LLMGEN: Evolving Interpretable Surrogate Programs for Genomics with LLMs

Chandana Rajesh
InstaDeep
c.rajesh@instadeep.com

Wilson Ho
InstaDeep, MIT
w.ho@instadeep.com

Ziqi Tang
InstaDeep
z.tang@instadeep.com

Peter K Koo
Cold Spring Harbor Laboratory
koo@cshl.edu

Thomas Pierrot
InstaDeep
t.pierrot@instadeep.com

Abstract

Deep learning models have achieved state-of-the-art performance in predicting complex regulatory tasks. Yet their black-box nature often limits the discovery of new biological insights. Here we present **LLMGEN**, a framework that leverages large language models (LLMs) and evolutionary algorithms to automate the discovery of compact, human-readable symbolic programs. LLMGEN integrates multiple modalities—including biological sequences, functional readouts, natural language descriptions, and executable Python functions—to bridge complex genomic models with interpretable rules. Inspired by recent program-synthesis frameworks such as FunSearch and AlphaEvolve, LLMGEN adapts LLM-guided program evolution to the genomic domain, with prior-guided seeding using biologically relevant attribution features to improve convergence. Across datasets including CRISPRi screens, STARR-seq enhancer assays, and ATAC-seq chromatin accessibility profiles, LLMGEN evolves concise prediction rules that are competitive with deep learning models, rediscovers known motifs and interactions, and generates testable mechanistic hypotheses. These results demonstrate that LLM-guided program evolution is a flexible, model-agnostic approach for building interpretable genomic predictors, advancing multi-modal foundation models toward trustworthy and transparent AI in the life sciences.

1 Introduction

Deep neural networks (DNNs) have advanced regulatory genomics by enabling accurate prediction of enhancer activity, chromatin accessibility, enhancer–promoter interactions (EPI), and variant effects [11, 20, 14, 5, 3]. Large language models (LLMs) trained on genomic sequences further extend these capabilities to representation learning and generative modeling [9, 10, 22, 28]. Yet despite their predictive power, both DNNs and LLMs remain largely opaque, offering limited insight into the regulatory logic that drives predictions [1, 31, 27, 7]. However, interpretability is essential for genomic models, as we want to use them for generating testable hypotheses about genomic regulations. To move beyond black-box predictions, we introduce **LLMGEN**, a framework that leverages LLMs and evolutionary search to discover compact, executable programs that encode regulatory logic. LLMGEN produces human-readable Python functions that serve as surrogates for DNNs or as standalone predictors of experimental data. Guided by biologically relevant priors (e.g., attribution maps, motif discovery), LLMGEN adapts recent advances in LLM-guided program evolution [23, 19] to the genomic domain, enabling interpretable and biologically grounded models.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: 2nd Workshop on Multi-modal Foundation Models and Large Language Models for Life Sciences.

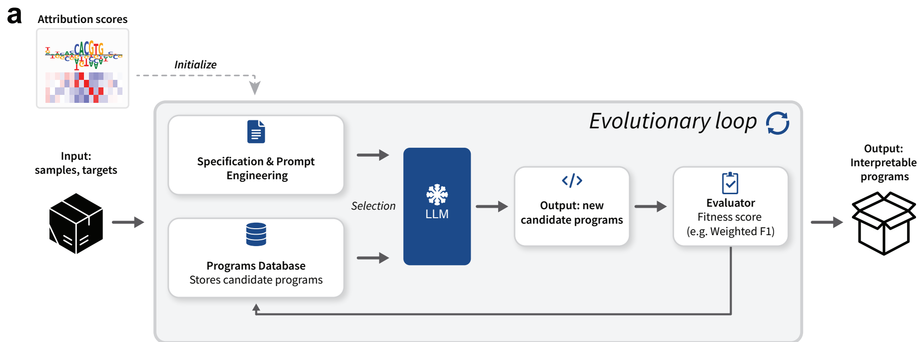

Figure 1: Overview of the LLMGEN pipeline. An LLM (e.g. Gemini [29]) generates/mutates Python programs within an evolutionary loop. Programs are evaluated for fidelity and interpretability (compactness). Prior-guided seeding (e.g., saliency maps, TF-MoDISco motifs, or key genomic features) is critical for search initialization and guidance. Island models (not shown) can manage sub-populations.

