GENOMECODER: NEURAL-GUIDED INFERENCE OF EXECUTABLE BIOLOGICAL PROGRAMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Many latent quantities in biology are short, executable procedures: an immune receptor records which gene segments were joined and trimmed, an expressed gene is a set of transcript isoforms, a genealogy is a sequence of local trees linked by recombination. Mechanistic models assign probabilities to such programs, while the candidate space grows combinatorially, so exact inference is affordable only in a narrow band. We present **GenomeCoder**, which trains a recognition model to construct a compact support of executable latent programs, scores that support under the mechanistic model, and, when generative parameters are unknown, learns them from observations without latent labels. Where the exact posterior is enumerable, GenomeCoder reaches the *Bayes-optimal recovery rate* in V(D)J recombination, against 0.605 for beam search, and reaches $\mathbf{0.981}$ of the splicing ceiling in 37 exact evaluations. For a fixed mechanistic model, the restricted posterior's error is exactly the posterior mass the support omits, measured exactly in V(D)J and splicing: at $7{,}000$ evaluations GenomeCoder retains $\mathbf{0.953}$ of the splicing posterior against 0.906 for beam search. The same marginal likelihood selects how much generative model updating the reads support, halving the distance to the generating parameters at matched compute. On the published IGoR benchmark (Marcou et al., 2018), GenomeCoder matches IGoR in complete-scenario recovery at mean rate $\mathbf{0.1330}$. Under the same learned generative model, its retained support matches full enumeration recovery while scoring orders of magnitude fewer programs per read.

1 INTRODUCTION

Many latent quantities in biology are short, executable procedures. An immune receptor records which gene segments were joined, trimmed and rejoined; a genealogy is a sequence of local trees linked at recombination breakpoints; an expressed gene is a choice of which transcript isoforms are present. In each case the latent object can be *executed*, applied to the biological context to produce the data we observe. We call such an object an **executable biological program** (Figure 2).

Biological knowledge often supplies the structure of a *mechanistic model* that scores such a program: a coalescent probability for a genealogy, a recombination-scenario probability for a receptor, a read model for the fragments a transcript set would produce. These models define the probabilities used for inference and therefore the posterior over latent programs. Two computational problems remain. The latent program space is combinatorial, so inference is expensive: enumerating the recombination scenarios consistent with one read averages 3.7×10^5 candidates. And some generative parameters are unknown, so they must be learned from observations.

GenomeCoder addresses both problems (Figure 1), in two distinct learning phases. *Recognition learning* fits $q_\phi(P \mid x, R)$ to find high-posterior programs efficiently, amortizing search. *Generative learning* uses posterior responsibilities to estimate those parts of the mechanistic model whose parameters are unknown, from observations alone, while mechanistic scoring is retained throughout.

The recognition model proposes legal programs, repeated draws collapse into the unique discovered support, and each unique program is scored exactly under the mechanistic model; renormalizing those scores gives a *restricted posterior*, whose entire error is the mass the support omits (§2.1). Across V(D)J, splicing and genealogy inference GenomeCoder reduces the number of exact model

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.

