DELTAGENOME: SPARSE-EDIT INFERENCE FOR GENOMIC FOUNDATION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Genomic variant analysis repeatedly evaluates sequences that differ by one or a few nucleotides, yet genomic foundation-model inference usually treats reference and alternate alleles as independent dense inputs. DeltaGenome formulates this workload as sparse-edit inference: given a computed reference, determine which latent states and operations remain reusable after a sparse genomic edit. Across NT-v2, DNABERT-2, Caduceus, and HyenaDNA, single-nucleotide edits induce structured and predictable latent perturbations, but locality alone does not determine computational value. The decisive systems property is reuse-frontier persistence: how long an architecture preserves exact reusable dependency structure after the edit. On a frozen 384-variant benchmark, masked reconstruction preserves dense alternate perturbations at controlled selection-policy budgets, yet runtime diverges sharply. DNABERT-2 remains slower than dense inference in the tested executor, whereas HyenaDNA obtains a 1.54x warm-query speedup with a persistent full-stack causal reuse frontier and a larger reference cache. DeltaGenome establishes reuse-frontier persistence as a unifying principle for efficient inference over sparse genomic edits around a shared reference sequence.

1 INTRODUCTION

Genomic foundation models increasingly support regulatory prediction, variant-effect modeling, and genome-scale representation learning (Ji et al., 2021; Zhou et al., 2024; Dalla-Torre et al., 2025; Nguyen et al., 2023; Schiff et al., 2024). Yet variant-centered genomic inference has an unusually strong form of input redundancy: many alternate sequences share the same reference and differ at only one or a few nucleotide positions. We represent a sparse edit by Δx, with $|\operatorname{supp}(\Delta x)| \ll L$, and write the resulting alternate sequence as $x_{\mathrm{alt}} = x_{\mathrm{ref}} \oplus \Delta x$, where $\oplus$ denotes position-wise nucleotide override. Conventional inference nevertheless recomputes the full model for every alternate. Genomic variant workloads therefore occupy a distinctive regime of *extreme input similarity but potentially nonlocal computational invalidation*, making sparse-edit inference a natural systems problem.

This structure differs from ordinary autoregressive language-model serving. Prefix caching exploits a shared prefix followed by divergent continuation (Kwon et al., 2023), whereas genomic alternates are nearly identical across the full sequence and may differ at an arbitrary internal coordinate. The reusable computation after an edit is therefore architecture dependent rather than defined by a single prefix boundary. DeltaGenome asks which reference computations remain valid after Δx, and which must be recomputed. Reference-only scoring can sometimes avoid explicit ALT inference for scalar variant scores (Benegas et al., 2023); DeltaGenome addresses the complementary setting where ALT-conditioned latent states or downstream computation are required.

The central difficulty is that *perturbation locality and exact computational reuse are distinct*. A sparse edit may produce a concentrated hidden-state perturbation field, yet concentrated perturbation energy does not imply that most prior computation remains valid. Conversely, an architecture may preserve a deep exact reuse frontier even when downstream representations change substantially. We therefore separate representation-level perturbation structure from systems-level executable reuse, defining

$$\Delta h_i^{(\ell)} = h_i^{(\ell)}(x_{\mathrm{alt}}) - h_i^{(\ell)}(x_{\mathrm{ref}}),$$

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Figure 1 DeltaGenome: architecture-aware inference for sparse genomic edits

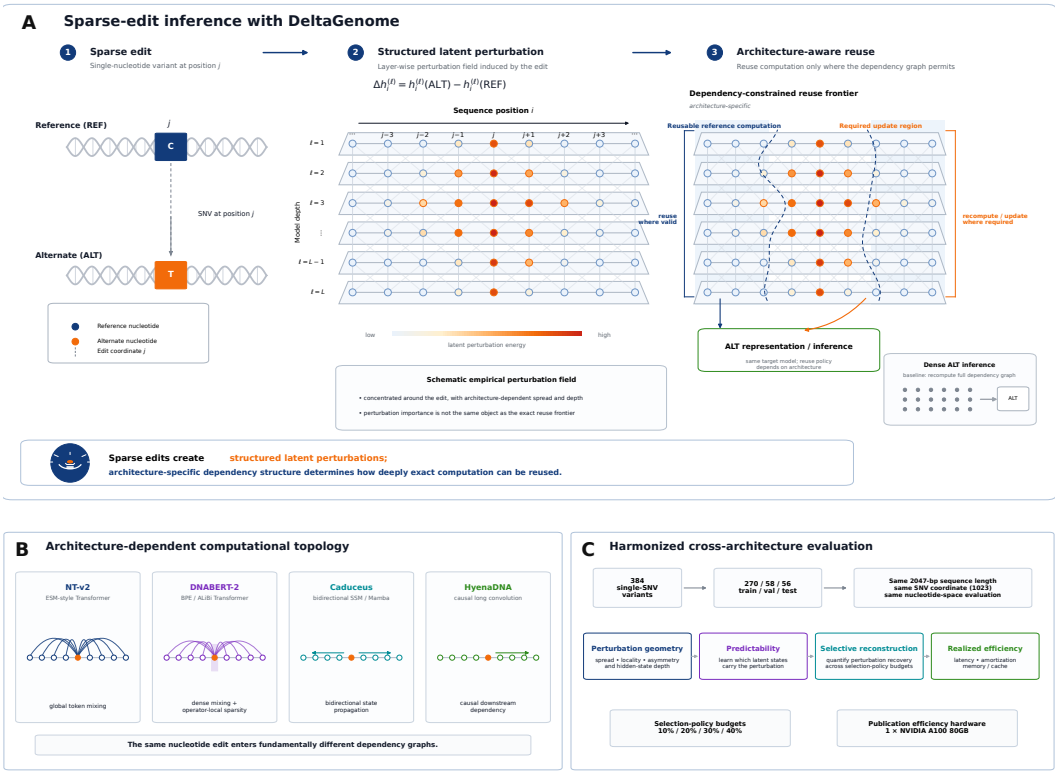


Figure 1: **DeltaGenome studies architecture-aware sparse-edit inference for genomic variants.** Panel A formulates alternate-sequence evaluation as an edit-conditioned query around reusable reference computation, separating structured latent perturbations from exact reuse frontiers. Panel B summarizes the evaluated computational topologies for NT-v2, DNABERT-2, Caduceus, and HyenaDNA. Panel C shows the frozen harmonized benchmark and analysis chain used to separate perturbation geometry, predictability, selective reconstruction, and realized efficiency.

and asking both where perturbation energy concentrates and how far exact computational validity persists through the model dependency graph.

