BEYOND CONDITIONAL COMPUTATION: RETRIEVAL-AUGMENTED GENOMIC FOUNDATION MODELS WITH GENGRAM

Huinan Xu Genos Team
Xuyang Feng Genos Team
Junhong Chen Genos Team
Junchen Liu Genos Team
Kaiwen Deng Genos Team
Kai Ding Genos Team

Shengning Long Genos Team
Jiaxue Shuai Genos Team
Zhaorong Li Genos Team
Shiping Liu Genos Team
Guirong Xue Genos Team
Zhan Xiao Genos Team
xz@zhejianglab.org

January 29, 2026

ABSTRACT

Current genomic foundation models (GFMs) rely on extensive neural computation to implicitly approximate conserved biological motifs from single-nucleotide inputs. We propose Gengram, a conditional memory module that introduces an explicit and highly efficient lookup primitive for multi-base motifs via a genomic-specific hashing scheme, establishing genomic "syntax". Integrated into the backbone of state-of-the-art GFMs, Gengram achieves substantial gains (up to 14%) across several functional genomics tasks. The module demonstrates robust architectural generalization, while further inspection of Gengram's latent space reveals the emergence of meaningful representations that align closely with fundamental biological knowledge. By establishing structured motif memory as a modeling primitive, Gengram simultaneously boosts empirical performance and mechanistic interpretability, providing a scalable and biology-aligned pathway for the next generation of GFMs. The code is available at `https://github.com/zhejianglab/Genos`, and the model checkpoint is available at `https://huggingface.co/ZhejiangLab/Gengram`.

1 Introduction

DNA sequences constitute the fundamental biological language of life, encoding the genetic information underlying cellular function and organismal development. Understanding the "syntax" of this language is central to deciphering biological processes and their dysfunctions [1, 2]. Recent advances in GFMs on architectures and tokenizers (e.g., k-mer, Byte Pair Encoding(BPE), single-base) have substantially accelerated progress in this direction.

Although single-nucleotide tokenization has demonstrated strong performance in downstream tasks, particularly those that evaluate fine-grained single-base effects [3], the majority of biologically functional genomic elements are defined by multi-nucleotide motifs with specific sequence patterns. Consequently, constrained by the Transformer architecture, existing Transformer-based GFMs lack an explicit mechanism to store and retrieve such functional motifs, instead relying on large-scale pretraining and dense computation to implicitly infer them from combinations of single-base representations [4]. This indirect inference process is inefficient and often limits performance on motif-dominated functional element detection tasks.

Inspired by DeepSeek's Engram—a conditional memory module that stores n-gram knowledge in a hash-based memory [5] — we propose Gengram, a conditional memory module specifically designed for genomic motif modeling. In Gengram, k-mers (with k = 1,2,3,4,5,6) and their corresponding embeddings are explicitly stored and leveraged through a gate-controlled conditional memory mechanism. Unlike Engram, which retrieves single preceding n-grams per position, Gengram introduces a **local window aggregation mechanism**, which aggregates multiple k-mer embeddings

within a fixed genomic window. This design allows the model to capture local contextual dependencies relevant to functional motifs while avoiding reliance solely on single-nucleotide composition.

Owing to the small vocabulary size of genomic sequences (ATCGN), Gengram introduces negligible additional computational overhead while enabling larger and more expressive motif memory tables. More importantly, Gengram can be seamlessly integrated into mainstream Transformer architectures. Our contributions can be summarized as follows:

We propose Gengram, a lightweight conditional motif memory module for Transformer-based GFMs that encodes multi-nucleotide motifs and integrates them via a structure-aware windowed aggregation mechanism, improving efficiency and interpretability without hard-coded biological rules.

Gengram demonstrates consistent and substantial performance improvements across downstream genomic tasks—especially motif-dominated functional element detection—achieving up to 14% improvement over state-of-the-art GFMs architectures under identical training protocols.

Gengram is architecture-agnostic, introducing negligible additional parameters (~60M) and computation, while consistently reducing training loss across diverse Transformer architectures, suggesting its potential as a reusable, standardized component for future GFMs.

Gengram is computationally efficient, achieving benchmark performance comparable to state-of-the-art models while utilizing fewer activated parameters and less training data. The Gengram module exhibits robust capability in modeling long sequences and enables stable load balancing in sparse Mixture of Experts (MoE) models.

Gengram spontaneously captures biologically meaningful structure, exhibiting reverse-complement symmetry in memory embeddings and context-dependent gating aligned with functional genomic regions (e.g., promoters and $5'$ UTRs), indicating structural reasoning rather than static pattern memorization.

2 Related Works

2.1 Current Architectures and Representation Paradigms in GFMs

2.1.1 Genomic Foundation Models

Current genomic modeling is dominated by Transformer-derived architectures [6] and non-Transformer alternatives, primarily represented by State Space Models (SSMs) [7]. Based on their underlying mechanisms, these models can be categorized into three paradigms: Bidirectional Encoders (e.g., DNABERT) [8]: Built upon the Transformer Encoder, these models utilize omnidirectional attention for local motif extraction. Autoregressive Decoders (e.g., Nucleotide Transformer) [9, 10]: Leveraging the Transformer Decoder, these models offer robust long-range modeling capabilities. State Space Models (e.g., Caduceus) [11]: Based on the Mamba architecture, SSMs achieve linear scaling via selective scanning.

3 Method

3.1 Architecture Overview

As shown in Figure 1, the Gengram module integrates efficient lookup with a motif prior: when predicting the current token, it uses a sliding-window mechanism within the Local Window immediately preceding it to retrieve all occurring contiguous substrings, i.e., motif priors. After deduplication, it looks up the corresponding values in a hash table; vectors with the same N are summed and averaged, vectors across different N are concatenated, and then, after a linear transformation, dot-product computation, and gating, the resulting signal is added to the backbone network via a residual connection.

3.2 Notation

Let the backbone produce an input representation and the previous-layer hidden states are

$$X \in \mathbb{R}^{n \times d_{\text{model}}}, X^{(\ell-1)} \in \mathbb{R}^{n \times d_{\text{model}}} \tag{1}$$

where n is the sequence length, d_{model} the hidden dimension, and X_{t}.

The N-gram lengths follow chapter 4.2

$$N \in \mathcal{N} = \{1, 2, 3, 4, 5, 6\} \tag{2}$$

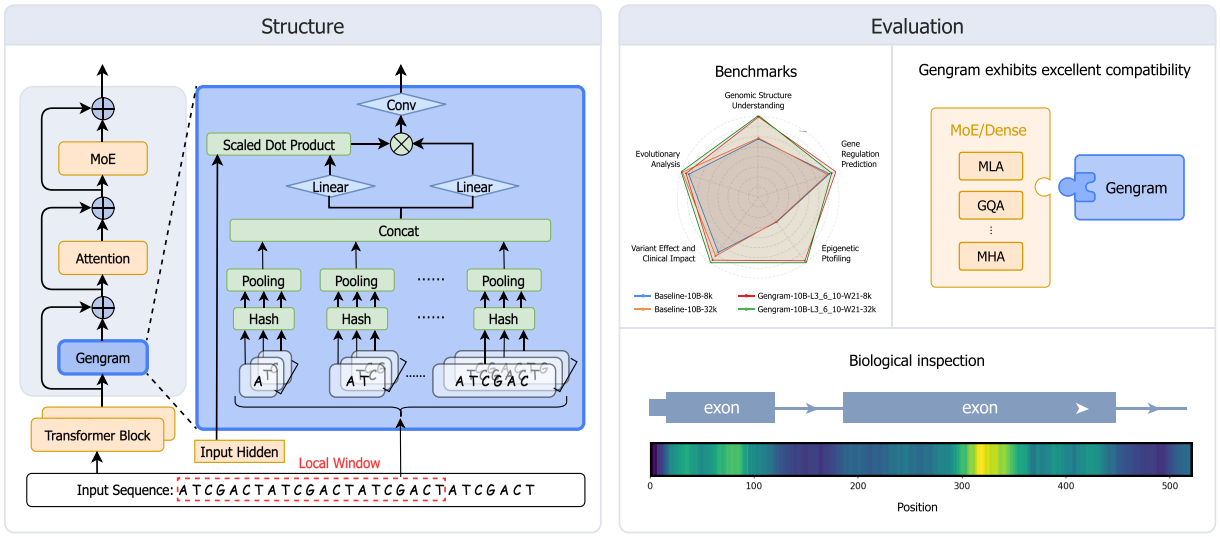

Figure 1: **Overview of the Gengram architecture and evaluation.** (Left) Gengram is integrated into a Transformer block via a residual connection and applied before the attention module. It maintains a motif memory implemented as a hash table with static keys derived from all possible k-mers (k=1,2,3,4,5,6) and learnable embedding values. For a given local window, k-mers are mapped to memory entries, aggregated within each k-mer level, and concatenated across levels. The resulting representation is then passed through a gate-controlled module to modulate the hidden states. (Right) Evaluation and analysis of Gengram. We conduct extensive benchmarks across diverse genomic tasks and Transformer architectures, demonstrating strong compatibility with different attention mechanisms (e.g., MHA, GQA, MLA). Biological inspection further shows that Gengram's activations align with functional genomic regions.

To capture multi-scale motifs. For each N, we maintain a learnable lookup hash table

$$\left\{\mathbf{M}^{(N)}\right\}_{N\in\mathcal{N}}, \quad \mathbf{M}^{(N)} \in \mathbb{R}^{|\Sigma|^N \times d_m} \tag{3}$$

where Σ is the token alphabet (A,T,C,G,N) and d_m is the memory vector dimension. Intuitively, $\mathbf{M}^{(N)}$ stores an embedding for each possible N-gram key; the key-space size is $|\Sigma|^N$. The mapping from an N-gram $g \in \Sigma^N$ to a row index is defined by a hash function

$$h_N(\cdot) : \Sigma^N \to \{0, 1, ..., |\Sigma|^N - 1\} \tag{4}$$

Since $|\Sigma|$ is small, $h_N(\cdot)$ can use a deterministic base-$|\Sigma|$ encoding, giving efficient collision-free indexing.