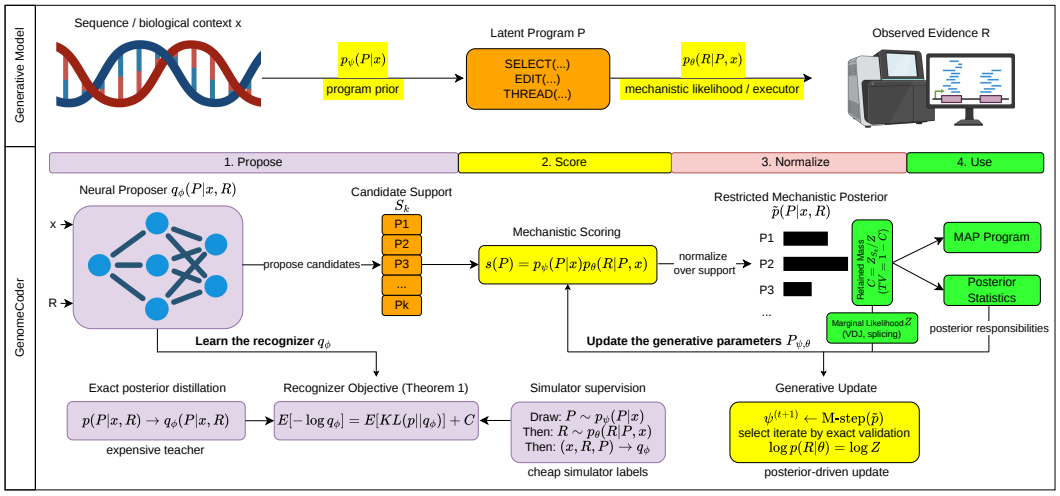


Figure 1: **The GenomeCoder inference framework.** *Top:* the generative model of equation 1. *Middle:* **propose** with q_ϕ, collapse repeated draws into the unique support S_K, **score** each unique program exactly, **normalize** those scores over S_K, and **use** the restricted posterior; for a fixed mechanistic model its approximation error is exactly the omitted mass $1 - C$, for retained mass $C = Z_{S_K}/Z$ (§3.4). *Bottom:* the two routes that train q_ϕ, which share a population optimum (Theorem 1), and the generative update, whose iterate exact validation $\log Z_{\theta,\psi}(x, R)$ selects (§3.6).

evaluations needed for latent-program recovery, learns missing generative parameters from observations alone, and matches IGoR end to end (Figure 3). Where the marginal likelihood is tractable, the omitted posterior mass can also be measured exactly. Evaluation includes controlled mechanistic comparisons, IGoR in V(D)J, the Sarantopoulou transcript-quantification benchmark in splicing, and ARGweaver in genealogy inference.

Our contributions are:

(1) **A shared mechanistic inference framework** with domain-specific proposal and normalization methods: learned candidate construction with exact mechanistic rescoring on it (§3).
(2) **Measuring approximation quality**: separating candidate discovery from mechanistic scoring lets the error from restricting inference be measured or bounded with each domain's own inference structure, and V(D)J and splicing additionally permit exact retained-posterior-mass calculation (§3.4).
(3) **Generative learning from posterior responsibilities**: generative-parameter learning from posterior responsibilities in V(D)J and splicing, without latent labels (§5.3).
(4) **Executable biological programs as an AI4Bio problem class**, instantiated by domain-specific program languages over sequential, set-valued, path-valued and supervised edit programs.

2 Problem Formulation

Every setting here shares one structure: a latent object was realized and what we observe is what it produced, so inference asks what the object was. Write x for the observed context (the germline segment set at a locus, the catalog of candidate transcripts, the sampled haplotypes), R for the evidence it generated (a read, a set of fragments, a genotype matrix), and P for the latent *executable program* that explains R. Three parameter blocks stay distinct throughout: ψ holds the program-prior parameters in $p_\psi(P \mid x)$, θ the mechanistic-likelihood parameters in $p_\theta(R \mid P, x)$, and ϕ the recognition-model parameters in $q_\phi(P \mid x, R)$. A supervised outcome y appears only in the supervised limiting case of §5.5. Depending on the experiment the generative parameters (θ, ψ) are either fixed or estimated from training observations; ϕ always parameterizes the recognition model.

The generative model is

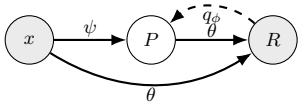
$$p_{\theta,\psi}(P, R \mid x) \;=\; p_\psi(P \mid x)\, p_\theta(R \mid P, x). \tag{1}$$

The factorization is specified by the mechanistic model; its parameter values may be known, partially known, or learned from observations. Where a domain names these blocks differently, the local convention is stated there. The target of inference is the posterior,

$$p_{\theta,\psi}(P \mid x, R) \;=\; \frac{p_\psi(P \mid x)\, p_\theta(R \mid P, x)}{Z_{\theta,\psi}(x, R)}, Z_{\theta,\psi}(x, R) \;=\; \sum_{P'} p_\psi(P' \mid x)\, p_\theta(R \mid P', x). \quad (2)$$

Its normalizing constant $Z_{\theta,\psi}(x, R)$ is the *marginal likelihood*, or evidence. It recurs as the denominator that makes posterior mass measurable and as an objective for learning (θ, ψ). We call estimating unknown components of (θ, ψ) from observations *generative-parameter learning* (system identification); it is studied in §5.3.