We evaluate this distinction across NT-v2, DNABERT-2, Caduceus, and HyenaDNA. Across 384 single-nucleotide variants, sparse edits induce structured, architecture-dependent latent perturbations; their positional importance is predictable; and predictor-selected positions retain most dense REF–ALT perturbation energy under controlled selection budgets. These properties, however, do not guarantee acceleration. DNABERT-2 remains slower than dense inference in the tested executor despite strong reconstruction metrics, whereas HyenaDNA benefits from a persistent full-stack causal reuse frontier at the cost of a larger reference cache. Thus, the computational value of sparse edits depends not on locality alone, but on whether reusable dependency structure persists deeply enough to support efficient incremental execution.

The contributions of this study are fourfold. First, we formulate genomic variant inference as *edit-conditioned computation around reusable reference state*, distinguishing it from both independent dense inference and ordinary prefix caching. Second, we characterize sparse-edit perturbation geometry across representative genomic foundation-model architectures. Third, we separate perturbation-importance prediction and masked reconstruction from exact executable reuse. Fourth, we show that realized latency, memory, and amortization are governed by *reuse-frontier persistence rather than perturbation locality alone*.

2 Related Work

Genomic foundation models and variant workloads. DNA language models extend self-supervised sequence modeling to genomic inputs, from DNABERT and DNABERT-2 to Nucleotide Transformer, HyenaDNA, and Caduceus (Ji et al., 2021; Zhou et al., 2024; Dalla-Torre et al., 2025; Nguyen et al., 2023; Schiff et al., 2024). Sequence models are also used for regulatory and variant-effect prediction, including Enformer and genome language models such as GPN (Avsec et al., 2021; Benegas et al., 2023). DeltaGenome is orthogonal to model development and downstream biological validation: it studies the repeated-inference pattern in which many sparse alternates share a reference.

Reference-relative variant scoring. Some zero-shot variant-effect methods can derive a scalar score from a reference-context prediction or allele log-likelihood ratio without constructing a full alternate-sequence representation (Benegas et al., 2023). Such reference-relative scoring can be highly efficient when the required output is only a scalar variant score. DeltaGenome studies the complementary workload in which alternate-conditioned latent states or downstream computation are required. DeltaGenome targets workloads that explicitly require ALT-conditioned inference beyond reference-only scoring.

Sparse and structure-aware computation. Efficient sequence models exploit computational structure through restricted attention, conditional computation, or hardware-aware kernels (Child et al., 2019; Beltagy et al., 2020; Zaheer et al., 2020; Fedus et al., 2022; Dao et al., 2022). DeltaGenome uses a different source of structure: sparsity arises between a reference and a sparsely edited alternate rather than within one input, so the relevant question is which prior computation remains valid after an edit.

State reuse and serving. Language-model serving systems reuse state through prefix caching and KV-cache management (Kwon et al., 2023). Sparse genomic workloads are not prefix-sharing workloads: the reference and alternate are identical almost everywhere, but tokenization and architecture determine how far the edit invalidates latent computation. DeltaGenome studies edit-conditioned reuse frontiers rather than prefix reuse alone.

3 DeltaGenome

3.1 Sparse-edit genomic inference

Let $x_{\mathrm{ref}} \in \Sigma^L$ be a reference sequence over the nucleotide alphabet $\Sigma = \{\mathrm{A}, \mathrm{C}, \mathrm{G}, \mathrm{T}\}$. We represent a position-wise edit as

$$\Delta x \in (\Sigma \cup \{\perp\})^L,$$

where $\Delta x_i = \perp$ denotes no edit at position i, and $\Delta x_i \in \Sigma$ specifies the substituted nucleotide. Its support is

$$\mathrm{supp}(\Delta x) = \{i : \Delta x_i \neq \perp\}, \qquad |\mathrm{supp}(\Delta x)| \ll L.$$

The alternate sequence is obtained by position-wise override,

$$x_{\mathrm{alt}} = x_{\mathrm{ref}} \oplus \Delta x, \tag{1}$$

where

$$(x_{\mathrm{ref}} \oplus \Delta x)_i = \begin{cases} \Delta x_i, & \Delta x_i \neq \perp, \\ x_{\mathrm{ref},i}, & \text{otherwise.} \end{cases}$$

Conventional variant inference evaluates $F(x_{\mathrm{ref}})$ and $F(x_{\mathrm{alt}})$ as independent dense forward passes. DeltaGenome instead treats x_{alt} as an edit-conditioned query around previously computed reference state and asks which intermediate computations remain valid after the edit and which must be recomputed.

This reuse structure is architecture dependent. Tokenization can turn one edit into multiple token changes; global mixing can invalidate distant states; bidirectional operators propagate dependence both ways; causal operators preserve pre-edit state while recomputing downstream. DeltaGenome therefore formulates a cross-architecture principle realized through architecture-specific reuse rules.