2 Related Work

Efforts to interpret genomic deep learning span post-hoc, surrogate, and ante-hoc approaches. Attribution methods such as saliency maps and mutagenesis [26, 33] highlight important positions, with extensions that capture global motif syntax and interactions [15, 4, 30]. Surrogate models (e.g., linear regressions, decision trees) approximate model behavior but often oversimplify [21, 24], while ante-hoc interpretable architectures [16, 31, 18] sacrifice predictive power for transparency. Together, these methods identify what matters but not the complete predictive logic.

Symbolic modeling offers a complementary direction. Symbolic regression and program synthesis can encode motifs, thresholds, and interactions in human-readable form, but historically faced intractable search spaces. Recent frameworks such as FunSearch [23] and AlphaEvolve [19] show that combining LLM-based code generation with evolutionary algorithms enables scalable program discovery. Building on this paradigm, LLMGEN adapts LLM-guided program evolution to regulatory genomics, yielding compact symbolic programs that approximate deep models or directly predict experimental outcomes while remaining interpretable.

3 Methods

3.1 The LLMGEN Evolutionary Pipeline

The LLMGEN pipeline (Fig. 1) is an iterative evolutionary process with four key steps. First, an LLM generates candidate programs (Python functions) by mutating high-performing individuals; mutations may alter motif selection, thresholds, positional weights, or interaction logic, and prompts can be conditioned on priors, previous programs, or objectives. Second, each program is evaluated for predictive performance (e.g., F1 score against DNN outputs or experimental labels) and compactness (e.g., file size, number of operations), with only syntactically valid and executable programs retained. Third, evolutionary selection keeps the best-performing candidates for further mutation, optionally using an island model to preserve population diversity. Finally, prior-guided seeding accelerates convergence by incorporating biologically relevant signals such as high-scoring k-mers from gradient$\times$input maps [26], TF-MoDISco motifs [25], or curated feature sets from [5]. Additional details on the pipeline, including prompt formatting and seeding templates, are provided in Appendix A.

3.2 Datasets, Target DNNs, and Baselines

LLMGEN was evaluated across three representative genomics tasks: (1) Enhancer–Promoter Interaction (Gasperini et al. [13]), predicting significant EPIs from CRISPRi data pre-processed into 532 features [5], where LLMGEN used only 6–8 key features such as H3K27ac, DHS, and distance; (2)

Enhancer Activity (DeepSTARR [11]), predicting enhancer strength in *Drosophila* S2 cells directly from sequence using either experimental activity bins or CNN predictions; and (3) Chromatin Accessibility (ChromBPNet [20]), predicting binned ATAC-seq log-counts in human H1-hESC cells from sequence using either experimental labels or ChromBPNet outputs. For all tasks, baselines included logistic regression on motif counts and human-crafted rules. For EPIs, we additionally compared to Enformer [3] (Appendix C), Activity-by-Contact scores, and a full-feature classifier trained on all 532 features. Evaluation focused on F1 score and model compactness (e.g., program size in KB).

4 Results

4.1 LLMGEN Excels on Enhancer–Promoter Interactions with Minimal Features

We first evaluated LLMGEN on enhancer–promoter interactions using the Gasperini et al. [13] CRISPRi dataset, previously modeled with 532 genomic features [5]. In contrast, LLMGEN was tasked with learning interpretable programs using fewer than 8 inputs from a pre-selected feature set including signals such as H3K27ac, DHS, and genomic distance (Appendix B.1). Despite this drastic reduction in the input feature set, an LLMGEN-evolved program achieved an F1 score of 0.298, close to the full-feature XGBoost model (0.315), competitive with the ABC model (0.301), and better than Enformer (0.270) (Fig. 2a). LLMGEN programs were also exceptionally compact (≈ 3.40 KB) compared to baselines (Fig. 2a). The evolution of model performance over LLMGEN iterations highlights rapid improvements in early iterations (Fig. 2b). Inspection of generated program revealed sparse and readable logic, with rules integrating known activators, repressors (e.g., SUZ12, H3K27me3), and distance effects (Fig. 2c).