3.3 Local Window and N-gram Enumeration

For each position t, we define a causal window index set

$$\mathcal{W}_t = \{t - W, t - W + 1, \ldots, t - 1\} \tag{5}$$

We scan N-grams over the local window $\mathcal{W}_t$ to obtain a complete set of substrings, enabling the model to capture multi-scale motif information. The local window also reduces noise and supports efficient retrieval.

Within this window, we enumerate all contiguous N-grams using a sliding window scan, as shown in Algorithm 1:

$$\mathcal{G}_t^{(N)} = \{(s_j, \ldots, s_{j+N-1}) \mid j \in [t - W, t - N]\} \tag{6}$$

where s_j is the discrete token at position j. To make the aggregation focus on distinct motif types rather than their raw frequency, we first deduplicate the retrieved substrings:

$$\tilde{\mathcal{G}}_t^{(N)} = \text{Unique}\left(\mathcal{G}_t^{(N)}\right) \tag{7}$$

Algorithm 1 Sliding window search

Retrieve at Position t
Input: position t
Output: $\{m_t^{(N)}\}_{N\in\mathcal{N}}$
for $N \in \mathcal{N}$ **do**
  $j_{start} \leftarrow \max(1,\, t-W)$
  $j_{end} \leftarrow t-N$
  $sum \leftarrow \mathbf{0}_{d_m}$ *initialize accumulator*
  $cnt \leftarrow 0$ *count valid N-grams*
  for $j = j_{start}$ **to** j_{end} **do**
    $key \leftarrow h_N(s[j \dots j+N-1])$ *build N-gram key*
    $sum \leftarrow sum + M^{(N)}[key]$ *$O(1)$ lookup*
    $cnt \leftarrow cnt + 1$
  end for
  $m_t^{(N)} \leftarrow sum/\max(cnt, 1)$ *mean pooling*
end for
return $\{m_t^{(N)}\}_{N\in\mathcal{N}}$

3.4 Hash Retrieval and Within-N Aggregation

Each N-gram $g \in \Sigma^N$ is retrieved from the correspoding table by

$$v(g) = \mathbf{M}^{(N)}\left[h_N(g)\right], \quad g \in \tilde{\mathcal{G}}_t^{(N)} \tag{8}$$

We then average the retrieved vectors to obtain a fixed-size summary for length N:

$$m_t^{(N)} = \frac{1}{|\tilde{\mathcal{G}}_t^{(N)}|} \sum_{g\in\tilde{\mathcal{G}}_t^{(N)}} \mathbf{M}^{(N)}[h_N(g)] \tag{9}$$

We apply mean pooling to compress a variable number of retrieved N-gram values into a fixed-dimensional summary $m_t^{(N)}$. This choice stabilizes the magnitude of $m_t^{(N)}$across positions and sequences. Such scale stability makes the contribution of the memory branch easier to control and more robust during training.

3.5 Cross-N Fusion

We fuse multi-scale summaries by concatenation:

$$m_t = \mathrm{Concat}(m_t^{(1)}, m_t^{(2)}, m_t^{(3)}, m_t^{(4)}, m_t^{(5)}, m_t^{(6)}) \tag{10}$$

We then apply two linear projections to the same m_t:

$$z_t = m_t W_z + b_z,\ u_t = m_t W_u + b_u \tag{11}$$

Here z_t is used to compute a gate conditioned on the token representation, while u_t carries the memory content to be injected.

3.6 Scaled Dot-product Gating

We compute a gating scalar via a scaled dot product between normalized vectors:

$$\tilde{u}_t = \sigma\left(\frac{\mathrm{RMSNorm}(X_t)^{\mathrm{T}}\mathrm{RMSNorm}(z_t)}{\sqrt{d_{\text{model}}}}\right) u_t \tag{12}$$

We compute a gating scalar from the scaled dot product of the normalized input X_t and normalized projection z_t, where $\sigma(\cdot)$ is the sigmoid function and RMSNorm is root mean square normalization, to control how strongly the retrieved motif memory affects the backbone at each position; it is then dotted with u_t.

Then we align and mix the gated memory signal through a lightweight transformation, and add it residually to the previous-layer hidden state:

$$Y_t = \mathrm{SiLU}((\mathrm{RMSNorm}(\tilde{u}_t))) + X_t^{(\ell-1)} \tag{13}$$

The resulting $Y \in \mathbb{R}^{n \times d_{\text{model}}}$is fed to subsequent network components. This residual formulation stabilizes training while allowing the model to selectively incorporate explicit motif-level information from the windowed context.

3.7 Time Comlexity (shows as $O(n)$)

For each position t and each $N \in \mathcal{N}$, we enumerate approximately $\max(0, W - N + 1)$ contiguous N-grams within the lock-back window and perform one table lookup ($O(1)$) per N-gram. Therefore, the total retrieval cost over a length-n sequence is:

$$O\left(n \sum_{N \in \mathcal{N}} \max(0, W - N + 1)\right) = O(n|\mathcal{N}|W) \tag{14}$$

When W and $\mathcal{N}$ are treated as fixed hyperparameters (constants), the retrieval time is linear in the sequence length: $O(n)$.

4 Experimental Design

4.1 Datasets

4.1.1 Pre-training Dataset

The pre-training dataset is curated from the Human Pangenome Reference Consortium (HPRC, release 2) [12] and NCBI RefSeq databases [13], encompassing DNA sequences from humans and various primate species. To support different stages of model development, we systematically constructed three datasets: (1) a 50B-token dataset with a sequence length of 8,192, designed for the ablation studies; (2) a 200B-token dataset with an 8k context length; and (3) a 100B-token dataset with a 32k context length, both designated for the formal pre-training of the 10B model. Across all datasets, a balanced 1:1 ratio between human and non-human sequences was strictly maintained to ensure taxonomic parity. Detailed list of the datasets are provided in the Appendix Table 4,5 and 6.

4.1.2 Benchmark Datasets

We utilized several standard benchmark datasets to evaluate the model's performance, including the Genomic Benchmarks (GB), the Nucleotide Transformer Benchmarks (NTB), the Long-Range Benchmarks (LRB), and the Genos Benchmarks (GeB).[14, 9, 15, 3] From these sources, we selected 18 datasets covering five major categories: Genomic Structure Understanding, Gene Regulation Prediction, Epigenetic Profiling, Variant Effect and Clinical Impact, and Evolutionary Analysis. Detailed descriptions of each dataset are provided in the Appendix Table 7. All tasks follow a zero-shot evaluation protocol, where the pre-trained model serves as a frozen backbone for feature extraction. Downstream predictions are evaluated via Multi-Layer Perceptron (MLP) and XGBoost classifiers to quantify the model's zero-shot performance.

4.2 Multi-scale K-mers

The selection of multi-scale k-mers is motivated by both biological and computational considerations. Biologically, different functional genomic elements are characterized by sequence patterns of varying lengths: splice sites are primarily determined by short dinucleotide signals [16], codons encoding amino acids consist of triplets [17], while other regulatory elements—such as promoter motifs (e.g., the TATA box)—span longer sequence segments [18]. Computationally, we restrict k-mers to lengths up to 6 based on the efficiency–precision trade-off validated in GENERator [19].

4.3 Gengram Layer Selection Experiments

Gengram, as a conditional memory module, only needs to be inserted into a small subset of Transformer layers [5]. In our 1.2B-parameter model with 0.3B activated parameters (configuration details are provided in Appendix Table 13), we vary only the insertion layer of Gengram and compare the validation loss on a unified validation set, thereby selecting the optimal single-layer placement that achieves the lowest validation loss.

Table 1: **Downstream Evaluation of Multi-Layer:** Average performance across five downstream categories for single-layer vs. multi-layer insertions. 3, 6, 10 performs best overall, ranking first in four of five categories.

SELECTED LAYERS	10	10,11,12	9,10,11,12	3,6,10	3,6	3,10
GENOMIC STRUCTURE UNDERSTANDING	**0.8944**	0.8866	0.8900	0.8937	0.8838	0.8883
GENE REGULATION PREDICTION	0.8206	0.8154	0.8188	**0.8222**	0.8181	0.8196
EPIGENETIC PROFILING	**0.8334**	0.8329	0.8333	**0.8334**	0.8276	0.8306
VARIANT EFFECT & CLINICAL IMPACT	0.7577	0.7553	0.7489	**0.7678**	0.7602	0.7649
EVOLUTIONARY ANALYSIS	0.9101	0.9049	0.9059	**0.9117**	0.9093	0.9064

As shown in Figure 2 top, the layer-sweep results indicate that, as Gengram is inserted into progressively deeper layers, the validation loss decreases significantly. The reason is that, in shallow layers, the backbone hidden states are more biased toward local morphology and short-range statistical representations and are sensitive to base-/short-fragment patterns [20], which can easily lead to over-retrieval; however, shallow injection makes the model less likely to miss truly useful motif evidence. In deep layers, after multiple rounds of attention aggregation, the deep hidden states have already integrated longer-range dependencies, positional relationships, and information beyond the upstream window, and the representations on which the gating decision relies are more informative and more aligned with the task objective, enabling better discrimination between functional motifs and non-functional repeats, thereby yielding lower validation loss [21, 22].

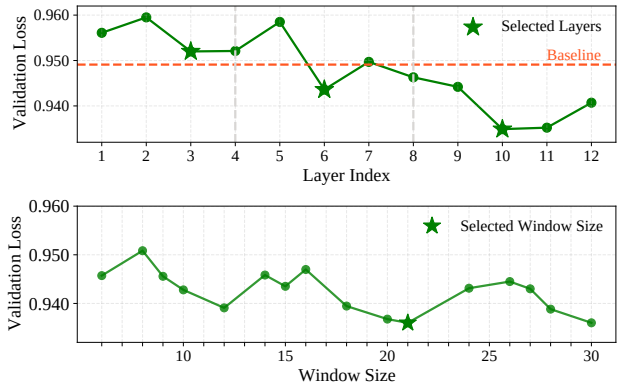

Figure 2: **Parameter Selection Experiments. Top:** Validation loss when inserting Gengram into individual Transformer layers in a 1.2B-parameter (0.3B activated) model trained on 50B tokens. The red curve denotes the baseline without Gengram. **Bottom:** Validation loss under different Gengram window sizes.