3.2 PERTURBATION GEOMETRY AND PREDICTION

For aligned nucleotide position i and layer or state index ℓ, the edit-induced perturbation is

$$\Delta h_i^{(\ell)} = h_i^{(\ell)}(x_{\mathrm{alt}}) - h_i^{(\ell)}(x_{\mathrm{ref}}), \tag{2}$$

with per-position perturbation energy

$$e_i^{(\ell)} = \left\| \Delta h_i^{(\ell)} \right\|_2^2. \tag{3}$$

All model states are projected to a common nucleotide/base-pair coordinate system before cross-architecture geometry, prediction, or reconstruction analyses. The main geometry summaries are oracle top-k perturbation-energy recall, support fractions, directional mass distribution, asymmetry, and Local16 enrichment; exact definitions are in Appendix A.1.

DeltaGenome next asks whether perturbation importance can be predicted from reference information before computing the dense alternate. For the single-SNV benchmark, let m denote the edited nucleotide coordinate and let $\ell_\star$ denote the deepest analyzed state of a backbone. The frozen predictor maps

$$g\Big(h^{(\ell_\star)}(x_{\mathrm{ref}}), m\Big) \longrightarrow \hat{\mathbf{s}} \in \mathbb{R}^L. \tag{4}$$

We train one lightweight convolutional predictor per backbone using deepest reference hidden states and six edit-relative positional features. Predictors are optimized against a standardized log-transformed perturbation-energy target, with checkpoint selection on the 58-variant validation/calibration split before the 56-variant test set is opened. At selection-policy budget b, we choose $K_b = \max(1, \lceil bL \rceil)$ positions, force inclusion of m, and fill the remainder by descending predicted score. Figure 3 evaluates predictor-profile Spearman correlation, predictor-selected perturbation-energy recall, and top-k overlap on held-out variants.

3.3 MASKED PERTURBATION RECONSTRUCTION

Figure 4 evaluates how well predictor-selected positions reconstruct the dense REF–ALT perturbation under a fixed selection budget. For policy budget b, the frozen evaluator computes $\Delta H = H(x_{\mathrm{alt}}) - H(x_{\mathrm{ref}})$ from dense hidden states, applies a binary position mask M_b, and reconstructs

$$\widehat{\Delta H}_b = M_b \odot \Delta H. \tag{5}$$

The primary fidelity readout is predictor-selected perturbation-energy recall,

$$R_{\mathrm{energy}}(b) = \frac{\|\widehat{\Delta H}_b\|_2^2}{\max(\|\Delta H\|_2^2, 10^{-12})}, \tag{6}$$

with the norm taken over the full position-by-hidden tensor. We also report relative reconstruction error and reconstruction-profile Spearman; implementation details and the recovered generator provenance are in Appendix A.5.

3.4 REUSE FRONTIERS AND RUNTIME

Selective reconstruction identifies where perturbation information is concentrated. Exact acceleration requires a stronger property: reference computation can be reused only where all edit-dependent ancestors leave the value unchanged. We call the boundary between reusable and invalidated computation the reuse frontier. In the frozen implementations, NT-v2 exposes a shallow exact reuse path, DNABERT-2 an operator-local selective reuse path, Caduceus a first-block directional reuse path, and HyenaDNA a persistent full-stack causal reuse frontier.

Formally, let the forward pass be represented by a dependency graph $G = (V, E)$. For edit e, let $I_e \subseteq V$ be the nodes whose values may change because they depend on edit-affected ancestors; the reusable set is $R_e = V \setminus I_e$, and the reuse frontier is the boundary between R_e and I_e. For a strictly causal architecture, if tokenization and positional indexing before edit coordinate j are unchanged, every state at positions $i < j$ is invariant at every layer and is therefore exactly reusable.

HyenaDNA's near-zero upstream perturbation is an empirical sanity check consistent with this dependency property, not the premise from which exact reuse is inferred.

Runtime is evaluated separately from reconstruction fidelity. Warm-query speedup excludes one-time reference-state construction,

$$S_{\text{warm}} = \frac{T_{\text{dense}}}{T_{\Delta,\text{warm}}}, \tag{7}$$

Under a fixed one-time setup cost and K-independent warm-query latency, an idealized amortization model is

$$S_{\text{amortized}}(K) = \frac{KT_{\text{dense}}}{T_{\text{setup}} + KT_{\Delta,\text{warm}}}. \tag{8}$$

Figure 5C reports empirically measured end-to-end setup-amortized speedups at each K. We report these together with persistent reference-cache size and query-time peak memory, treating unmeasured fields as unmeasured rather than inferred.

4 Experimental Setup

Benchmark and models. We use the frozen DeltaGenome V1.0 harmonized sparse-edit benchmark: 384 single-nucleotide variants split into 270 training, 58 validation, and 56 test variants. Each example has a 2,047-bp reference sequence, a matched alternate sequence, and an edit at zero-based coordinate 1,023. We evaluate NT-v2, DNABERT-2, Caduceus, and HyenaDNA, covering Transformer, BPE/ALiBi Transformer, bidirectional state-space/Mamba, and causal long-convolution architectures. Hidden states are projected to a common nucleotide/base-pair coordinate system before cross-model comparisons.

Evaluation. Figure 2 reports perturbation geometry over the frozen benchmark. Figures 3 and 4 use the 56 held-out variants to evaluate perturbation-importance prediction and masked perturbation reconstruction at 10%, 20%, 30%, and 40% selection-policy budgets. Figure 5 evaluates each architecture's frozen executable reuse path on one NVIDIA A100 80GB GPU, separating warm-query speedup from setup-amortized speedup over repeated alternates sharing a reference. DNABERT-2 uses the 30% operator-local selection policy, whereas the NT-v2, Caduceus, and HyenaDNA runtime paths are determined by architecture-specific exact reuse structure rather than a shared positional selection budget. Reported summaries are medians unless stated otherwise; variant-level dispersion for Figures 2–4 is summarized by interquartile ranges in Appendix A.4, and metrics absent from the frozen runtime path are left unmeasured.