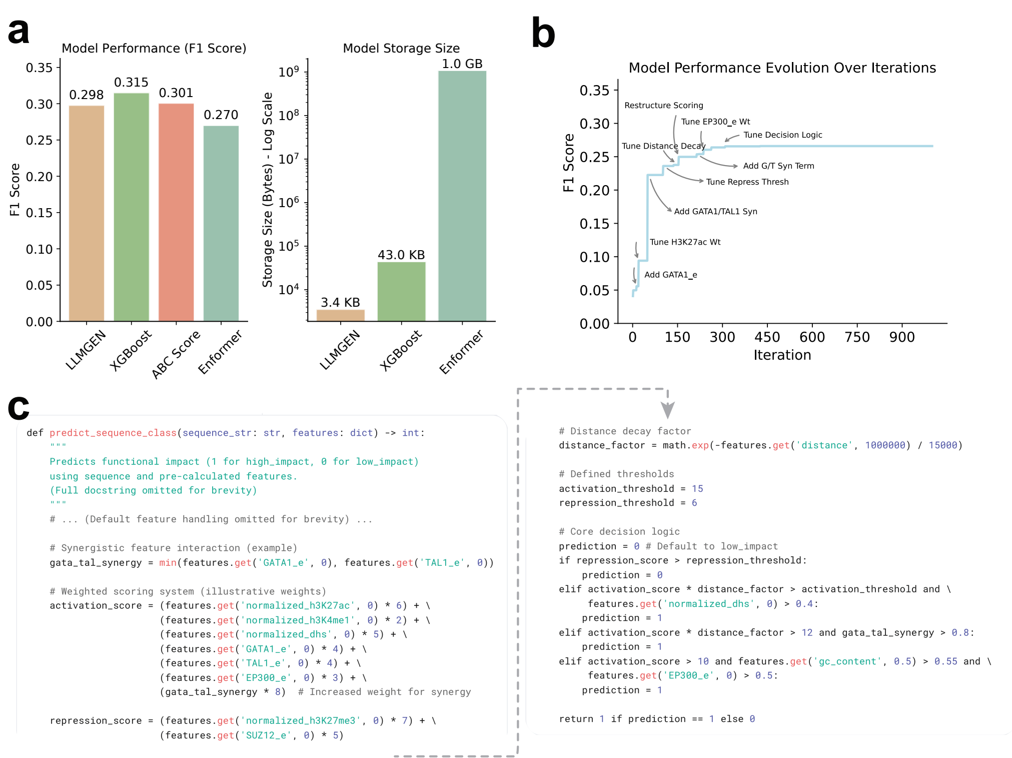

Figure 2: LLMGEN for Enhancer-Promoter (EP) specificity. (a) (Left) F1 scores comparing an LLMGEN program (F1 0.298 <10 features), against prior XGBoost (F1 0.315, >500 features) [5], the ABC model score (F1 0.301) [12], and Enformer (F1 0.270) [3]. (Right) Comparison of model storage sizes. (b) Evolution of F1 score for the LLMGEN program over optimization iterations. (c) Example of an LLMGEN-evolved Python program, illustrating decision logic based on subselected features like H3K27ac, DHS, specific TF activities (GATA1 e, TAL1 e), and distance.

4.2 Compact and Interpretable Programs Across Enhancer and Chromatin Accessibility Tasks

We further evaluated LLMGEN on two more representative sequence-to-function prediction tasks: enhancer activity in *Drosophila* S2 cells (DeepSTARR) and chromatin accessibility in human H1 embryonic stem cells (ChromBPNet). For both settings, we considered three scenarios: (1) direct prediction from experimental data, (2) global surrogate modeling of a deep neural network, and (3) local surrogate modeling from mutagenesis libraries (DeepSTARR only).

LLMGEN consistently outperformed linear and rule-based baselines while approaching CNN performance (Fig.3a). On DeepSTARR, it reached F1 scores of 0.448 ± 0.005 (experimental), 0.530 ± 0.002 (CNN surrogate), and 0.852 ± 0.003 (mutagenesis). On ChromBPNet, it achieved 0.576 ± 0.018 (experimental) and 0.587 ± 0.004 (surrogate). Programs remain highly compact (5–9 KB) compared to CNNs ($\sim$MB–GB scale). Inspection of top-performing programs revealed recovery of key regulatory elements. GATA and AP-1 with positional and combinatorial logic were highlighted in *Drosophila*. Pluripotency factors (SOX2, POU5F1), CTCF, and SP1 were identified in human cells, with distance-weighted interactions consistent with prior studies [11, 8, 2].

Overall, LLMGEN programs capture interpretable motif syntax and combinatorial rules while achieving predictive performance competitive with much larger models. These results demonstrate the framework's ability to bridge accuracy and transparency, enabling biologically grounded hypothesis generation across species and tasks.