Validation loss from single-layer insertion alone is insufficient to conclude. Because shallow and deep layers are complementary, we evaluate multi-layer combinations. Prior studies commonly partition Transformer depth into shallow, middle, and deep stages: shallow layers mainly form the basic representational space, middle layers emphasize abstraction and compression, and deep layers exhibit stronger reasoning capability and task specialization [23]. Following this view and inspired by the hierarchical processing of the cerebral cortex—transitioning from basic feature extraction in posterior sensory regions (corresponding to shallow layers) and information integration in medial regions (middle layers) to high-level cognitive decision-making in anterior regions (deep layers)—we select the best-performing layer from each stage (namely layers {3, 6, 10}) as a representative shallow/middle/deep trio [24]. This combination effectively captures the model's full trajectory from base-level detail scanning to high-level functional abstraction. In addition, guided by the single-layer rankings, we test several competitive combinations, including {10, 11, 12}, {9, 10, 11, 12}, {3, 6}, and {3, 10}. We assess multi-layer benefits with a suite of downstream evaluations. As reported in Table 1, we summarize the average performance across five downstream evaluation categories for single-layer versus multi-layer insertions. The {3, 6, 10 configuration performs best overall, ranking first in four out of five categories.

4.4 Window Size Selection Experiments

Using the 1.2B parameter model's training setup (see Appendix Table 13), we determined the optimal window length for Gengram with multiples of 2 and 3. Keeping the backbone, data, training budget, and all training hyperparameters—such as sequence length, batch size, optimizer, and learning rate—exactly the same, and inserting the Gengram module at the tenth layer based on preliminary single-layer experiments, we trained models with different Gengram window lengths under the same token/step budget (i.e., 50B tokens). We then compared the validation loss on a fixed validation set to select the window length that achieved the lowest validation loss. This optimal window length was subsequently used as the baseline for follow-up experiments.

By comparing the validation loss under different window sizes, as shown in Figure 2 bottom, we found that the loss reached its minimum when the window size was set to approximately 21. This optimal length coincides with the fundamental structural property of B-form DNA, the predominant natural DNA: it completes one helical turn approximately every 10.4 10.5 base pairs (bp) in cellular environments[25]. Consequently, base pairs separated by every two turns (i.e., approximately 21 bp) are positioned in the same face of the helix [26]. This spatial arrangement has a significant impact on functions related to transcriptional regulation, particularly those involving protein binding. This structural consistency provides a universal constraint across the entire genome, including non-coding regions [27]. By explicitly modeling this 21-bp periodicity, our model effectively captures these phasing correlations, allowing it to leverage the stereochemical patterns that recur with every two helical turns—such as mutation susceptibility or protein-binding preferences. Therefore, the Gengram module moves beyond local linear context to incorporate global structural priors, significantly improving the performance of whole-genome modeling [28].

4.5 Pretraining Setup

To validate the effectiveness of the proposed Gengram module, we conducted experiments by training a 10B-parameter model named Gengram-10B-L3_6_10-W21-8k using the Megatron-LM framework on a distributed system with 8 GPUs per node and pipeline parallelism of 2. The model architecture consisted of 12 transformer layers with a hidden size of 4096 and 16 attention heads, incorporating RMSNorm, Rotary Position Embedding (RoPE), Switched Gated Linear Unit (SwiGLU) activation, and group query attention with 8 query groups (see Appendix Table 12). The Gengram module was enabled at layers 3, 6, and 10, configured with n-gram sizes from 1 to 6, an embedding dimension of 1024 per n-gram, token IDs (8, 5, 6, 7, 9) representing nucleotides A, C, G, T, and N respectively, a short convolution of kernel size 4, and a local window size of 21. The design also employed MoE architecture with 8 experts, top-k routing (k=2), auxiliary loss-based load balancing, and Grouped GEMM (General Matrix Multiplication) operations. The model was trained on a mixed human and non-human genomic dataset comprising approximately 25 million samples (200B tokens), with a sequence length of 8192 and maximum position embeddings of 8192. Key hyperparameters included a global batch size of 1024, an initial learning rate of 1e-4 decaying cosine to 1e-5, weight decay of 0.1, gradient clipping at 1.0, and BF16 precision with flash attention. To ablate the contribution of the local window mechanism, a variant without the window, Gengram-10B-L3_6_10-8k, was also trained. Additionally, a baseline model without the Gengram module, Baseline-10B-8k, was trained for comparison.

5 Results & Analysis

5.1 Enhanced Performance of the Gengram Module and the Role of the Window Mechanism

A systematic evaluation demonstrates the significant contribution of the Gengram module to genomic sequence modeling. Compared to the Baseline-10B-8k, as can be seen from Figure 3 top, the Gengram-10B-L3_6_10-W21-8k model exhibits consistent improvements across almost all assessed metrics. This comprehensive enhancement underscores the overall effectiveness of the Gengram module in strengthening the model's capacity to capture intricate genomic patterns. Furthermore, an ablation study on the window mechanism—comparing Gengram-10B-L3_6_10-W21-8k with its window-free counterpart, as illustrated in Table 2, Gengram-10B-L3_6_10-8k—reveals that the local window contributes substantially to these gains. The performance improvement is particularly pronounced in tasks requiring the recognition of fine-grained local sequence patterns, such as splice site detection and histone mark prediction, highlighting the mechanism's role in processing immediate genomic context.

The Gengram module's capability is further underscored when comparing the Gengram-10B-L3_6_10-W21-8k model to the Baseline-10B-32k (see Figure 3). The former, trained on 8k sequences, achieves performance that not only matches but exceeds the latter, which was trained on significantly longer 32k sequences based on Baseline-10B-8k. This result indicates that the Gengram module confers a powerful long-context modeling ability, effectively compensating for the shorter nominal context length without requiring the substantial additional computational cost typically associated with continued pretraining (CPT) on extended sequences. The most compelling evidence for the window mechanism's

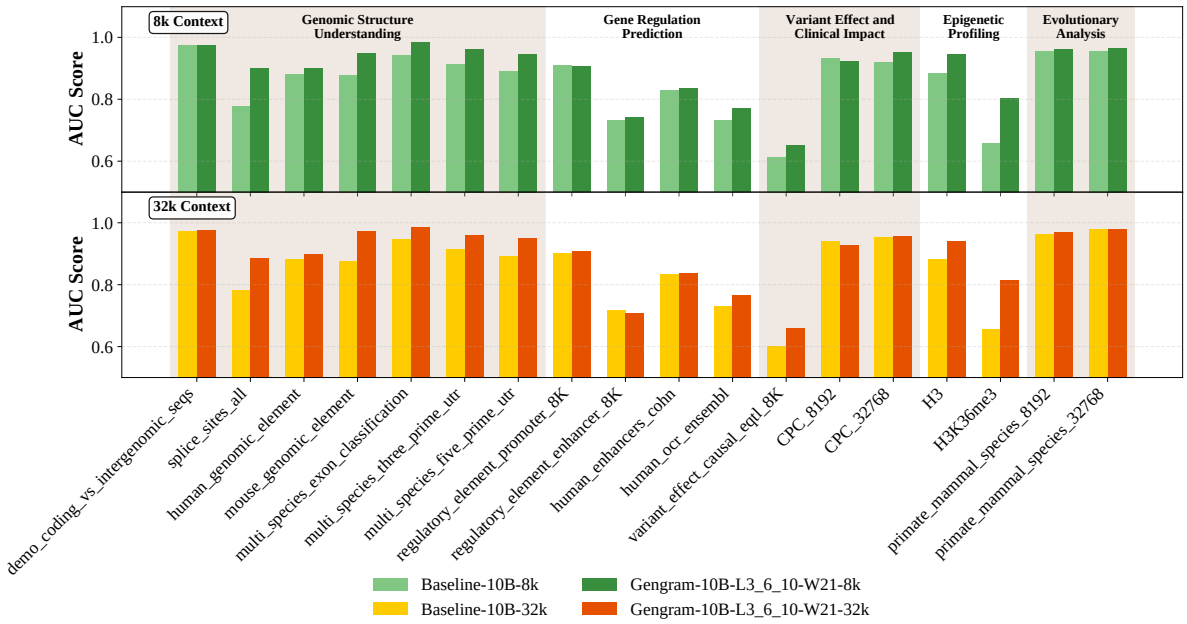

Figure 3: Performance Comparison of Baseline and Gengram Models at 8k and 32k Context Lengths

intrinsic long-context capability comes from the CPT experiment. Adapting the window-less Gengram-10B-L3_6_10-8k to 32k sequences yields a massive performance boost. In stark contrast, applying the same CPT to the window-equipped Gengram-10B-L3_6_10-W21-8k model results in only marginal gains (as shown in table 2). This stark divergence strongly suggests that the local window mechanism itself inherently equips the model with a capability analogous to long-context modeling.

This phenomenon can be interpreted through the lens of inductive bias. The local window acts as a powerful architectural constraint that guides the model to prioritize locally contiguous information. When this mechanism is stacked across multiple layers of the network, it progressively expands the model's effective receptive field. This structured approach allows the model to efficiently build meaningful representations of extensive genomic regions by systematically composing local features, thereby learning to capture longer-range dependencies even within a shorter nominal context window. This inductive bias not only enhances local pattern recognition but fundamentally strengthens the model's capacity for integrated long-range dependency modeling, providing a computationally efficient pathway to sophisticated genomic sequence analysis.