5 Results

5.1 Sparse Edits Induce Structured but Architecture-Dependent Perturbations

Single-nucleotide edits produced structured latent perturbation fields in all four architectures (Figure 2). At the deepest evaluated state, the top 30% of positions ranked by true perturbation energy captured 0.90 of total energy in NT-v2, 0.99 in DNABERT-2, and more than 0.99 in both Caduceus and HyenaDNA. Perturbations were therefore far from uniformly distributed.

The geometry was not architecture invariant. HyenaDNA showed essentially zero upstream mass and placed approximately 0.78 downstream of the edit, consistent with its causal dependency structure. Caduceus concentrated approximately 0.65 of deepest-state mass at the edited coordinate. NT-v2 was more diffuse, requiring approximately 71% of positions to capture 99% of perturbation energy. Sparse edits thus create structure, but the structure follows model topology.

The next result shows that systems efficiency depends on structure beyond perturbation geometry alone. DNABERT-2 provides a direct counterexample to locality-as-efficiency: despite a deepest-state Local16 enrichment of 58.6, its tested executor achieves only 0.47x warm-query speedup. HyenaDNA combines maximal downstream asymmetry with the strongest measured warm speedup, but the executable advantage arises from its persistent causal reuse frontier rather than asymmetry alone.

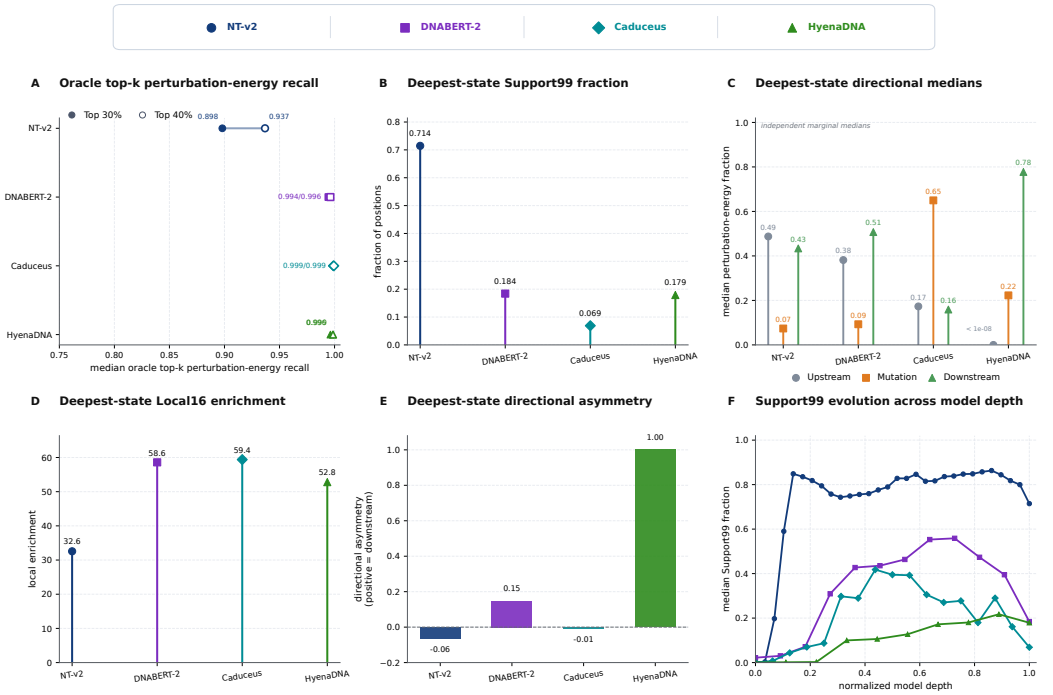


Figure 2: **Sparse edits induce structured but architecture-dependent perturbation geometry.** Panel A reports oracle top-k perturbation-energy recall from the true deepest-state energy ranking: filled and open markers denote the top 30% and top 40% of positions, respectively, not predictor-selected positions. Panel B reports Support99, the minimum fraction of positions required to capture 99% of perturbation energy. Panel C shows upstream, mutation-coordinate, and downstream perturbation-energy fractions as independently summarized medians; they are not renormalized as a closed composition. Local16 enrichment is measured in the radius-16 neighborhood around the edit, including the mutation coordinate, for up to 33 positions. HyenaDNA upstream mass is displayed as zero only after the frozen causal numerical-tolerance audit passes.

5.2 Edit-Induced Perturbation Geometry Is Predictable

Figure 3 asks whether spatial perturbation importance can be predicted before dense alternate evaluation. Across the 56 held-out variants, median predictor-profile Spearman ranged from 0.61 for NT-v2 to 0.87 for HyenaDNA. At a 30% selection budget, median predictor-selected perturbation-energy recall was 0.79 for NT-v2, 0.99 for DNABERT-2, and above 0.99 for Caduceus and HyenaDNA. Predictor top-k overlap was lower, ranging from 0.55 to 0.90, showing that predictors recovered most perturbation energy even when they did not exactly reproduce the oracle ranking.

The model ordering also cautions against treating rank agreement and retained energy as interchangeable. NT-v2 has the weakest rank agreement and the largest oracle-to-predictor energy-recall gap, whereas HyenaDNA combines near-saturated energy recall with the strongest rank agreement; the intermediate architectures do not define a strict monotonic trend.

5.3 Predicted Importance Supports Perturbation Reconstruction

At a 30% selection-policy budget, predictor-selected perturbation-energy recall was 0.79 for NT-v2, 0.99 for DNABERT-2, and more than 0.99 for Caduceus and HyenaDNA. Reconstruction-profile Spearman was 0.52, 0.70, 0.70, and 0.81, with relative reconstruction errors of 0.46, 0.09, 0.05, and 0.07. Because the mask preserves complete positional perturbation vectors, retained energy and relative error are coupled; profile correlation reports whether the spatial organization is also preserved. These metrics remain distinct from Figure 3 predictor quality and from Figure 5 execution.