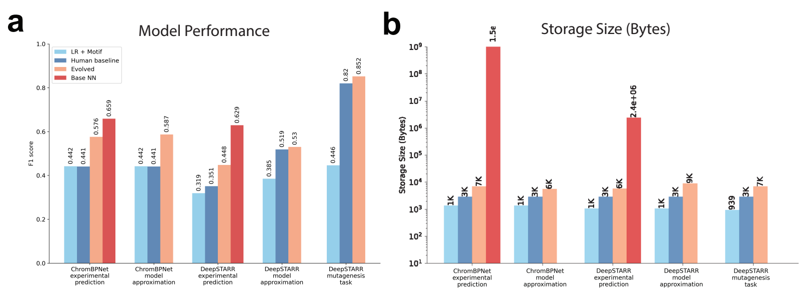

Figure 3: LLMGEN performance and analysis on DeepSTARR [11] and ChromBPNet [20] datasets. F1 scores (a) and model storage sizes (b, log scale) for LLMGEN ('Evolved') versus baselines and the original Neural Network (Base NN) across DeepSTARR and ChromBPNet tasks.

5 Discussion and Conclusion

LLMGEN introduces a new paradigm for interpretable genomic modeling by combining the symbolic reasoning of large language models with the iterative refinement of evolutionary search. It translates the behavior of deep models and functional track signals into concise, human-readable Python functions that enable mechanistic insight, hypothesis generation, and transparency.

Our results show that LLMGEN distills compact, high-performing rules across diverse tasks, with evolved programs rediscovering known regulatory motifs and grammars [17, 2, 8]. These findings highlight the biological relevance of the discovered rules. Integrating multiple modalities, LLMGEN programs directly encode functional logic and biologically grounded hypotheses. While prior-guided seeding accelerated search, the framework also produced plausible rules without explicit priors, suggesting adaptability.

Challenges remain as evolutionary search is computationally intensive [23, 32], LLM outputs may introduce biases [6], and compactness does not guarantee interpretability [7]. Experimental validation of evolved rules is also essential. Future work will focus on improving scalability, incorporating richer symbolic representations, and integrating human-in-the-loop refinement [31]. With advances in LLMs and stronger biological priors, LLMGEN could become a central tool for multi-modal, interpretable, and biologically grounded discovery in computational genomics.

References

[1] KD Ahlquist, Lauren A Sugden, and Sohini Ramachandran. Enabling interpretable machine learning for biological data with reliability scores. *PLOS Computational Biology*, 19(5):e1011175, 2023.

[2] Roadmap Epigenomics Consortium Integrative analysis coordination Kundaje Anshul 1 2 3 Meuleman Wouter 1 2 Ernst Jason 1 2 4 Bilenky Misha 5, Scientific program management Chadwick Lisa H. 53, and Principal investigators Bernstein Bradley E. 2 26 42 Costello Joseph F. 14 Ecker Joseph R. 9 Hirst Martin 5 18 Meissner Alexander 2 6 Milosavljevic Aleksandar 7 Ren Bing 8 13 Stamatoyannopoulos John A. 10 Wang Ting 21 Kellis Manolis 1 2. Integrative analysis of 111 reference human epigenomes. *Nature*, 518(7539):317–330, 2015.

[3] Žiga Avsec, Vikram Agarwal, Daniel Visentin, Joseph R Ledsam, Agnieszka Grabska-Barwinska, Kyle R Taylor, Yannis Assael, John Jumper, Pushmeet Kohli, and David R Kelley. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature methods*, 18(10):1196–1203, 2021.

[4] Žiga Avsec, Melanie Weilert, Avanti Shrikumar, Sabrina Krueger, Amr Alexandari, Khyati Dalal, Robin Fropf, Charles McAnany, Julien Gagneur, Anshul Kundaje, et al. Base-resolution models of transcription-factor binding reveal soft motif syntax. *Nature Genetics*, 53(3):354–366, 2021.

[5] Dylan Barth, Richard Van, Jonathan Cardwell, and Mira V Han. Supervised learning of enhancer–promoter specificity based on genome-wide perturbation studies highlights areas for improvement in learning. *Bioinformatics*, 40(6):btae367, 2024.

[6] Gonzalo Benegas, Chengzhong Ye, Carlos Albors, Jianan Canal Li, and Yun S Song. Genomic language models: opportunities and challenges. *Trends in Genetics*, 2025.

[7] Ashley Mae Conard, Alan DenAdel, and Lorin Crawford. A spectrum of explainable and interpretable machine learning approaches for genomic studies. *Wiley Interdisciplinary Reviews: Computational Statistics*, 15(5):e1617, 2023.