Solid arrows denote the **generative model**: $p_\psi(P \mid x)$ assigns probabilities to latent programs and $p_\theta(R \mid P, x)$ assigns probabilities to observations conditional on a program. The dashed arrow denotes the **recognition model** $q_\phi(P \mid x, R)$. All reported posterior quantities are recomputed under $p_{\theta,\psi}$.



2.1 Desired properties of inference

Three properties define success, and each names a measurement reported later.

Bayes-optimal latent decisions. For a single explanation the optimal choice is $P^\star_{\mathrm{MAP}} = \arg\max_P p_{\theta,\psi}(P \mid x, R)$, whose exact recovery rate is set by the evidence and fixed for any method: several programs can execute to the same observation, so a method can be Bayes-optimal while exact recovery sits well below one. Recovery is therefore read *relative to* the posterior's own rate, which is why the experiments report a Bayes ceiling, the fraction of it attained, and exact-program recovery together.

Retained posterior mass. The retained posterior mass is how much of the full posterior probability the retained support contains. For a support S_K,

$$C(S_K) \;=\; \sum_{P \in S_K} p_{\theta,\psi}(P \mid x, R), \qquad \tilde{p}(P \mid x, R, S_K) \;=\; p_{\theta,\psi}(P \mid x, R,\, P \in S_K). \quad (3)$$

Since the restricted posterior is the full posterior conditioned on S_K, its total variation distance to the full posterior is exactly $1 - C$. GenomeCoder therefore reports retained mass directly wherever Z is tractable, and Appendix C gives those measurements. Downstream posterior expectations inherit an error bound from $1 - C$, developed in §3.3.

Generative-parameter learning. The generative parameters are often the quantity of interest and are not known in advance. When some are unknown they are estimated from observations by maximizing the observed-data objective $\ell(\theta, \psi) = \sum_i \log Z_{\theta,\psi}(x_i, R_i)$, which requires no latent labels.

The three regimes. GenomeCoder is evaluated in three regimes, which the Results follow in order.

(RQ1) **Latent inference.** With (θ, ψ) fixed, can GenomeCoder recover Bayes-optimal decisions and preserve posterior mass?

(RQ2) **Generative-parameter learning.** When some generative parameters are unknown, can posterior inference support accurate learning of those parameters from observations?

(RQ3) **End-to-end inference.** After learning generative parameters from training observations, can GenomeCoder recover latent programs on held-out observations at competitive accuracy and reduced mechanistic evaluation?

3 Method

3.1 Program languages and executable latent programs

A latent program P is a typed, ordered composition of primitive actions, legal under domain constraints and executable by a mechanistic model. Each domain has its own program language (Table 1); the latent variable is sequence-valued in V(D)J, set-valued in splicing, path-valued in ARG, and an ordered edit program in structural variation. A V(D)J program is $P_{\mathrm{VDJ}} ::= \mathrm{CHOOSE}(V, D, J);\ \mathrm{TRIM}(\delta_V, \delta_{D5}, \delta_{D3}, \delta_J);\ \mathrm{INSERT}(s_{VD}, s_{DJ})$, with trims integer-valued