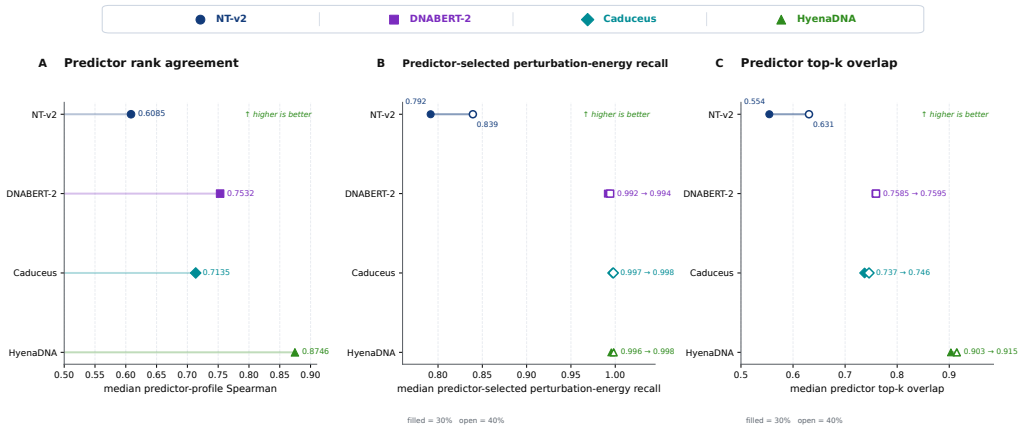


Figure 3: **Edit-induced perturbation importance is predictable on held-out variants.** Metrics are evaluated on the 56 held-out variants. Filled and open markers denote 30% and 40% selection budgets, respectively. Predictor-selected perturbation-energy recall can be high even when exact top-k overlap is lower, because the predictor may recover most perturbation energy without exactly matching the oracle ranking.

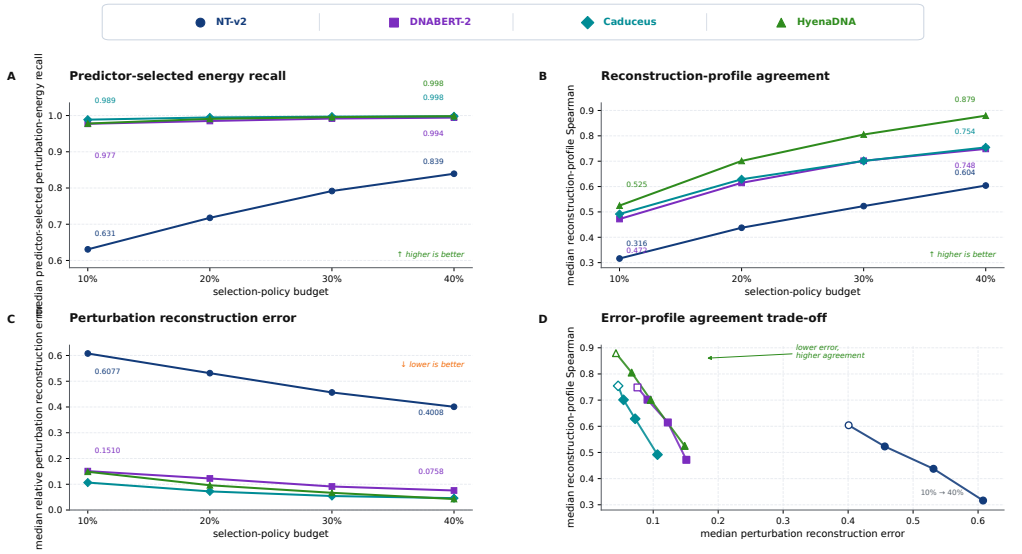


Figure 4: **Predicted importance supports masked perturbation reconstruction.** Selection-policy budgets indicate the fraction of positions retained by the reconstruction policy. Figure 4A reports the same predictor-selected perturbation-energy recall shown in Figure 3B, now interpreted within the masked reconstruction analysis. Figure 4 evaluates masked/selective perturbation reconstruction, not an actual sparse executor: dense REF–ALT perturbation vectors are retained at selected positions and masked to zero elsewhere. Under this masking, predictor-selected perturbation-energy recall and relative perturbation reconstruction error are analytically coupled, while reconstruction-profile Spearman provides complementary information about spatial agreement.

5.4 Reusable Dependency Structure Determines Realized Acceleration

Figure 5 tests whether representation-level opportunity becomes executable reuse. Warm-query speedup on one NVIDIA A100 80GB GPU was 1.02x for NT-v2, 0.47x for DNABERT-2, 1.00x for Caduceus, and 1.54x for HyenaDNA. DNABERT-2 is therefore the critical negative result: despite a