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

[9] Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, Nicolas Lopez Carranza, Adam Henryk Grzywaczewski, Francesco Oteri, Christian Dallago, Evan Trop, Bernardo P de Almeida, Hassan Sirelkhatim, et al. Nucleotide transformer: building and evaluating robust foundation models for human genomics. *Nature Methods*, 22(2):287–297, 2025.

[10] Bernardo P de Almeida, Hugo Dalla-Torre, Guillaume Richard, Christopher Blum, Lorenz Hexemer, Maxence Gélard, Javier Mendoza-Revilla, Priyanka Pandey, Stefan Laurent, Marie Lopez, et al. Segmentnt: annotating the genome at single-nucleotide resolution with dna foundation models. *bioRxiv*, pages 2024–03, 2024.

[11] Bernardo P de Almeida, Franziska Reiter, Michaela Pagani, and Alexander Stark. Deepstarr predicts enhancer activity from dna sequence and enables the de novo design of synthetic enhancers. *Nature genetics*, 54(5):613–624, 2022.

[12] Charles P Fulco, Joseph Nasser, Thouis R Jones, Glen Munson, Drew T Bergman, Vidya Subramanian, Sharon R Grossman, Rockwell Anyoha, Benjamin R Doughty, Tejal A Patwardhan, et al. Activity-by-contact model of enhancer–promoter regulation from thousands of crispr perturbations. *Nature genetics*, 51(12):1664–1669, 2019.

[13] Molly Gasperini, Andrew J Hill, José L McFaline-Figueroa, Beth Martin, Seungsoo Kim, Melissa D Zhang, Dana Jackson, Anh Leith, Jacob Schreiber, William S Noble, et al. A genome-wide framework for mapping gene regulation via cellular genetic screens. *Cell*, 176(1):377–390, 2019.

[14] Yanrong Ji, Zhihan Zhou, Han Liu, and Ramana V Davuluri. Dnabert: pre-trained bidirectional encoder representations from transformers model for dna-language in genome. *Bioinformatics*, 37(15):2112–2120, 2021.

[15] Peter K Koo, Antonio Majdandzic, Matthew Ploenzke, Praveen Anand, and Steffan B Paul. Global importance analysis: An interpretability method to quantify importance of genomic features in deep neural networks. *PLoS Computational Biology*, 17(5):e1008925, 2021.

[16] Peter K Koo and Matt Ploenzke. Improving representations of genomic sequence motifs in convolutional networks with exponential activations. *Nature Machine Intelligence*, 3(3):258–266, 2021.

[17] Evgeny Z Kvon, Tomas Kazmar, Gerald Stampfel, J Omar Yáñez-Cuna, Michaela Pagani, Katharina Schernhuber, Barry J Dickson, and Alexander Stark. Genome-scale functional characterization of drosophila developmental enhancers in vivo. *Nature*, 512(7512):91–95, 2014.

[18] Gherman Novakovsky, Oriol Fornes, Manu Saraswat, Sara Mostafavi, and Wyeth W Wasserman. Explainn: interpretable and transparent neural networks for genomics. *Genome Biology*, 24(1):154, 2023.

[19] Alexander Novikov, Ngan Vu, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco J. R. Ruiz, Abbas Mehrabian, M. Pawan Kumar, Abigail See, Swarat Chaudhuri, George Holland, Alex Davies, Sebastian Nowozin, Pushmeet Kohli, and Matej Balog. Alphaevolve: A coding agent for scientific and algorithmic discovery. 05 2025.

[20] Anusri Pampari, Anna Shcherbina, Evgeny Z Kvon, Michael Kosicki, Surag Nair, Soumya Kundu, Arwa S Kathiria, Viviana I Risca, Kristiina Kuningas, Kaur Alasoo, et al. Chrombpnet: bias factorized, base-resolution deep learning models of chromatin accessibility reveal cis-regulatory sequence syntax, transcription factor footprints and regulatory variants. *BioRxiv*, pages 2024–12, 2025.

[21] Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Model-agnostic interpretability of machine learning. *arXiv preprint arXiv:1606.05386*, 2016.

[22] Guillaume Richard, Bernardo P de Almeida, Hugo Dalla-Torre, Christopher Blum, Lorenz Hexemer, Priyanka Pandey, Stefan Laurent, Marie Lopez, Alexandre Laterre, Maren Lang, et al. Chatnt: A multimodal conversational agent for dna, rna and protein tasks. *bioRxiv*, pages 2024–04, 2024.