5.2 Benchmarking Results: State-of-the-Art Performance with Superior Efficiency

Despite being trained on a significantly smaller dataset of only 200B tokens—an order of magnitude less than competitors like Genos-10B (2.2T) and Evo2-40B (9.3T) — and despite utilizing significantly fewer activated parameters (2.87B activated parameters in Gengram-10B, compared to 40.3B total/activated parameters in Evo2-40B and 3B in GENERATOR-3B, as shown in Table 8), the Gengram-10B-L3_6_10-W21-8k model demonstrates highly competitive and often superior performance across key genomic benchmarks (see Table 3). It achieves state-of-the-art AUC score in Multi-species Exon Classification (0.9832) and excels in Splice Site Identification (0.9009), as shown in Table 3, significantly outperforming its direct counterpart Genos-10B. Furthermore, Gengram-10B shows strong evolutionary analysis capabilities, outperforming the much larger Evo2-40B in Primate-Mammal Species Classification tasks. This combination of top-tier performance in specific tasks like genomic structure understanding and its remarkable training efficiency highlights Gengram-10B as an exceptionally parameter-effective model for genomic sequence analysis.

Table 2: CPT Adaptation Results for Models With and Without Local Window.

		GENGRAM-10B-L3_6_10-8K	GENGRAM-10B-L3_6_10-W21-8K	GENGRAM-10B-L3_6_10-32K	GENGRAM-10B-L3_6_10-W21-32K
GENOMIC STRUCTURE UNDERSTANDING	DEMO_CODING_VS_INTERGENOMIC_SEQS	0.9412	0.9733	0.9818	0.9776
	SPLICE_SITES_ALL	0.8453	0.9009	0.8465	0.8858
	HUMAN_GENOMIC_ELEMENT	0.8842	0.8982	0.8911	0.8998
	MOUSE_GENOMIC_ELEMENT	0.9603	0.9495	0.9422	0.9722
	MULTI_SPECIES_EXON_CLASSIFICATION	0.8903	0.9832	0.9794	0.9858
	MULTI_SPECIES_THREE_PRIME_UTR	0.8759	0.9601	0.9603	0.9598
	MULTI_SPECIES_FIVE_PRIME_UTR	0.8397	0.9443	0.9521	0.9508
GENE REGULATION PREDICTION	REGULATORY_ELEMENT_PROMOTER_8K	0.8861	0.9074	0.9121	0.9097
	REGULATORY_ELEMENT_ENHANCER_8K	0.7054	0.7401	0.7267	0.7065
	HUMAN_ENHANCERS_COHN	0.7605	0.8358	0.8513	0.8375
	HUMAN_OCR_ENSEMBL	0.7216	0.7714	0.7785	0.7663
VARIANT EFFECT AND CLINICAL IMPACT	VARIANT_EFFECT_CAUSAL_EQTL_8K	0.6078	0.6518	0.6189	0.6607
	CPC_8192	0.8875	0.9219	0.9269	0.9283
	CPC_32768	0.864	0.952	0.9606	0.9572
EPIGENETIC PROFILING	H3	0.8934	0.9434	0.941	0.9417
	H3K36ME3	0.7227	0.8039	0.8045	0.8161
EVOLUTIONARY ANALYSIS	PRIMATE_MAMMAL_SPECIES_8192	0.7275	0.9611	0.9676	0.9691
	PRIMATE_MAMMAL_SPECIES_32768	0.6933	0.963	0.9785	0.9793

5.3 Generalizability Assessment of Gengram

The results are shown in Figure 4, where we observe that across different architectures and different attention mechanisms, adding the Gengram module consistently leads to lower validation loss. Gengram is integrated into each Transformer block as a residual branch, which enables strong compatibility with different network architectures. To systematically verify its cross-architecture generality and consistent gains, we conduct controlled experiments across two architectural families, including MoE (1.2B total parameters, 0.3B activated parameters) and Dense (0.3B parameters) [29]. Within each family, we further perform cross integration with three mainstream attention mechanisms, Multi-Head Attention(MHA; [6]), Multi-Head Latent Attention(MLA; [29]), and Grouped-Query Attention(GQA; [30]), and train all settings for 50B tokens. This yields 6 experimental groups for a systematic comparison of Gengram's adaptability and effectiveness (detailed configurations are provided in Appendix Table 9, Appendix Table 10 and Appendix Table 11). These results indicate that Gengram has strong adaptability and stable gains; moreover, as a lightweight module with extremely small parameter size, it can reliably improve performance without increasing training overhead, and has the potential to become an essential component of GFMs in the future.

Table 3: **Benchmarking Results:** GENGRAM-10B-L3_6_10-W21-8k vs other GFMs. The first place is in bold; the second placeis in italics.

	Model	GENGRAM-10B-L3_6_10-W21-8K	GENOS-10B	EVO2-40B	GENERATOR-3B	NT-V3-PRETRAIN (650M)
	TRAINED TOKENS	**200B** (1×)	2.2T (11×)	9.3T (46.5×)	386B (1.93×)	11T (55×)
GENOMIC STRUCTURE UNDERSTANDING	DEMO_CODING_VS_INTERGENOMIC_SEQS	0.9733	**0.9914**	*0.9886*	0.9855	0.983
	SPLICE_SITES_ALL	*0.9009*	0.7990	**0.9138**	0.8071	0.8644
	HUMAN_GENOMIC_ELEMENT	0.8982	0.8890	0.9103	*0.9173*	**0.9189**
	MOUSE_GENOMIC_ELEMENT	0.9495	*0.9521*	**0.9888**	0.9015	0.9092
	MULTI_SPECIES_EXON_CLASSIFICATION	**0.9832**	*0.9755*	0.9332	0.9455	0.9615
	MULTI_SPECIES_THREE_PRIME_UTR	*0.9601*	0.9511	**0.9634**	0.9400	0.9408
	MULTI_SPECIES_FIVE_PRIME_UTR	*0.9443*	0.9317	**0.9895**	0.8841	0.9084
GENE REGULATION PREDICTION	REGULATORY_ELEMENT_PROMOTER_8K	0.9074	**0.9249**	*0.9227*	0.9195	0.922
	REGULATORY_ELEMENT_ENHANCER_8K	0.7401	**0.7532**	*0.7527*	0.7390	0.7415
	HUMAN_ENHANCERS_COHN	*0.8358*	**0.8552**	0.7756	0.8181	0.8184
	HUMAN_OCR_ENSEMBL	**0.7714**	0.7623	*0.7635*	0.7270	0.7426
VARIANT EFFECT AND CLINICAL IMPACT	VARIANT_EFFECT_CAUSAL_EQTL_8K	0.6518	0.6773	**0.7054**	*0.6920*	0.687
	CPC_8192	0.9219	**0.9522**	0.9401	0.9315	*0.9468*
	CPC_32768	0.952	**0.9625**	*0.9611*	0.9237	0.9426
EPIGENETIC PROFILING	H3	*0.9434*	0.9400	0.9311	0.9163	**0.9505**
	H3K36ME3	0.8039	0.7658	**0.8823**	0.8247	*0.8535*
EVOLUTIONARY ANALYSIS	PRIMATE_MAMMAL_SPECIES_CLASSIFICATION_8192	*0.9611*	**0.9739**	0.8867	0.8424	0.8652
	PRIMATE_MAMMAL_SPECIES_CLASSIFICATION_32768	*0.9630*	**0.9804**	0.9101	0.8709	0.8928

5.4 Gengram Stabilizes MoE's Load Balancing

The load-balancing results in Figure 5 show that Gengram exhibits strong stability across different sparsity levels.

Load imbalance instability is frequently observed in mainstream MoE architectures, and has long been a key challenge during sparse model training; for example, the ST-MoE model introduces router z-loss during training to improve stability [31]. During Gengram training, we observed that load balancing is more stable than training without Gengram. To validate this observation, we conducted load-balancing tests on a 10B model while keeping all other conditions unchanged, under Top-2 / 128 experts, Top-2 / 64 experts, and Top-2 / 32 experts (Detailed configurations are provided in Appendix Table 14), and monitored the load-balancing loss curves.

Under sparse computation, load imbalance often arises from a "positive feedback chain": routing/activation scores are contaminated by task-irrelevant high-frequency noise, and once such noise is repeatedly injected, it amplifies the advantage of certain routes/keys, eventually causing a small number of paths to be persistently selected [32]. Meanwhile, genomic data further exacerbates imbalance: substantial evidence indicates that repetitive and low-complexity regions occupy a large fraction of the genome, and many k-mers are highly frequent yet weakly informative for biological function [33, 34]. We argue that the key of Gengram is that the module first absorbs such local-pattern noise evidence into the memory branch, but does not write it back to the backbone unconditionally: we introduce a gate at the output of memory aggregation; gating can increase sparsity [35] and filter out noise, such that when the backbone is insufficient to distinguish functional motifs from noise, the gate attenuates this memory injection, thereby breaking the feedback loop;

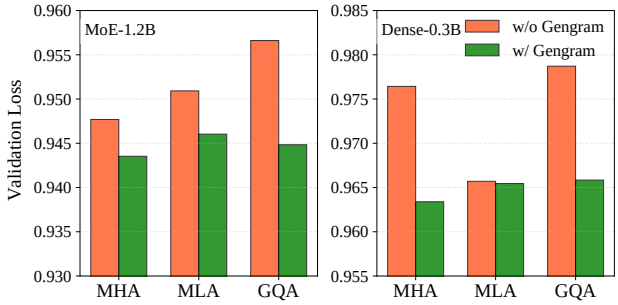

Figure 4: **Validation-Loss Comparison:**Validation-loss bar chart under MoE and Dense architectures across GQA/MLA/MHA.

consequently, even at higher sparsity, the variance of activation/routing scores is suppressed, and the load distribution becomes closer to uniform and more stable.