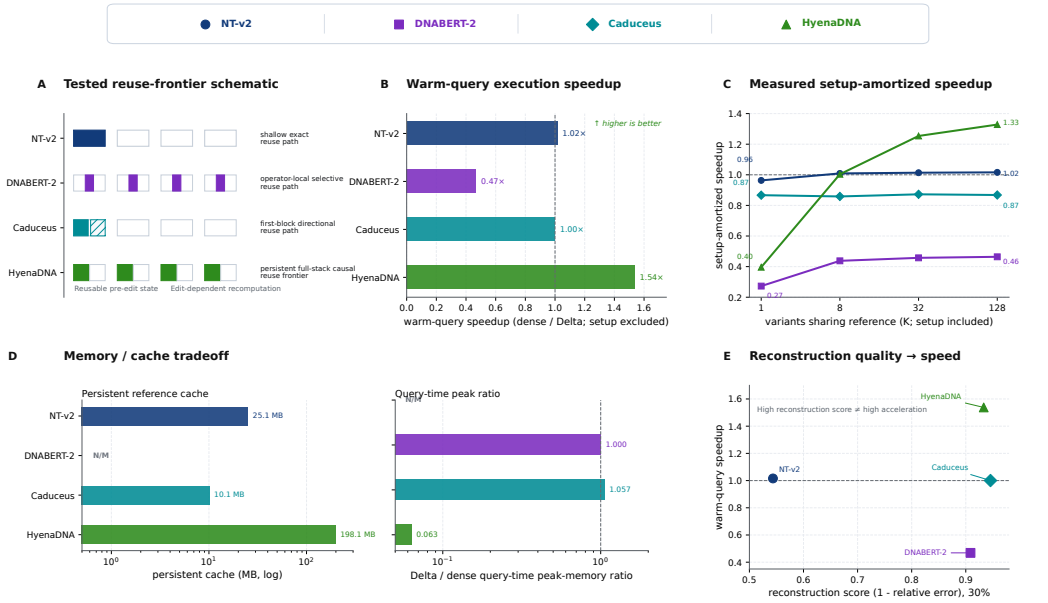


Figure 5: **Executable reuse depends on the tested reuse frontier and runtime accounting.** Panel A summarizes the tested reuse path for each implementation: shallow exact reuse path for NT-v2, operator-local selective reuse path for DNABERT-2, first-block directional reuse path for Caduceus, and persistent full-stack causal reuse frontier for HyenaDNA. DNABERT-2 uses the 30% operator-local selection policy; the other tested runtime paths are architecture-derived exact reuse paths rather than a common positional-budget comparison. For HyenaDNA, the reusable state is the pre-edit prefix and the downstream suffix is recomputed. Panel B reports warm-query execution speedup and excludes setup. Panel C reports empirical measured setup-amortized speedup for each K-query workload; the Caduceus values are frozen measurements and should not be interpreted through a fixed-cost amortization mechanism unless separately validated. Panel D separates persistent reference-cache size from query-time peak-memory ratio, and unmeasured quantities remain marked as unmeasured. Panel E uses $1-$ relative perturbation reconstruction error at the 30% selection-policy budget as a reconstruction score, not biological-task fidelity.

0.09 reconstruction error at 30% selection, its tested selective execution path was slower than dense inference. Sparsity and reconstruction fidelity alone do not predict systems efficiency.

Setup amortization sharpened the same conclusion. At $K = 8$ alternates per reference, setup-amortized speedup was 1.01x, 0.44x, 0.86x, and approximately 1.00x for NT-v2, DNABERT-2, Caduceus, and HyenaDNA; at $K = 128$, the values were 1.02x, 0.46x, 0.87x, and 1.33x. HyenaDNA crossed the measured amortization threshold as reference reuse accumulated, while Caduceus and DNABERT-2 remained below parity in the measured range (idealized break-even analysis in Appendix A.8). Caduceus remains near 0.87x over the measured K range rather than approaching its approximately 1.00x warm-query parity, indicating that the measured amortized path is not well described by the simple fixed-cost model over this range.

The memory trade-off is equally important. HyenaDNA obtained the strongest warm speedup and a query-time incremental peak-memory ratio of 0.063, but required a 198.06-MB persistent reference cache. NT-v2 used a 25.08-MB cache for only a modest gain but has no frozen query-time peak-memory ratio measurement; Caduceus used 10.07 MB but stayed near parity, and DNABERT-2 has no frozen persistent-cache-size measurement for the tested path. DeltaGenome therefore identifies the systems constraint: sparse genomic edits are computationally valuable only when the architecture preserves a sufficiently deep and inexpensive reuse frontier.

6 DISCUSSION

DeltaGenome reframes genomic foundation-model inference around edit-conditioned reuse rather than independent dense evaluation of each sequence. This abstraction matches variant-centered workloads, where many alternates share a reference, but the benchmark is deliberately computational, isolating shared sparse-edit structure as a systems property.

Such reference-centered reuse arises naturally in saturation mutagenesis and clinical or cohort-scale annotation, where many candidate variants share a reference context.

The main lesson is that **perturbation locality and exact computational reuse are distinct**. Sparse edits can create concentrated and predictable latent perturbations, yet architecture-level dependencies may invalidate computation far beyond the high-energy perturbation region. DNABERT-2 makes this distinction concrete: its perturbation structure is recoverable under selective masking, yet the tested executor remains slower than dense inference, showing that nominal sparse selection does not guarantee runtime improvement. Nucleotide-level sparsity need not stay one-token-local after BPE tokenization: 30.5% of variants changed REF/ALT token count (median retokenized span, 8 bp). Coarse retokenization measures did not explain per-variant runtime, but stable coordinate-level token correspondence was associated with higher speedup (Appendix Table 1), indicating that tokenization shapes reusable structure without alone explaining the slowdown.

HyenaDNA illustrates the complementary regime. Its causal dependency structure preserves pre-edit computation across the stack and recomputes the downstream suffix, producing a persistent full-stack causal reuse frontier. The same design yields a cache/speed trade-off: strong warm-query acceleration and low incremental peak memory require a larger persistent reference cache, which becomes useful only when amortized across enough alternates.

Alongside accuracy, context length, and training efficiency, variant-centered serving should evaluate how cheaply an architecture updates internal state after sparse edits. More broadly, DeltaGenome connects genomics to structure-aware execution by exploiting structure between highly similar inputs, not only sparsity within a single input.

7 LIMITATIONS

The benchmark covers single-nucleotide variants in 2,047-bp contexts and four architectures; larger substitutions, insertions, deletions, multi-edit haplotypes, longer contexts, and other genomic foundation models may produce different perturbation geometries and reuse boundaries. The predictor and reconstruction analyses operate on latent perturbations, not downstream biological outputs: Figure 4 evaluates masked perturbation-space reconstruction from dense REF–ALT hidden states, not an executed sparse partial forward pass, and the 10–40% policy budgets are selected-position fractions rather than FLOP or wall-clock savings.