[23] Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, et al. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475, 2024.

[24] Evan E Seitz, David M McCandlish, Justin B Kinney, and Peter K Koo. Interpreting cis-regulatory mechanisms from genomic deep neural networks using surrogate models. *Nature machine intelligence*, 6(6):701–713, 2024.

[25] Avanti Shrikumar, Katherine Tian, Žiga Avsec, Anna Shcherbina, Abhimanyu Banerjee, Mahfuza Sharmin, Surag Nair, and Anshul Kundaje. Technical note on transcription factor motif discovery from importance scores (tf-modisco) version 0.5. 6.5. *arXiv preprint arXiv:1811.00416*, 2018.

[26] Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

[27] Amlan Talukder, Clayton Barham, Xiaoman Li, and Haiyan Hu. Interpretation of deep learning in genomics and epigenomics. *Briefings in Bioinformatics*, 22(3):bbaa177, 2021.

[28] Ziqi Tang, Nirali Somia, Yiyang Yu, and Peter K Koo. Evaluating the representational power of pre-trained dna language models for regulatory genomics. *bioRxiv*, 2024.

[29] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

[30] Shushan Toneyan and Peter K Koo. Interpreting cis-regulatory interactions from large-scale deep neural networks. *Nature Genetics*, 56(11):2517–2527, 2024.

[31] David S Watson. Interpretable machine learning for genomics. *Human genetics*, 141(9):1499–1513, 2022.

[32] Xingyu Wu, Sheng-hao Wu, Jibin Wu, Liang Feng, and Kay Chen Tan. Evolutionary computation in the era of large language model: Survey and roadmap. *IEEE Transactions on Evolutionary Computation*, 2024.

[33] Jian Zhou and Olga G Troyanskaya. Predicting effects of noncoding variants with deep learning–based sequence model. *Nature Methods*, 12(10):931–934, August 2015.

A Detailed LLMGEN Evolutionary Pipeline Steps

The LLMGEN pipeline (Sec 3.1, Fig. 1) operates through an iterative evolutionary process:

1. **Program Representation and Population:** Candidate solutions are Python functions. These may internally use features such as motif presence, *k*-mer content, GC content, or positional information. A population of candidate programs is maintained and updated across generations. To maintain diversity and reduce premature convergence, we optionally use an island model similar to FunSearch [23], where sub-populations evolve independently and periodically exchange promising candidates.
2. **LLM as Evolutionary Engine (Program Mutation):** An LLM (e.g., Gemini 1.5 Flash) serves as the mutation operator in the evolutionary loop. For each generation, high-performing programs are selected and passed to the LLM as context. The prompt requests an improved version of the provided function without specifying explicit operations or requiring few-shot examples. The LLM autonomously proposes modified versions that it deems more effective. This approach allows the LLM to flexibly explore variations without hand-crafted mutation templates.
3. **Automated Evaluation (Fitness Function):** Each candidate program $g(s)$ is evaluated using a multi-objective fitness function that balances:
   - **Fidelity:** The program's accuracy in predicting the output of the target DNN $f_{\text{DNN}}(s)$ or matching experimental labels. Metrics include F1 score for classification or MSE for regression.
   - **Validity:** All programs must be syntactically valid and executable Python code.
4. **Selection and Population Update:** Based on fitness scores, a selection algorithm (e.g., tournament selection) chooses which programs survive and which are replaced. The top programs are retained using elitism. Selected candidates are used to generate the next batch of LLM prompts for mutation.
5. **Termination:** The process continues for a predefined number of generations or until convergence. The final output is a set of high-performing, compact programs that approximate the desired target function or prediction task.

B Dataset Details

B.1 Enhancer–Promoter Dataset

- Source Data: Experimental results from the Gasperini et al., 2019 CRISPRi screen [13], as processed and modeled by Barth et al., 2024 [5].
- Total Candidate Pairs: The initial feature matrix in Barth et al. (2024) contained 65,374 candidate enhancer-promoter (EP) pairs before filtering for experimental validation.
- Input Features (Barth et al., 2024): The full feature set comprised 542 genomic features, including histone modifications (e.g., H3K27ac, H3K4me3, H3K27me3) at enhancers and TSS, gene expression, genomic distance, Hi-C contact, relative contact strength, regional chromatin density, and presence of 250 TFs at enhancers and TSS. Non-negative matrix factorization (NMF) was also used to generate clustered TF binding profiles.
- Output: Binary classification of functional vs. non-functional EPI pairs.