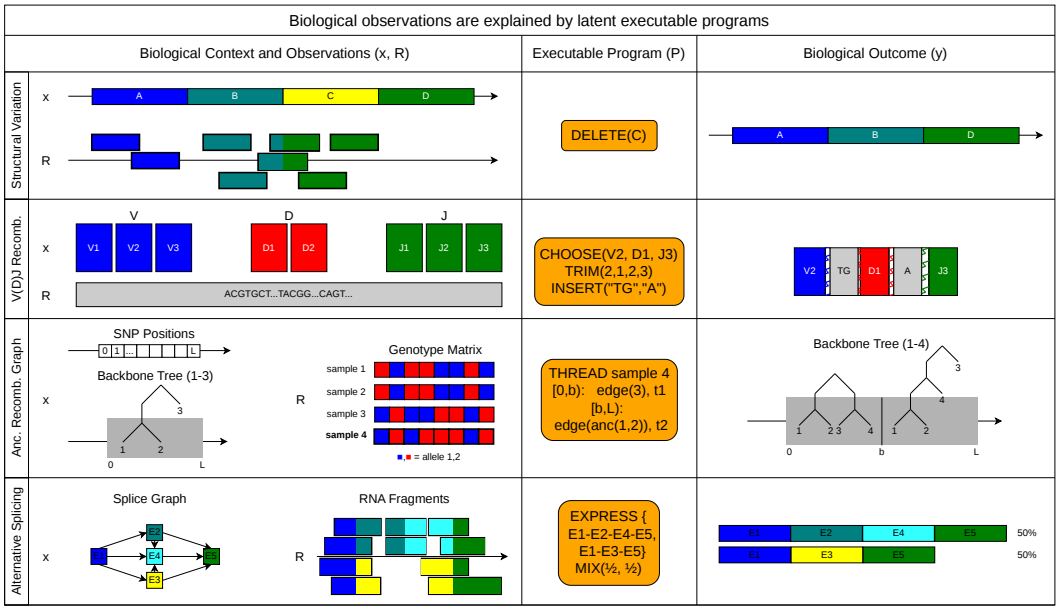


Figure 2: **Executable programs across biological settings.** Each row shows the context and evidence (x, R), the latent program P that explains them, and the outcome of executing it. The programs are short; the space of candidates consistent with the evidence is combinatorially large.

within segment-dependent ranges and insertions strings over the nucleotide alphabet. Legality is a type constraint: a TRIM applies only to a segment a preceding CHOOSE selected, and only within that segment's range. "Program" is therefore a technical description of the latent object, and each domain uses its own program language (Appendix E).

3.2 Proposal, unique support, exact scoring

The framework is shared across domains, while the proposal mechanism and the normalization method are domain-specific (Figure 1). Draw candidate programs from q_ϕ, collapse repeated draws into the unique discovered support S_K, exact-score each unique program under the mechanistic model, and normalize those mechanistic model scores over S_K,

$$\tilde{p}(P \mid x, R, S_K) \;=\; \frac{p_{\theta,\psi}(P, R \mid x)}{\sum_{P' \in S_K} p_{\theta,\psi}(P', R \mid x)}, \qquad P \in S_K, \tag{4}$$

and use that posterior for the MAP program, posterior expectations or parameter updates. The recognizer determines which programs enter S_K, and the mechanistic model determines every weight, so proposal frequencies leave no trace in equation 4: duplicate draws add no support elements once a program has been discovered, deduplication makes exact mechanistic work proportional to the number of unique hypotheses, and the approximation error is characterized by the retained mass $C(S_K)$. Appendix B derives this estimator from the proposal samples and contrasts it with the p/q importance-weighted alternative.

3.3 Posterior-statistic error from omitted mass

Exact rescoring removes score-approximation error on retained programs, so the restricted posterior of equation 3 is the full posterior conditioned on S_K, and $1-C$ is the total variation distance between the two. Use is usually an expectation, $\widehat{\mathbb{E}}[T] = \sum_{P \in S_K} \tilde{p}(P \mid x, R, S_K)\, T(P)$, of a posterior statistic T. Write $p_{S_K}(P) = p(P \mid x, R, P \in S_K)$ and $p_{S_K^c}(P) = p(P \mid x, R, P \notin S_K)$ for the full posterior conditioned on the retained and the omitted supports, so that $p = C\, p_{S_K} + (1-C)\, p_{S_K^c}$; the discrepancy has a closed form,

$$\widehat{\mathbb{E}}[T] - \mathbb{E}_p[T] \;=\; (1-C)\Big(\mathbb{E}_{p_{S_K}}[T] - \mathbb{E}_{p_{S_K^c}}[T]\Big), \tag{5}$$

the omitted mass times how differently the statistic behaves on the programs that were never scored.