Runtime results are specific to the evaluated architecture–executor–hardware combinations and depend on tokenizer behavior, controller implementation, kernels, cache construction, memory traffic, and runtime mode. DeltaGenome therefore does not provide a universal acceleration guarantee: its claim is conditional on an architecture preserving a persistent, inexpensive reuse frontier.

Reproducibility Statement. All quantitative claims are drawn from the frozen DeltaGenome V1.0 source package, with full numerical provenance and implementation audits maintained in the project repository. Code, the frozen benchmark, and provenance artifacts will be released publicly upon acceptance.

AI Use Statement. Generative AI assisted with manuscript drafting and the organization of provenance checks. All results, numerical claims, and source-of-truth decisions derive from the frozen DeltaGenome V1.0 registry and were independently verified by the authors before submission.

REFERENCES

Ziga Avsec, Vikram Agarwal, Daniel Visentin, et al. Effective gene expression prediction from sequence by integrating long-range interactions. *Nature Methods*, 18:1196–

1203, 2021. doi: 10.1038/s41592-021-01252-x. URL https://doi.org/10.1038/s41592-021-01252-x.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. URL https://arxiv.org/abs/2004.05150. Preprint.

Gonzalo Benegas, Sanjit Singh Batra, and Yun S. Song. DNA language models are powerful predictors of genome-wide variant effects. *Proceedings of the National Academy of Sciences of the United States of America*, 120(44):e2311219120, 2023. doi: 10.1073/pnas.2311219120. URL https://doi.org/10.1073/pnas.2311219120.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. In *arXiv preprint arXiv:1904.10509*, 2019. URL https://arxiv.org/abs/1904.10509. Preprint.

Hugo Dalla-Torre, Liam Gonzalez, Javier Mendoza-Revilla, et al. Nucleotide transformer: building and evaluating robust foundation models for human genomics. *Nature Methods*, 22(2): 287–297, 2025. doi: 10.1038/s41592-024-02523-z. URL https://doi.org/10.1038/s41592-024-02523-z.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Re. FlashAttention: Fast and memory-efficient exact attention with IO-awareness. In *Advances in Neural Information Processing Systems*, volume 35, 2022. URL https://proceedings.neurips.cc/paper_files/paper/2022/hash/67d57c32e20fd0a7a302cb81d36e40d5-Abstract.html.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022. URL https://jmlr.org/papers/v23/21-0998.html.

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. doi: 10.1093/bioinformatics/btab083. URL https://doi.org/10.1093/bioinformatics/btab083.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, pp. 611–626, 2023. doi: 10.1145/3600006.3613165. URL https://doi.org/10.1145/3600006.3613165.

Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Callum Birch-Sykes, Michael Wornow, Aman Patel, Clayton Rabideau, Stefano Massaroli, Yoshua Bengio, Stefano Ermon, Stephen A. Baccus, and Christopher Re. HyenaDNA: Long-range genomic sequence modeling at single nucleotide resolution. *arXiv preprint arXiv:2306.15794*, 2023. URL https://arxiv.org/abs/2306.15794. Preprint; metadata cross-checked through PubMed record PMID: 37426456.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Re. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 28043–28078, 2023. URL https://proceedings.mlr.press/v202/poli23a.html.

Yair Schiff, Chia-Hsiang Kao, Aaron Gokaslan, Tri Dao, Albert Gu, and Volodymyr Kuleshov. Caduceus: Bi-directional equivariant long-range DNA sequence modeling. In *International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 43632–43648, 2024. URL https://proceedings.mlr.press/v235/schiff24a.html.

Manzil Zaheer, Guru Guruganesh, Kumar Avinava Dubey, Joshua Ainslie, Chris Alberti, Santiago Ontanon, Philip Pham, Anirudh Ravula, Qifan Wang, Li Yang, and Amr Ahmed. Big bird: Transformers for longer sequences. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/c8512d142a2d849725f31a9a7a361ab9-Abstract.html.

Zhihan Zhou, Yanrong Ji, Weijian Li, Pratik Dutta, Ramana V. Davuluri, and Han Liu. DNABERT-2: Efficient foundation model and benchmark for multi-species genomes. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=oMLQB4vvJY.

A Additional Method and Runtime Details

A.1 Metric Definitions

For perturbation energy $e_i = \|\Delta h_i\|_2^2$, normalized mass is $m_i = e_i / \sum_j e_j$ when total energy is positive. Oracle top-k perturbation-energy recall at budget b selects $K_b = \max(1, \lceil bL \rceil)$ positions with largest true energy and reports their summed mass. Predictor-selected perturbation-energy recall uses the same normalized mass but sums it over positions selected by the predictor-driven policy. Support fraction at threshold τ is the smallest energy-ranked prefix fraction whose cumulative mass reaches τ. Directional summaries use upstream mass $\sum_{i<m} m_i$, mutation mass m_m, and downstream mass $\sum_{i>m} m_i$. Asymmetry is $(M_{\text{down}} - M_{\text{up}})/(M_{\text{down}} + M_{\text{up}})$, with value zero when the denominator is zero. Local16 enrichment is the mass in the clipped window $[m - 16, m + 16]$ divided by that window's expected length fraction.

Profile Spearman uses finite entries only, average ranks, and returns undefined for too-short or constant inputs. Top-k overlap is the intersection size of predicted and oracle top-k sets divided by k.

A.2 Coordinate Alignment

All perturbation profiles are evaluated in nucleotide/base-pair coordinates. Native token hidden states are projected to length-L base-pair profiles using tokenizer offsets. Multi-base tokens contribute the same hidden vector to every covered base, overlapping contributions are averaged, and special tokens are excluded. DNABERT-2 BPE REF and ALT tokenizations may have different token counts, so both are projected independently before differencing. NT-v2 uses fixed multi-base spans; Caduceus and HyenaDNA use character-level genomic tokenization in the frozen geometry path.