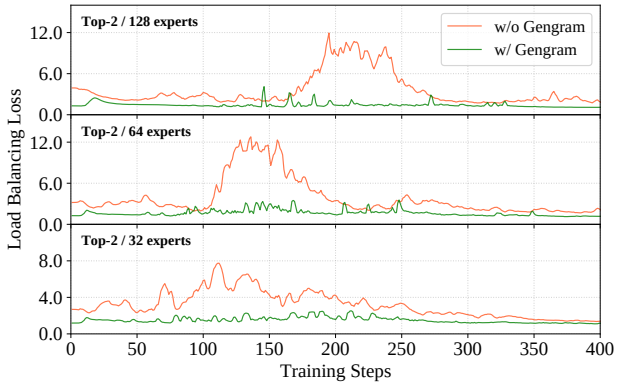

Figure 5: **Cross-Sparsity Load-Balancing:**Load-balancing loss curves with and without the Gengram module under Top-2 / 128, 64, and 32 experts, showing consistent stabilization across sparsity settings.

5.5 Intrinsic Mechanisms and Interpretability

5.5.1 Window Selection

We analyze the norm of the block output hidden states: for each position t, we take the final hidden state vector after self-attention and Gengram residual injection, and compute its $L2$ norm

$$\|h_t\|_2 \leftarrow h + \text{Gengram}(h, \text{input\_ids}) \tag{15}$$

This metric captures the position-wise magnitude of representations and serves as an interpretable signal closely related to numerical stability and the strength of residual updates. Since the hidden state already includes the attention contribution, the observed changes reflect how Gengram modulates the entire block representation via its residual update, rather than the Engram branch alone. We take a checkpoint trained with the non-windowed Gengram configuration and run two inference-time forward passes:

(i) the original non-windowed implementation to obtain the position-wise $\|h_t\|_2$ curve;

(ii) keeping the checkpoint weights unchanged, we replace the Engram readout/aggregation with a windowed variant (e.g., window size 21) and recompute $\|h_t\|_2$.

As shown in Figure 6, this is a "structure intervention with frozen weights": if the windowed variant substantially changes the magnitude fluctuation pattern of hidden states, it indicates that the window's effect is not due to additional

training, but due to a structural constraint on memory retrieval/aggregation, which alters the statistics of the residual update and thus the block's output dynamics.

In the non-windowed implementation, $\|h_t\|_2$ exhibits more pronounced high-frequency jitter and spiky fluctuations across positions, suggesting that the Gengram residual injection introduces more uneven magnitude perturbations along the sequence. After introducing a window (e.g., 21 bp), the $\|h_t\|_2$ curve becomes smoother, local spikes are suppressed, and the magnitude variation is more bounded/regulated. This indicates that windowed retrieval converts highly position-sensitive "point-like" memory triggering into a more consistent local aggregation response, reducing abrupt local changes in representation magnitude—an effect of structural stabilization.

Importantly, since we measure the full block output, this stability implies not only a smoother Engram branch, but also reduced uncertainty in how the Engram residual composes with the attention residual, yielding more stable hidden states overall.

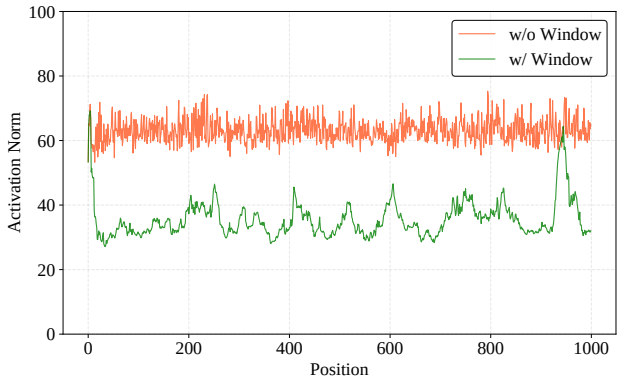

Figure 6: **Windowed Gengram stabilizes block output representation magnitude.** We plot the position-wise L2 norm of the Transformer block output hidden states $\|h_t\|_2$, which includes self-attention and the Gengram residual injection. Using the same checkpoint trained without windowing, we compare inference-time forward passes with (green) vs. without (orange) windowed Gengram retrieval. Windowing produces smoother and more bounded magnitude dynamics, suggesting that local aggregation in memory readout suppresses spiky residual perturbations and stabilizes the composition between Engram and attention updates.

5.5.2 Mechanistic Interpretation of Training Acceleration

Gengram accelerates optimization by front-loading predictive computation: shallow representations become aligned with the model's final decision substantially earlier, effectively reducing the effective depth required to reach a prediction-ready state [5].

We quantify layer-wise prediction readiness with a LogitLens-style diagnostic[36]. For each layer hidden state $h^{(l)}$, we apply the final LM head to obtain an intermediate next-token distribution $p^{(l)}(\cdot \mid x)$. We then compute $\mathrm{KL}\big(p^{(L)}(\cdot \mid x) \,\|\, p^{(l)}(\cdot \mid x)\big)$, where $p^{(L)}$ is the final-layer norm distribution. Smaller KL means that the layer l representation, under a fixed readout, already induces a distribution close to the model's final decision—i.e., it is closer to being prediction-ready.

Benefiting from Gengram, at injected layers ($l \in \{3, 6, 10\}$), the model performs a residual update $h^{(l)} \leftarrow h^{(l)} + \mathrm{engram}\big(h^{(l)}, \mathrm{input\_ids}\big)$, which explicitly writes reusable motif memory into the residual stream. This changes the allocation of work across depth: rather than requiring deeper layers to repeatedly reconstruct local predictive primitives from scratch, Gengram makes those primitives available earlier and in a form that is directly consumable by the same final head. The local window constraint (W=21) further stabilizes this contribution by enforcing local, phase-consistent aggregation, limiting noisy or inconsistent retrieval and enabling intermediate states to settle into a prediction-aligned regime sooner.

Earlier prediction readiness shortens the effective depth over which the model must transform representations before they become decision-relevant. Concretely, when shallow layers already produce head-readable predictive structure, gradient credit assignment no longer needs to traverse as many layers to reinforce useful features. The layer-wise KL

collapse therefore offers an interpretable mechanism for training acceleration: Gengram shifts predictive representation formation to earlier layers, reducing the burden on deeper layers and improving optimization efficiency under identical training and readout.

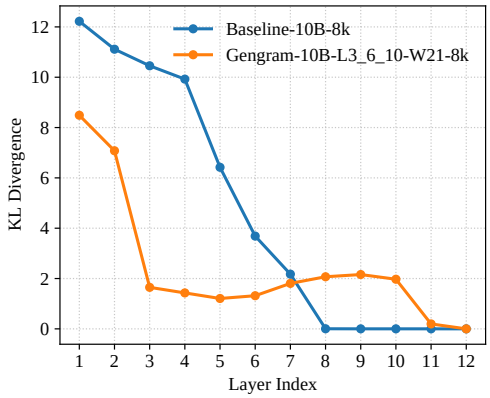

Figure 7: **Layer-wise prediction readiness via LogitLens-KL:** Compared to baseline-10B-8k, Gengram-10B-L3/6/10-W21-8k exhibits a much faster KL drop in early layers.

5.5.3 Gengram residual contribution aligns with motif

We find that Gengram's intervention along a genomic sequence is extremely sparse and high-contrast. Defining the activation strength as $s_\ell(t) = \|\mathbf{r}_\ell(t)\|_2$, where $\mathbf{r}_\ell(t) = \mathrm{Gengram}_\ell(\mathbf{h}_\ell(t), x_t)$. Then we plot the per-position residual-write magnitude $s_\ell(t)$, the signal concentrates into sharp peaks occupying only a small fraction of positions, while the majority of the sequence remains near baseline (Fig. 8). Importantly, these peaks are not randomly distributed: they are enriched around motif-like elements and functional boundaries, including promoter-proximal signals (TATA-like substring) and low-complexity tracts (poly-T).

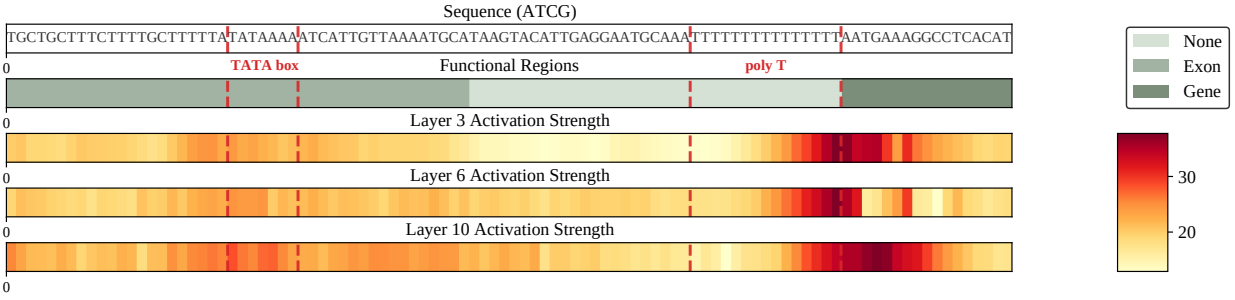

Figure 8: **Gengram residual contribution aligns with functional elements.** Top: Sequence with motif hits (TATA-like, poly-T) and region labels. Bottom: Gengram residual-write strength $s_\ell(t)$ at L3/6/10. Peaks align with motifs/boundaries and are slightly left-shifted to motif ends, consistent with forward-only windowed matching.

By design, $s_\ell(t)$ is derived from the additive residual written by Gengram at position t. Therefore, a localized peak means that the module chooses to modify the hidden state strongly at that locus. The systematic enrichment near motifs/boundaries supports the hypothesis that Gengram acts as a motif-sensitive memory pathway. While the alignment does not by itself prove causal necessity of the motif for downstream predictions, it establishes a direct mechanistic footprint: Gengram's writes are targeted to loci carrying concentrated sequence evidence.

