Genesis: A Trait-Based Evolutionary Framework for Simulated Architectural Development in Large Language Models

Keywords: Large Language Models, Evolutionary Algorithms, Model Optimization, Digital DNA, Genetic Simulation, Artificial Intelligence, Model Architecture Evolution, PyTorch

Abstract

The rapid advancement of large language models (LLMs) raises questions about their architectural evolution. This paper introduces 'Genesis,' a novel framework that simulates architectural evolution in LLMs using genetic principles. By encoding traits such as layers, attention heads, dimensionality, and activation functions into digital DNA structures, we simulate evolutionary selection via a fitness-based mechanism. Our approach models how LLM traits may converge, mutate, and diverge under simulated selection pressureoffering insights into emergent architectural robustness.

I. Introduction

Large Language Models (LLMs) have emerged as transformative tools in artificial intelligence. Traditional design approaches rely on manual architecture tuning or automated neural architecture search (NAS). Inspired by natural evolution, we propose an alternative pathwaysimulated trait evolutionto discover optimal architectures. 'Genesis' explores this concept through digital genetic codes encoding model traits and simulates intergenerational selection dynamics.

II. Problem Statement and Proposed Solution

Current LLMs lack a self-evolving architecture mechanism. The absence of generational adaptation limits the exploration of alternative, possibly superior, architectural forms. We propose a simulation of architectural evolution based on digital trait recombination, crossover, and mutation using evolutionary algorithms. This allows a controlled yet dynamic discovery of high-performing architectural configurations.

III. Implementation / Development

A. Digital DNA Encoding: Each model architecture is encoded into a digital DNA sequence: num_layers, num_heads, hidden_dim, activation_fn.

These traits form the building blocks for crossover, mutation, and selection.

B. Genetic Operations:

Selection: Based on a fitness function reflecting architectural strength.

Crossover: Two parent DNAs exchange traits.

Mutation: Random alteration of DNA traits to explore diversity.

C. Phase 1: Evolutionary Simulation - We simulate several generations of architectural evolution using Python-based pseudocode. Top performers are preserved, weak configurations are eliminated.

D. Phase 2: PyTorch-Based Mini-Model Simulation - Each evolved DNA structure is used to instantiate a real PyTorch-based model. Though minimal in training, this phase validates DNA-architecture correlation.

IV. Evaluation Metrics

We consider: Trait diversity per generation, convergence speed, architectural fitness improvement, and mutation/crossover rate impact.

V. Model Training and Evaluation

Models instantiated from evolved DNA are compiled and minimally trained to validate architectural viability. Performance metrics such as accuracy, loss reduction, and computational cost are considered.

VI. Feature Importance Analysis and Expected Outcomes

While not trained on datasets, DNA trait impact is statistically analyzed. num_layers shows strongest correlation with fitness. activation_fn shows nuanced influence under specific combinations. Expected outcomes include natural emergence of robust trait configurations and converged architectures mirroring modern transformer best practices.

VII. Evaluation and Results

Key traits began converging towards optimal configurations early. High-performing DNA structures retained dominance across generations. Trait inheritance through crossover was effective in combining strong traits. Mutation introduced diversity while maintaining progression in fitness.

VIII. Future Directions

Integrate real dataset-based model training per generation. Include additional DNA traits (residual paths, normalization types). Extend to cross-modal evolutionary simulation. Introduce speciation and multi-objective fitness. Real-time adaptive evolution during model training.

IX. Related Work

Genesis diverges from traditional Neural Architecture Search (NAS) by simulating trait evolution rather than using search space heuristics. While NAS algorithms such as ENAS and AutoFormer focus on efficient exploration via reinforcement learning or differentiable architecture search, Genesis emphasizes long-term trait evolution, generational learning, and digital DNA diversity.

X. Conclusion

Genesis opens new doors to organic architecture discovery in AI. By treating model design as an evolving ecosystem, we unlock a new layer of understanding and creativity in artificial intelligence.

XI. References

[1] Vaswani et al., "Attention is All You Need," 2017.

[2] Real et al., "Regularized Evolution for Image Classifier Architecture Search," 2019.

[3] LeCun et al., "Deep Learning," Nature, 2015.

[4] Mitchell, M., "An Introduction to Genetic Algorithms," MIT Press, 1998.

[5] Liu et al., "Autoformer: Searching Transformers for Visual Recognition," 2021.