A.3 Predictor Implementation

The implementation trains one model-specific `ConvEnergyPredictor` for each backbone. The predictor is a one-dimensional convolutional network applied position-wise to the deepest reference state after base-pair coordinate projection. It concatenates the reference hidden vector with six edit-relative positional features: signed normalized distance to the mutation, absolute normalized distance, $\exp(-|i - m|/16)$, and upstream, mutation-coordinate, and downstream indicators. Hidden features are standardized per position before concatenation with the positional features.

The architecture is `Conv1d(hidden_dim+6,96,kernel=1)`, GELU, `Conv1d(96,96,kernel=9,padding=4)`, GELU, `Conv1d(96,96,kernel=9,padding=4)`, GELU, and `Conv1d(96,1,kernel=1)`, producing one unnormalized continuous score per position. We train one predictor per genomic model with hidden dimensions 1024 for NT-v2, 768 for DNABERT-2, and 256 for Caduceus and HyenaDNA; the input channel counts are therefore 1030, 774, 262, and 262 after adding the six positional features. The target is the standardized log transform $\log(1 + e_i/\bar{e})$, with nonnegative energy clamp, mean clamp 10^{-12}, and standard-deviation clamp 10^{-5}.

Formally, for perturbation energy $e_i = \|\Delta h_i\|_2^2$, the target is $z_i = \log(1 + e_i/\bar{e})$, where $\bar{e} = \max(\text{mean}_j e_j, 10^{-12})$, followed by within-variant standardization $y_i = (z_i -$

$\mathrm{mean}(z))/\max(\mathrm{std}(z), 10^{-5})$. The predictor regresses y_i with Smooth L1 loss using AdamW, learning rate 3×10^{-4}, weight decay 10^{-4}, gradient clipping at norm 1.0, maximum 30 epochs, early-stopping patience 5, and seed 20260821. Checkpoint selection uses the average of the median 30% and 40% predictor-selected perturbation-energy recall on the 58-variant validation/calibration split; the 56 held-out test variants are evaluated only after this selection.

The selected checkpoints occurred at epochs 5 for NT-v2, 17 for DNABERT-2, 5 for Caduceus, and 22 for HyenaDNA.

A.4 Variant-Level Uncertainty Summaries

For Figures 2–4, uncertainty summaries are computed across variants and reported as median with interquartile range. The central 30% budget readouts are:

Model	Fig. 3 Spearman	Fig. 3 pred.-selected recall	Fig. 4 rel. error
NT-v2	$0.61\ [0.43, 0.72]$	$0.79\ [0.70, 0.88]$	$0.46\ [0.35, 0.55]$
DNABERT-2	$0.75\ [0.65, 0.82]$	$0.99\ [0.98, 0.99]$	$0.09\ [0.07, 0.13]$
Caduceus	$0.71\ [0.64, 0.77]$	$1.00\ [0.99, 1.00]$	$0.05\ [0.04, 0.08]$
HyenaDNA	$0.87\ [0.87, 0.88]$	$1.00\ [0.99, 1.00]$	$0.07\ [0.05, 0.08]$

Figure 5 runtime variability is reported separately because its timing measurements follow a different aggregation structure.

A.5 Figure 4 reconstruction evaluator

The reconstruction evaluator computes

$$\Delta H = H_{\mathrm{ALT}} - H_{\mathrm{REF}},$$

with per-position perturbation energy

$$e_i = |\Delta h_i|_2^2.$$

For a selection mask M_b at budget b, the masked perturbation is

$$\widehat{\Delta H}_b = M_b \odot \Delta H.$$

The relative perturbation reconstruction error is

$$\epsilon_\Delta(b) = \frac{\left|\widehat{\Delta H}_b - \Delta H\right|_2}{\max(|\Delta H|_2, 10^{-12})}.$$

When the numerical floor is inactive, complete-vector masking gives

$$\epsilon_\Delta(b) = \sqrt{1 - R_{\mathrm{energy}}(b)}.$$

The evaluator also reports Spearman agreement between the dense and masked energy profiles, together with mutation, Local16, and global scalar summaries.

A.6 Architecture-Specific Executors

The runtime benchmark evaluates the implemented reuse mechanism for each architecture. NT-v2 uses a shallow exact reuse path. DNABERT-2 uses an eager operator-local selective controller at the 30% policy budget and yields a slowdown relative to dense inference. Caduceus uses a first-block directional reuse path. HyenaDNA reuses the pre-edit prefix and recomputes the downstream suffix through the stack, creating the persistent full-stack causal reuse frontier used in the main systems result.

A.7 Runtime Methodology

Runtime measurements distinguish reference-state setup, warm alternate-query execution, setup-amortized totals, persistent reference-cache memory, standalone workspace memory, dense peak memory, delta peak memory, and peak-memory ratios. The runtime protocol specifies warmups, repeats, CUDA/eager mode, correctness thresholds, and hardware provenance.

Table 1: **DNABERT-2 tokenization structure and execution benefit.** Associations are computed across the frozen 56-variant test set under the 30% operator-local runtime policy. p values are descriptive. Stable coordinate-level token correspondence shows the strongest association with execution speedup, whereas coarse retokenization measures do not.

Structural metric	Spearman ρ	Descriptive p
Retokenized span (bp)	-0.084	0.537
Invalidated REF tokens	0.077	0.572
Stable reuse fraction	0.193	0.155
Stable coordinate token fraction	**0.344**	**0.009**
Mean active density	-0.161	0.237

A.8 Analytical Break-Even

Under a constant setup cost and constant warm-query latency, solving

$$KT_{\mathrm{dense}} = T_{\mathrm{setup}} + KT_{\Delta,\mathrm{warm}}$$

gives $K^* = T_{\mathrm{setup}}/(T_{\mathrm{dense}} - T_{\Delta,\mathrm{warm}})$ when $T_{\Delta,\mathrm{warm}} < T_{\mathrm{dense}}$. We report measured crossover points in the Results because setup and warm-query latencies are observed directly at each operating point.