The spatial distribution of residual writes changes systematically with depth. In early/mid Gengram layers (L3/L6), the strongest residual writes are tightly anchored to the poly-T or boundary locus: both L3 and L6 reach their maxima

at position 82, i.e., at the end of the poly-T tract immediately preceding the intergenic region $\rightarrow$ Gene transition. In contrast, the deepest Engram layer (L10) reaches its maximum at position 86, inside the Gene region, and its top-10 peaks are predominantly in Gene (8/10 for L10 vs. 3/10 for L6). This structured shift—from boundary-adjacent peaks in early/mid layers to gene-internal concentration in deeper layers—supports a hierarchical mechanism: earlier layers act as motif/boundary "writers" that inject localized evidence, while deeper layers integrate and propagate this evidence into more context-dependent representations.

Finally, these patterns are more consistent with motif memory than a rigid motif dictionary. A dictionary-like mechanism would tend to collapse biologically "equivalent" k-mers into shared prototypes, producing uniform responses across synonymous realizations. Instead, the observed residual-write signatures are compatible with context-dependent retrieval: biologically related patterns can remain separable when embedded in different surrounding contexts. This is consistent with memory retrieval indexed jointly by substring and hidden state, rather than by a fixed equivalence class over k-mers.

6 Conclusion

In this work, we introduced Gengram, a conditional motif memory module that addresses a core limitation of Transformer-based GFMs by explicitly encoding and indexing biologically functional multi-nucleotide motifs. Through a gate-controlled memory pathway with local window aggregation, Gengram enables direct motif-level access while preserving the flexibility of standard attention-based sequence modeling.

Across extensive experiments, Gengram consistently improves both performance and training efficiency over prior state-of-the-art GFMs, yielding substantial gains on motif-dominated tasks, long-range sequence modeling, and improved load balancing behavior. Moreover, Gengram remains effective across diverse architectures and attention mechanisms while introducing negligible computational overhead and enabling stable load balancing in sparse MoE models.

Beyond empirical improvements, mechanistic analyses reveal that Gengram learns biologically meaningful structure in its memory space, supporting its role as a reusable motif memory rather than a static pattern dictionary. Taken together, these results suggest that Gengram holds promise for scaling GFMs to longer sequence contexts and extending them to multi-omics modeling settings. We believe that Gengram represents an important architectural direction for future GFM development.

7 Limitations and Future Works

This work has several limitations that point to important directions for future research. First, our current study primarily validates Gengram as a modular architectural component under relatively constrained training scales. A systematic investigation of scaling behavior—including larger training corpora, longer context lengths, and extended training schedules—remains an important next step. Second, Gengram currently employs fixed k-mer levels and predefined window sizes, motivated by biological priors and computational considerations. Although this design proves effective, exploring adaptive designs on learnable k-mer granularities or adaptive window mechanisms is a promising direction for extending Gengram's expressiveness. Finally, while our evaluation focuses on standard genomic benchmarks, translating motif-level modeling improvements to application-driven biological settings remains an important challenge. For example, assessing the impact of Gengram on tasks involving subtle genetic variations will require closer integration with downstream biological and experimental validation.

Impact Statement

This paper presents work whose goal is to advance the interaction field of Machine Learning and Genomics. There are many potential societal consequences of our work in this interdisciplinary area, none of which we feel must be specifically highlighted here.

Acknowledgement

The model training process was conducted entirely on the 021 Large Science Model and Zero2X open platform. We acknowledge CycloneSEQ for providing high quality sequencing data used in this study. Thanks to the DeepSeek team for their pioneering work in the field of large language models. Their research achievements provided important inspiration for our study. We extend our special thanks to Cheng Wang and Huijun Shen for their significant contributions. Additionally, we appreciate all participants who provided assistance during the research process.

References

[1] Kristin D Kernohan and Kym M Boycott. The expanding diagnostic toolbox for rare genetic diseases. *Nature Reviews Genetics*, 25(6):401–415, 2024.

[2] Zhao Yang, Bing Su, Chuan Cao, and Ji-Rong Wen. Regulatory dna sequence design with reinforcement learning. *arXiv preprint arXiv:2503.07981*, 2025.

[3] Adi Lin, Bin Xie, Cheng Ye, Cheng Wang, Duoyuan Chen, Ercheng Wang, Fanfeng Lu, Guirong Xue, Haiqiang Zhang, Jiajie Zhan, et al. Genos: a human-centric genomic foundation model. *GigaScience*, 14:giaf132, 2025.

[4] Alexandros Tzanakakis, Ioannis Mouratidis, and Ilias Georgakopoulos-Soares. Fundamental limitations of genomic language models for realistic sequence generation. *bioRxiv*, pages 2026–01, 2026.

[5] Xin Cheng, Wangding Zeng, Damai Dai, Qinyu Chen, Bingxuan Wang, Zhenda Xie, Kezhao Huang, Xingkai Yu, Zhewen Hao, Yukun Li, et al. Conditional memory via scalable lookup: A new axis of sparsity for large language models. *arXiv preprint arXiv:2601.07372*, 2026.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[7] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. In *First conference on language modeling*, 2024.

[8] 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.

[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] Sam Boshar, Benjamin Evans, Ziqi Tang, Armand Picard, Yanis Adel, Franziska K Lorbeer, Chandana Rajesh, Tristan Karch, Shawn Sidbon, David Emms, et al. A foundational model for joint sequence-function multi-species modeling at scale for long-range genomic prediction. *bioRxiv*, pages 2025–12, 2025.

[11] Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range dna sequence modeling, 2024. *URL https://arxiv. org/abs/2403.03234*.

[12] Wen-Wei Liao, Mobin Asri, Jana Ebler, Daniel Doerr, Marina Haukness, Glenn Hickey, Shuangjia Lu, Julian K Lucas, Jean Monlong, Haley J Abel, et al. A draft human pangenome reference. *Nature*, 617(7960):312–324, 2023.

[13] Tamara Goldfarb, Vamsi K Kodali, Shashikant Pujar, Vyacheslav Brover, Barbara Robbertse, Catherine M Farrell, Dong-Ha Oh, Alexander Astashyn, Olga Ermolaeva, Diana Haddad, et al. Ncbi refseq: reference sequence standards through 25 years of curation and annotation. *Nucleic Acids Research*, 53(D1):D243–D257, 2025.

[14] Katarína Grešová, Vlastimil Martinek, David Čechák, Petr Šimeček, and Panagiotis Alexiou. Genomic benchmarks: a collection of datasets for genomic sequence classification. *BMC Genomic Data*, 24(1):25, 2023.

[15] Evan Trop, Yair Schiff, Edgar Mariano Marroquin, Chia Hsiang Kao, Aaron Gokaslan, McKinley Polen, Mingyi Shao, Bernardo P de Almeida, Thomas Pierrot, Yang I Li, et al. The genomics long-range benchmark: Advancing dna language models. 2024.

[16] Stephen M Mount. A catalogue of splice junction sequences. *Nucleic acids research*, 10(2):459–472, 1982.

[17] Francis HC Crick. The origin of the genetic code. *Journal of molecular biology*, 38(3):367–379, 1968.

[18] Stephen T Smale and James T Kadonaga. The rna polymerase ii core promoter. *Annual review of biochemistry*, 72(1):449–479, 2003.

[19] Wei Wu, Qiuyi Li, Mingyang Li, Kun Fu, Fuli Feng, Jieping Ye, Hui Xiong, and Zheng Wang. Generator: a long-context generative genomic foundation model. *arXiv preprint arXiv:2502.07272*, 2025.

[20] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. What does bert learn about the structure of language? In *ACL 2019-57th Annual Meeting of the Association for Computational Linguistics*, 2019.

[21] Jesse Vig and Yonatan Belinkov. Analyzing the structure of attention in a transformer language model. *arXiv preprint arXiv:1906.04284*, 2019.

[22] 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.

[23] Oscar Skean, Md Rifat Arefin, Dan Zhao, Niket Patel, Jalal Naghiyev, Yann LeCun, and Ravid Shwartz-Ziv. Layer by layer: Uncovering hidden representations in language models. *arXiv preprint arXiv:2502.02013*, 2025.

[24] Daniel S Margulies, Satrajit S Ghosh, Alexandros Goulas, Marcel Falkiewicz, Julia M Huntenburg, Georg Langs, Gleb Bezgin, Simon B Eickhoff, F Xavier Castellanos, Michael Petrides, et al. Situating the default-mode network along a principal gradient of macroscale cortical organization. *Proceedings of the National Academy of Sciences*, 113(44):12574–12579, 2016.

[25] James C Wang. Helical repeat of dna in solution. *Proceedings of the National Academy of Sciences*, 76(1):200–203, 1979.

[26] Erwin Chargaff. Chemical specificity of nucleic acids and mechanism of their enzymatic degradation. *Experientia*, 6(6):201–209, 1950.

[27] Karolin Luger, Armin W Mäder, Robin K Richmond, David F Sargent, and Timothy J Richmond. Crystal structure of the nucleosome core particle at 2.8 å resolution. *Nature*, 389(6648):251–260, 1997.

[28] James C Wang. Dna topoisomerases. *Annual review of biochemistry*, 65(1):635–692, 1996.

[29] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Dengr, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.

[30] Joshua Ainslie, James Lee-Thorp, Michiel De Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. *arXiv preprint arXiv:2305.13245*, 2023.

[31] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. St-moe: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

[32] Anzhe Cheng, Shukai Duan, Shixuan Li, Chenzhong Yin, Mingxi Cheng, Heng Ping, Tamoghna Chattopadhyay, Sophia I Thomopoulos, Shahin Nazarian, Paul Thompson, et al. Ermoe: Eigen-reparameterized mixture-of-experts for stable routing and interpretable specialization. *arXiv preprint arXiv:2511.10971*, 2025.

[33] Bernhard Haubold and Thomas Wiehe. How repetitive are genomes? *BMC bioinformatics*, 7(1):541, 2006.

[34] Wentian Li, Jan Freudenberg, and Pedro Miramontes. Diminishing return for increased mappability with longer sequencing reads: implications of the k-mer distributions in the human genome. *BMC bioinformatics*, 15(1):2, 2014.

