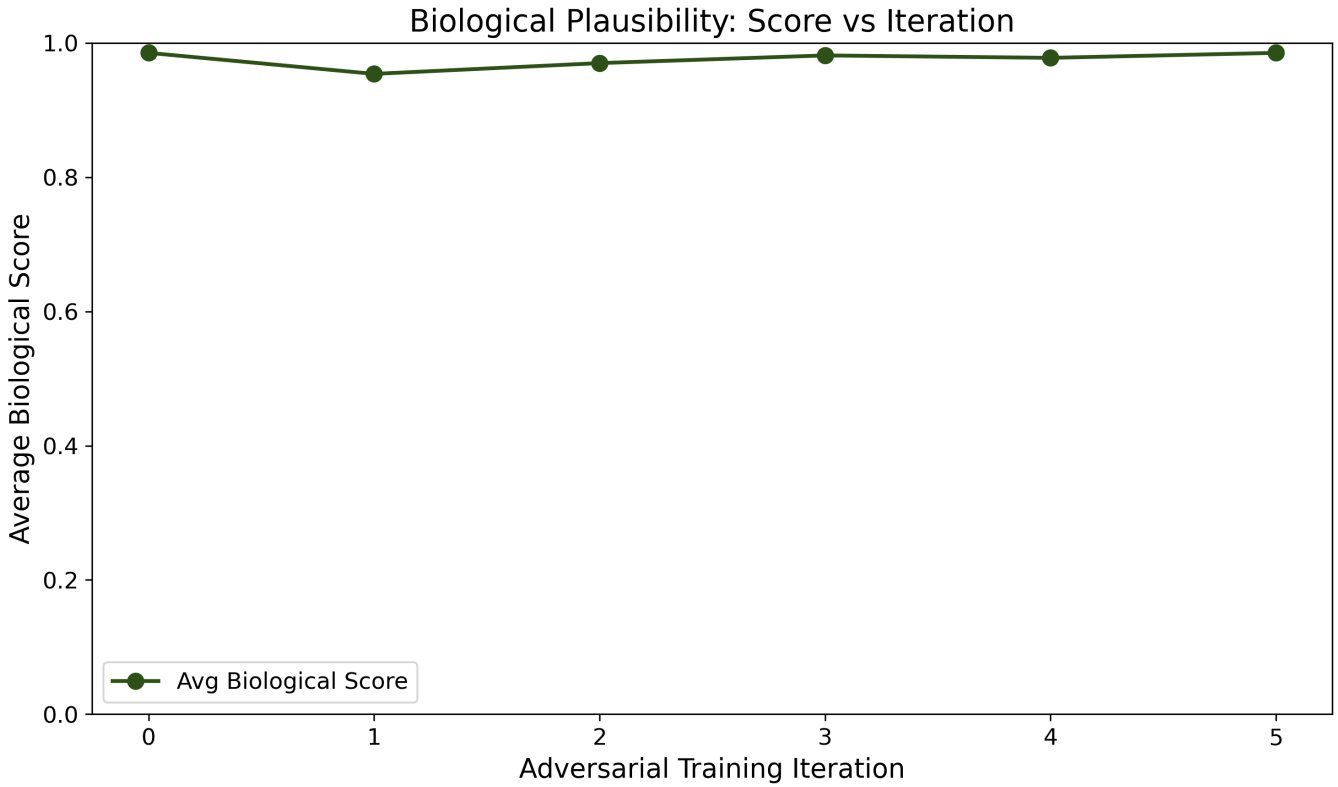
- Iteration 1: Attack success increases (model learns adversarial patterns)
- Iteration 2+: Attack success decreases (model becomes robust)
- Clean accuracy maintained throughout

Comprehensive Results Overview



Key Findings:

Biological Plausibility Results



Maintained Throughout Training:

- GC content preservation
- Regulatory motif conservation
- Transition preference patterns
- Biological sequence properties

Training Dynamics

Iteration 0 (Baseline):

- 93.73% test accuracy
- High vulnerability to attacks
- 2-3 nucleotide edits sufficient

Iteration 1 (First Adversarial Training):

- Attack success rate increases
- Model learns specific adversarial patterns
- 94.09% accuracy with 25 adversarial examples

Iteration 2+ (Robustness Improvement):

- Attack success rate decreases
- Model generalizes against adversarial patterns
- 94.04% accuracy maintained

Key Achievements

Attack Effectiveness:

- Minimal adversarial examples (0.13%) can threaten model
- Only 2 average nucleotide edits needed
- High attack success rates (20-50%)

Robustness Improvement:

- Iterative training increases attack difficulty
- Clean accuracy preserved at 94%

Biological Validity:

- All constraints maintained throughout training
- Realistic adversarial examples generated
- Clinically relevant attack scenarios

Key Contributions

First Iterative Black-Box Attack Framework:

- Novel approach for genomic foundation models
- Genetic algorithm-based adversarial generation
- Biological constraint integration

Significant Findings:

- DNABERT-2 highly vulnerable to minimal perturbations
- Iterative training improves robustness
- Biological plausibility essential for realistic threats

Clinical Implications:

- Highlights security risks in genomic AI
- Provides pathway for robustness improvement
- Informs safety evaluation protocols

Future Directions

Immediate Extensions:

- Multi-class classification tasks
- Additional genomic foundation models
- Diverse attack methods evaluation

Advanced Attacks:

- Universal adversarial examples
- Transfer attacks across models
- Real-time attack generation

Defense Mechanisms:

- Adversarial training improvements
- Detection methods for adversarial examples
- Robust architecture design

Clinical Applications:

- Safety evaluation protocols

Thank You!

Questions?

Robust Iterative Black-Box Attack on GFMs for Classification Problems

Jeyashree Krishnan - RWTH Aachen University
Ajay Mandyam Rangarajan - RWTH Aachen University Code

available at:
`https://github.com/krishnanj/adversarial_attack_gfm.git`