- Binarization: An EPI pair was labeled positive if CRISPRi perturbation of the enhancer resulted in significant down-regulation of the target gene (adjusted P-value < 0.1). Up-regulation was not considered positive.
- LLMGEN Feature Subset: For the EPI task, LLMGEN operated on a reduced subset of < 10 features, specifically: `normalized_h3K27ac`, `normalized_dhs`, `normalized_h3K4me1`, `normalized_h3K27me3`, `GATA1_e`, `TAL1_e`, `SUZ12_e`, `CTCF_e`, `distance`, and sequence.
- Train/Test Split (Gasperini data in Barth et al., 2024):
  - The dataset from Gasperini et al. (2019) was divided by setting aside chromosomes 5, 10, 15, and 20 as a dedicated test set.
  - After filtering out pairs with NA for pValueAdjusted, this resulted in 29,291 EPI pairs for training and 4,274 EPI pairs for the test set.

B.2 DeepSTARR Enhancer Dataset

- Source: Published data from de Almeida et al., 2022 [11].
- Cell type: *Drosophila melanogaster* S2 cells.
- Focus: Developmental enhancers.
- Sequence length: 249 bp.
- Global Enhancers Dataset: The original DeepSTARR test set comprised ≈40,000 sequences. This was further split into internal training (32,948 sequences) and test (8,238 sequences) sets for LLMGEN evaluation.
  - 3-bin discretization for activity: Low/Medium/High activity based on thresholds derived from the original DeepSTARR continuous prediction scores: Low < 0.0, Medium $0.0 \leq \text{score} \leq 2.0$, High > 2.0.
- Local Mutant Library: ≈30,000 variants generated around a specific reference sequence. This dataset was split into internal training (24,000 sequences) and test (6,000 sequences) sets. DNN predictions for these variants were binarized using an activity threshold of 0.5 from the original DeepSTARR model's output.

B.3 Human H1-hESC ATAC-seq Dataset

- Source: ATAC-seq data for the H1 human embryonic stem cell line (`syn63862944`), retrieved from the Synapse repository at `https://www.synapse.org/Synapse:syn63862944`.
- Target DNN: Pre-trained ChromBPNet model [**?**], also available at the same Synapse repository.
- Input sequences: 2,114 bp DNA sequences.
- Prediction task for ChromBPNet: Base-resolution accessibility profiles and total accessibility logcounts. LLMGEN interpreted the logcounts predictions.
- Dataset: The original ChromBPNet test dataset split, for fold-0 comprising 96,619 sequences from chromosomes 1, 3, and 5. Further split into train/test dataset using an 80/20 split (train: 77,295, test: 19,324).
- Discretization of logcounts:
  - 3 bins (Low/Medium/High): Using logcounts thresholds of 4.7 and 5.7.

C Enformer-Based Scoring for EPI Benchmark

To benchmark against Enformer [3] for the EPI task, our approach was consistent with methods described in the original Enformer publication for deriving enhancer-gene scores. Enformer is a deep learning architecture that predicts various genomic tracks, including gene expression (e.g., CAGE signal at a transcription start site, TSS), from input DNA sequences of up to approximately 200kb, thereby learning to integrate information from long-range interactions.

As detailed in [3], an enhancer's contribution to a gene's activity can be quantified by computing scores that reflect the impact of the enhancer's sequence on the predicted output at the gene's TSS. The Enformer paper describes several such methods, including using input gradients (gradient $\times$

input), attention weights, or *in silico* mutagenesis (ISM) to score the enhancer's influence relative to a specific TSS output (e.g., K562 CAGE prediction). We used the input gradients approach: for each enhancer-promoter pair, a continuous score was derived representing the enhancer's predicted regulatory effect on the promoter's activity, based on Enformer's output for the relevant TSS.

This continuous score was then binarized to enable classification. An optimal binarization threshold was determined using the Area Under the Precision-Recall curve (AUPR) on an appropriate data partition. Enhancer-promoter pairs with Enformer-derived scores above this threshold were classified as positive (i.e., predicted to be interacting or functional). The F1 score was then used as the final metric for evaluating the performance of this Enformer-based benchmark.