[35] Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, et al. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *arXiv preprint arXiv:2505.06708*, 2025.

[36] Nostalgebraist. Interpreting gpt: The logit lens, 2020. Accessed: 2026-01-29.

A Data Collection and Processing

A.1 Data Collection

The training dataset for this study consists of 145 high-quality haplotype-resolved assemblies, including both human and non-human primate genomes. Human sequences were primarily sourced from the Human Pangenome Reference Consortium (HPRC, release2), supplemented by the GRCh38 and CHM13 reference genomes. Non-human primate sequences were integrated from the NCBI RefSeq database to incorporate evolutionary diversity.

A.2 Tokenization and Encoding

All sequences were processed using one-hot encoding. The vocabulary includes the four canonical bases (A, T, C, G), the ambiguous nucleotide N, and the end-of-document token <EOD>.

A.3 Two-Stage Data Processing Pipeline

A.3.1 Stage I: Base Pre-training (8k Context)

In the first stage, Datasets 1 and 2 were processed into 8,192 bp fragments 4,5. To focus on functional representations, we filtered out fragments positioned beyond 5,120 bp of any gene boundary. This yielded a total of 250B tokens, with a 50B subset allocated for architectural ablation studies and the remaining 200B for formal pre-training. Sequences in this stage were restricted to the forward-strand orientation.

A.3.2 Stage II: Continued Pre-training (CPT, 32k Context)

In the CPT stage, Dataset 3 was formatted into 32,768 bp fragments 6. This stage introduced a dual-strand randomized assignment mechanism (56% negative strand and 44% positive strand) and removed the gene-distance filtering. Integrating distal intergenic regions and repetitive elements exposes the model to the full genomic landscape, enabling robust feature extraction from functionally 'negative' background sequences.

Table 4: Datasets for 1.2B model, 50B token total, 8k seq length.

CURRENT_ACCESSION	ORGANISM_NAME	CURRENT_ACCESSION	ORGANISM_NAME
HUMAN (HPRC)	*Homo sapiens*	GCA_021498455.1	*Colobus guereza*
GCA_047047975.1	*Trachypithecus germaini*	GCA_026956025.1	*Macaca cyclopis*
GCA_049640505.1	*Gorilla beringei beringei*	GCA_046862485.1	*Callithrix penicillata*
GCA_030222135.1	*Aotus nancymaae*	GCA_965153135.1	*Papio papio*
GCA_047655295.1	*Semnopithecus entellus*	GCA_023783065.1	*Nomascus siki*

Table 5: Datasets for 10B model, 200B token total, 8k seq length.

CURRENT_ACCESSION	ORGANISM_NAME	CURRENT_ACCESSION	ORGANISM_NAME
HUMAN (HPRC)	*Homo sapiens*	GCA_014849445.1	*Cercopithecus mona*
GCA_047047975.1	*Trachypithecus germaini*	GCA_031835075.1	*Saguinus oedipus*
GCF_008728515.1	*Papio anubis*	GCA_023783475.1	*Daubentonia madagascariensis*
GCF_009663435.1	*Callithrix jacchus*	GCF_009828535.3	*Hylobates moloch*
GCA_030128845.1	*Pan troglodytes*	GCA_965153135.1	*Papio papio*
GCA_000235385.1	*Saimiri boliviensis boliviensis*	GCF_008122165.1	*Gorilla gorilla gorilla*
GCA_947095605.1	*Pongo pygmaeus*	GCA_026956025.1	*Macaca cyclopis*
GCA_046862485.1	*Callithrix penicillata*	GCA_049640505.1	*Gorilla beringei beringei*
GCF_003339765.1	*Macaca mulatta*	GCA_023764695.1	*Pygathrix nigripes*
GCA_008086735.1	*Cheirogaleus medius*	GCA_048565385.1	*Saimiri boliviensis*
GCA_028878085.3	*Symphalangus syndactylus*	GCF_041146395.1	*Eulemur rufifrons*
GCF_024542745.1	*Macaca thibetana thibetana*	GCA_023807365.1	*Macaca silenus*
GCF_009764315.1	*Trachypithecus francoisi*	GCA_047655295.1	*Semnopithecus entellus*
GCA_023783065.1	*Nomascus siki*	GCF_000165445.2	*Microcebus murinus*
GCA_040869165.1	*Theropithecus gelada*	GCA_030222135.1	*Aotus nancymaae*
GCA_023783095.1	*Macaca assamensis*	GCA_040437455.1	*Plecturocebus cupreus*
GCA_047496115.1	*Brachyteles arachnoides*	GCA_023783575.1	*Cebus albifrons*
GCF_012559485.2	*Macaca fascicularis*	GCA_047834645.1	*Nasalis larvatus*
GCA_021498455.1	*Colobus guereza*		

Table 6: Datasets for 10B model, 100B token total, 32k seq length..

CURRENT_ACCESSION	ORGANISM_NAME	CURRENT_ACCESSION	ORGANISM_NAME
HUMAN(HPRC)	*Homo sapiens*	GCA_003339765.3	*Macaca mulatta*
GRCH38	*Homo sapiens*	GCA_013052645.3	*Pan paniscus*
GCA_002880755.3	*Pan troglodytes*	GCA_023764695.1	*Pygathrix nigripes*
GCF_028878055.3	*Symphalangus syndactylus*	GCA_007565055.1	*Rhinopithecus roxellana*
GCF_030222135.1	*Aotus nancymaae*	GCA_023767775.1	*Pongo pygmaeus*
GCA_028878055.3	*Symphalangus syndactylus*	GCA_965153045.1	*Macaca nemestrina*
GCF_003255815.1	*Theropithecus gelada*	GCF_041146395.1	*Lemur catta*
GCA_021498475.1	*Saguinus midas*	GCA_023761135.1	*Symphalangus syndactylus*
GCA_903645265.1	*Macaca fascicularis*	GCA_012559485.3	*Macaca fascicularis*

Table 7: Benchmarks Datasets.

	Task	Source	Length (bp)	Classifier
Genomic Structure Understanding	demo_coding_vs _intergenomic_seqs	GB	200	MLP
	splice_sites_all	NTB	400	MLP
	human_genomic_element	GeB	up to 1024	MLP
	mouse_genomic_element	GeB	up to 1024	MLP
	multi_species_ exon_classification	GeB	1024	MLP
	multi_species_ three_prime_utr	GeB	1024	MLP
	multi_species_ five_prime_utr	GeB	1024	MLP
Gene Regulation Prediction	regulatory_element_ promoter_8K	LRB	8192	MLP
	regulatory_element_ enhancer_8K	LRB	8192	MLP
	human_enhancers_cohn	GB	500	MLP
	human_ocr_ensembl	GB	up to 593	MLP
Variant Effect and Clinical Impact	variant_effect_ causal_eqtl_8K	LRB	8192	MLP
	CPC_8192	GeB	8192	MLP
	CPC_32768	GeB	32768	MLP
Epigenetic Profiling	H3	NTB	up to 500	MLP
	H3K36me3	NTB	up to 500	MLP
Evolutionary Analysis	primate_mammal_species_ classification_8192	GeB	8192	XGBoost
	primate_mammal_species_ classification_32768	GeB	32768	XGBoost

Table 8: Comparison of various GFMs.

Model	Total Parameters	Activated Parameters	Trainned Tokens	Seq. Length	Tokenizer	Dataset
GENGRAM-10B-L3_6_10-W21-8K	10B	2.87B	200B	8K	Single Base	HPRC, RefSeq Database
GENOS-10B	10B	2.87B	2.2T	1M	Single Base	HPRC,HGSVC, CEPH,GRCH38, CHM13
EVO2-40B	40.3B	40.3B	9.3T	1M	Single Base	OpenGenome2
NTV3-PRETRAIN	650M	650M	11T	1M	Single Base	OpenGenome2
GENERATOR-3B	3B	3B	386B	98K	6-mer	RefSeq eukaryotic genomes

Table 9: MHA-based Gengram Configurations (Dense vs. MoE)

PARAMETER	0.3B DENSE (BASE/ENGRAM)	1.2B MOE (BASE/ENGRAM)
TOTAL PARAMETERS	0.3B	1.2B
ACTIVATED PARAMETERS	0.3B	
HIDDEN SIZE	1280	1024
NUM LAYERS	12	
ATTENTION HEADS	16	
NORMALIZATION	RMSNORM	
ACTIVATION	SWIGLU	
POSITIONAL EMB	ROPE	
NUM EXPERTS	\	8
ROUTER TOP-K	\	2
MOE OPERATORS	\	GROUPED GEMM
LOAD BALANCING	\	AUX LOSS
GENGRAM LAYERS	OPTIONAL {3, 6, 10}	
GENGRAM NGRAM-SIZES	OPTIONAL (1 TO 6)	
GENGRAM WINDOW-SIZES	OPTIONAL (21)	
GENGRAM CONV-KERNEL-SIZE	OPTIONAL (4)	
GENGRAM EMBED-DIM-PER-NGRAM	OPTIONAL (1024)	
SEQ LENGTH	8192	
GLOBAL BATCH SIZE	1024	
LEARNING RATE	1E-4 TO 1E-5	
LR SCHEDULER	COSINE DECAY	

Table 10: MLA-based Gengram Configurations (Dense vs. MoE)

PARAMETER	0.3B DENSE (BASE/ENGRAM)	1.2B MOE (BASE/ENGRAM)
TOTAL PARAMETERS	0.3B	1.2B
ACTIVATED PARAMETERS	0.3B	
HIDDEN SIZE	1280	1024
NUM LAYERS	12	
ATTENTION HEADS	16	
KV-LORA RANK	512	256
QK HEAD DIM	128	64
QK POS-EMB DIM	128	64
V HEAD DIM	256	128
NORMALIZATION	RMSNORM	
ACTIVATION	SWIGLU	
POSITIONAL EMB	ROPE	
NUM EXPERTS	\	8
ROUTER TOP-K	\	2
MOE OPERATORS	\	GROUPED GEMM
LOAD BALANCING	\	AUX LOSS
GENGRAM LAYERS	OPTIONAL {3, 6, 10}	
GENGRAM NGRAM-SIZES	OPTIONAL (1 TO 6)	
GENGRAM WINDOW-SIZES	OPTIONAL (21)	
GENGRAM CONV-KERNEL-SIZE	OPTIONAL (4)	
GENGRAM EMBED-DIM-PER-NGRAM	OPTIONAL (1024)	
SEQ LENGTH	8192	
GLOBAL BATCH SIZE	1024	
LEARNING RATE	1E-4 TO 1E-5	
LR SCHEDULER	COSINE DECAY	

Table 11: GQA-based Gengram Configurations (Dense vs. MoE)

PARAMETER	0.3B DENSE (BASE/ENGRAM)	1.2B MOE (BASE/ENGRAM)
TOTAL PARAMETERS	0.3B	1.2B
ACTIVATED PARAMETERS	0.3B	
HIDDEN SIZE	1280	1024
NUM LAYERS	12	
ATTENTION HEADS	16	
GQA QUERY GROUPS	8	
NORMALIZATION	RMSNORM	
ACTIVATION	SWIGLU	
POSITIONAL EMB	ROPE	
NUM EXPERTS	\	8
ROUTER TOP-K	\	2
MOE OPERATORS	\	GROUPED GEMM
LOAD BALANCING	\	AUX LOSS
GENGRAM LAYERS	OPTIONAL {3, 6, 10}	
GENGRAM NGRAM-SIZES	OPTIONAL (1 TO 6)	
GENGRAM WINDOW-SIZES	OPTIONAL (21)	
GENGRAM CONV-KERNEL-SIZE	OPTIONAL (4)	
GENGRAM EMBED-DIM-PER-NGRAM	OPTIONAL (1024)	
SEQ LENGTH	8192	
GLOBAL BATCH SIZE	1024	
LEARNING RATE	1E-4 TO 1E-5	
LR SCHEDULER	COSINE DECAY	

Table 12: 10B model with and without Gengram

PARAMETER	BASELINE-10B	GENGRAM-10B-L3_6_10	GENGRAM-10B-L3_6_10-W21
TOTAL PARAMETERS	10B		
ACTIVATED PARAMETERS	2.87B		
HIDDEN SIZE	4096		
NUM LAYERS	12		
ATTENTION HEADS	16		
GQA QUERY GROUPS	8		
NORMALIZATION	RMSNORM		
ACTIVATION	SWIGLU		
POSITIONAL EMB	ROPE		
NUM EXPERTS	8		
ROUTER TOP-K	2		
MOE OPERATORS	GROUPED GEMM		
LOAD BALANCING	AUX LOSS		
GENGRAM LAYERS	\	3, 6, 10	3, 6, 10
GENGRAM NGRAM-SIZES	\	1 TO 6	
GENGRAM WINDOW-SIZES	\	\	21
GENGRAM CONV-KERNEL-SIZE	\	4	
GENGRAM EMBED-DIM-PER-NGRAM	\	1024	
SEQ LENGTH	8192 / 32768		
GLOBAL BATCH SIZE	1024		
LEARNING RATE	1E-4 TO 1E-5		
LR SCHEDULER	COSINE DECAY		

Table 13: MoE-1.2B Configurations For Layer and Window

PARAMETER	1.2B-SELECTED-LAYERS	1.2B-SELECTED-WINDOWS
TOTAL PARAMETERS	1.2B	
ACTIVATED PARAMETERS	0.3B	
HIDDEN SIZE	1024	
NUM LAYERS	12	
ATTENTION HEADS	16	
GQA QUERY GROUPS	8	
NORMALIZATION	RMSNORM	
ACTIVATION	SWIGLU	
POSITIONAL EMB	ROPE	
NUM EXPERTS	8	
ROUTER TOP-K	2	
MOE OPERATORS	GROUPED GEMM	
LOAD BALANCING	AUX LOSS	
GENGRAM LAYERS	L $\in$ [1, 12]	10
GENGRAM NGRAM-SIZES	1 TO 6	
GENGRAM WINDOW-SIZES	\	S $\in$ [6, 8,9,10,12,14,15,16, 18,20,21,22,24,26,27,28,30]
GENGRAM CONV-KERNEL-SIZE	4	
GENGRAM EMBED-DIM-PER-NGRAM	1024	
SEQ LENGTH	8192	
GLOBAL BATCH SIZE	1024	
LEARNING RATE	1E-4 TO 1E-5	
LR SCHEDULER	COSINE DECAY	

Table 14: Load Balancing Test

PARAMETER	10B MOE (ENGRAM)		
TOTAL PARAMETERS	10B		
ACTIVATED PARAMETERS	0.15625B	0.3125B	0.625B
HIDDEN SIZE	1280		
NUM LAYERS	12		
ATTENTION HEADS	16		
GQA QUERY GROUPS	8		
NORMALIZATION	RMSNORM		
ACTIVATION	SWIGLU		
POSITIONAL EMB	ROPE		
NUM EXPERTS	128	64	32
ROUTER TOP-K	2		
LOAD BALANCING	AUX LOSS		
GENGRAM LAYERS	{3, 6, 10}		
GENGRAM NGRAM-SIZES	1 TO 6		
GENGRAM WINDOW-SIZES	/		
GENGRAM CONV-KERNEL-SIZE	4		
GENGRAM EMBED-DIM-PER-NGRAM	1024		
SEQ LENGTH	8192		
GLOBAL BATCH SIZE	1024		
LEARNING RATE	1E-4 TO 1E-5		
LR SCHEDULER	COSINE DECAY		

B Biological interpretability of the hash table

We further probe the interpretability of the 3-gram hash table learned by Gengram. Because DNA is double-stranded, signals tied to structure/accessibility are expected to be approximately invariant under reverse complementation (RC). Meanwhile, if the table behaves like a motif dictionary, codon representations would tend to collapse by amino-acid semantics (i.e., synonymous codons cluster together). We therefore examine the hash table from two complementary angles: RC symmetry and synonymous-codon clustering consistency, to determine whether it functions as a reusable motif memory rather than a static motif dictionary.

For each layer, we enumerate all 64 codons and construct the reverse complement $rc(c)$ for each codon c. We compute the cosine distance $d(c, rc(c))$ in representation space. Using a cluster map obtained by clustering codon embeddings, we categorize each RC pair as:

within: c and $rc(c)$ fall into the same cluster, so $d(c, rc(c))$ contributes to the within distribution;

between: otherwise, it contributes to the between distribution.

We compare within vs. between using a Mann–Whitney U test (as implemented in your script and visualized by the violin plots). If RC equivalence is learned, RC pairs should be more "cluster-aligned," i.e., within distances become significantly smaller than between.

The RCwithin vs RCbetween comparison reveals a clear layer-wise pattern. In Layer3 and Layer10, within distances are generally smaller than between distances, with statistically significant separation (marked as ** and * in the Figure 9) left, while in Layer6 the difference is not significant (ns). This suggests that the 3-gram memory does not enforce RC equivalence uniformly across layers; instead, it exhibits a division of labor where early or deep layers are more RC-aligned, whereas the middle layer is more task-sensitive. This is biologically plausible: intermediate representations may emphasize strand-aware cues such as transcription directionality and splice donor/acceptor patterns (which are inherently not RC-invariant), while deeper layers integrate broader context and favor features tied to double-stranded structure/accessibility, leading to stronger RC pairing alignment (more "same-cluster" RC pairs and smaller within distances).

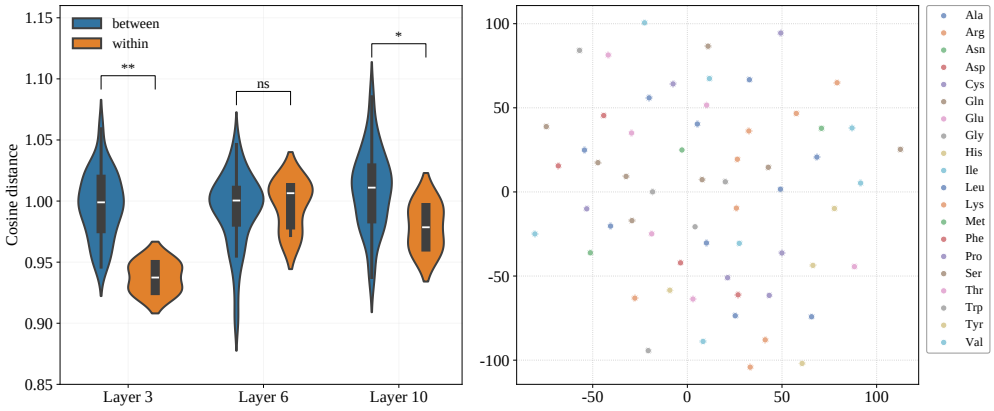

Figure 9: **RC violin (within vs between) at left:** demonstrates whether RC pairs are aligned in the learned memory space. The key takeaway is: smaller within ⇒ RC pairs tend to co-cluster and be more similar ⇒ memory captures strand/double-helix equivalence, with a meaningful layer-wise specialization (significant at L3/L10 but not at L6). **t-SNE by AA at right:** serves as a counter-evidence showing the representations do not trivially cluster by amino-acid identity, supporting "memory rather than dictionary."

We also find that synonymous codons do not systematically collapse into the same cluster. For many amino acids, their synonymous codons are distributed across multiple clusters; quantitatively, the agreement between cluster labels and amino-acid labels is weak (e.g., near-zero ARI and only moderate NMI). This behavior argues against the hash table being a static motif dictionary that compresses codons purely by amino-acid semantics. Instead, it is more consistent with a motif memory: representations jointly reflect codon usage bias, local sequence context, coupling with regulatory constraints (e.g., splice-related short motifs), and mixed coding/non-coding pressures, hence they are not fully governed by the "synonymous = equivalent" rule (Figure 9 right).