D Example System Prompt

```
You are an expert in computational biology and machine learning, with a focus on predicting
regulatory element activity from DNA sequences.

Your task is to evolve and improve a Python function called
`predict_sequence_class(sequence_str)` that predicts the activity class of **diverse
Drosophila developmental enhancers** (DeepSTARR Global dataset). These sequences are 249bp
long.

The function will take a DNA sequence as input and must output an integer class label (e.g., 0
for 'low activity', 1 for 'medium activity', 2 for 'high activity', or a similar binned scheme
depending on the target data). The exact meaning and number of these integer classes are
defined by the pre-binned target data your function's output will be compared against.

Your goal is to iteratively improve the prediction accuracy of this function through code
modifications, aiming for solutions that are both high-performing and interpretable.

**Key Considerations for Generating Candidate Functions:**

1.  **Focus on Diverse Enhancer Logic**: The dataset contains a variety of enhancers. Aim for
rules that can distinguish between different activity levels across this diverse set.
2.  **Explore Diverse Sequence Features & Logic**:
    * **Motif Discovery**: Since no motifs are provided upfront, the function should learn to
identify and utilize important short DNA sequences or patterns that correlate with different
activity levels.
    * **K-mer Analysis**: Consider k-mer frequencies or the presence/absence of specific k-
mers.
    * **Simple Sequence Properties**: GC content, CpG density, or other compositional
features.
    * **Positional Context**: Explore whether the location of certain discovered features
within the 249bp sequence influences activity.
    * **Combinatorial Rules**: Investigate how combinations of different discovered features
or sequence properties might predict enhancer activity.
    * **Rule-Based Logic**: Develop functions that use conditional logic (if/elif/else) based
on the features you derive from the sequence to assign it to an activity class.
    * **Thresholding**: Calculate scores based on features and apply thresholds to determine
the class.
3.  **Code Style & Constraints**:
    * The function must be pure Python. Standard library imports like `re` and `numpy` are
generally permissible if your evolutionary framework supports them. Avoid  large scientific
libraries unless explicitly allowed and necessary for a *simple, interpretable* rule.
    * **CRITICAL**: Do **not** instantiate or train complex machine learning models (e.g.,
SVMs, RandomForests, Neural Networks) inside the `predict_sequence_class` function. The
evolved logic should be rule-based or involve straightforward calculations on sequence
features.
    * Strive for functions that are reasonably concise and human-interpretable.
    * Ensure the function always returns an integer corresponding to a valid class label.
    * The wrapper function `run_surrogate` will handle basic error catching and length
validation, but `predict_sequence_class` should be robust if possible.

Provide clear, well-documented Python code for the `predict_sequence_class` function. The
evolutionary process will aim to find functions that accurately map DNA sequences to their
enhancer activity classes.
```

Figure 4: System prompt used to guide the LLM during function evolution for the DeepSTARR Global classification task. The prompt provides instructions for predicting enhancer activity classes from DNA sequence input using Python functions, emphasizing interpretability, motif logic, and GC/*k*-mer features.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: In the results, we have provided performance of LLMGEN produced programms as well as examples of human readable Python functions, supporting our claim that LLMGEN provides interpretable genomic predictors.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: In the main results we have clearly described that we still underperformed DNNs given we are providing a interpretable surrogate model. We also discussed computation costs in the discussion section.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [NA]

Justification: Our paper does not contain theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: We have described our dataset and pipeline, as well as how the baselines are generated. Datasets and baseline models used in this work is all open-access.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [No]

   Justification: All datasets described in our paper are open access. We are working on open-sourcing the code as this is still work in progress.

   Guidelines:

   - The answer NA means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: In our appendix we described how our data is retrieved and split into training and testing. We also provided an example prompt that we used for our framework training

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [NA]

   Justification: We do not include experiments needing statistcal significance analysis.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We have explicitly included memory usage of generated programs in our figures.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: Our research comply to the NeurIPs Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [NA]

    Justification: Our work does not involve societal impact as it works purely with open-sourced functional genomics data that comes from model organisms and cell lines.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.

- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: We do not release model and all datasets were previously open sourced

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: All creators are properly credited.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.

- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [NA]

    Justification: No new assets introduced.

    Guidelines:

    - The answer NA means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [NA]

    Justification: No crowdsourcing experiments and human subject research was conducted.

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [NA]

    Justification: The paper does not involve human subjects and does not need IRB approval

    Guidelines:

    - The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.

- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

    Answer: [Yes]

    Justification: LLMs are used as part of the methodology to generate symbolic programs.

    Guidelines:

    - The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.