Here T is vector-valued, $T : \mathcal{P} \to \mathbb{R}^d$, with the scalar case $d = 1$ included; $\|\cdot\|$ is the Euclidean norm on $\mathbb{R}^d$ and $\|T\|_\infty = \sup_P \|T(P)\|$ is the largest norm the statistic attains on any legal program. For bounded T,

$$\left\|\widehat{\mathbb{E}}[T] - \mathbb{E}_p[T]\right\| \;\le\; 2(1-C)\,\|T\|_\infty \tag{6}$$

(Appendix H). For scalar T the bound tightens to $(1-C)(\sup_P T - \inf_P T)$, which depends on the statistic's range and not its absolute scale, since both sides of equation 5 are unchanged by adding a constant to T. Either way, wherever C is computable a posterior expectation yields a computable worst-case bound on the posterior expectation error. equation 6 is that bound: realized errors are much smaller in the measured regimes, while high retained mass tracks sharply improved sufficient-statistic fidelity (Appendix H).

3.4 Measuring approximation quality

GenomeCoder always exact-scores the retained candidate set. How closely that restricted computation approximates the full posterior depends on what further computation each domain allows.

V(D)J and splicing. When the full marginal likelihood Z of equation 2 can be evaluated, the retained posterior mass is computed exactly as $C(S_K) = Z_{S_K}/Z$, where $Z_{S_K} = \sum_{P \in S_K} p_{\theta,\psi}(P, R \mid x)$ comes from the exact scores already computed. How Z becomes tractable differs between the two. In V(D)J the program sum is reorganized so that the normalizer avoids explicit scenario enumeration. In splicing the legal program set is finite and is still summed in full, while compatibility-mask regrouping compresses the repeated fragment-likelihood computation inside that sum. Both group terms of the same sum, so Z is exact in each.

V(D)J. The recombination factorization, a likelihood depending on the program only through insertion length, and finite trimming and insertion ranges together permit an inverted lookup followed by a short convolution, at $O(|s|^2)$ table lookups per read in the span length $|s|$.

Splicing. The fragment-factorized likelihood $p(R \mid S) = \prod_f \sum_{i \in S} \pi_i\, p(f \mid i)$ and a finite isoform-set space permit exact regrouping by compatibility masks, since a fragment enters the product only through the isoforms it is consistent with. Distinct masks are few (a mean of 6.1 against 271 fragments per locus), which makes Z affordable over the 3,354–21,050 programs per locus. The regrouping is exact for the evaluated fragment model because fragments sharing a mask have identical likelihood contributions under the bias-free emission model; Appendix C states what that assumption requires.

Genealogy inference. The retained object need not be a program set. In ARG, S_K is a retained set of branch-time states supplied to the exact HMM, which sums over the complete threading paths those states support; Appendix J compares the restricted and full computations and reports retained mass as a bound.

Other settings. Elsewhere, latent decisions are evaluated against an exact, Bayes-optimal or published reference method appropriate to the task.

Exactness is checked in both domains, against full enumeration to a worst relative error of 4.07×10^{-12} in V(D)J and a worst absolute difference of 4.09×10^{-12} over 132 cells in splicing. Appendix C gives the constructions and the complexity.

3.5 Posterior-free recognition learning

The scalable route trains q_ϕ from simulation alone. Draw $P \sim p_\psi(\cdot \mid x)$, execute it to obtain $R \sim p_\theta(\cdot \mid P, x)$, and train q_ϕ to recover the generating program P: one forward simulation per example, at a cost that does not grow with the posterior's support. Taking the expectation over the training joint of equation 1, the cross-entropy to the single generating program equals $\mathbb{E}_{x,R}\,\mathrm{KL}(p_{\theta,\psi}(P \mid x, R) \,\|\, q_\phi(P \mid x, R))$ up to a constant in ϕ, so its population optimum *is* the exact posterior, at the (θ, ψ) that generated the data and in the population limit (Theorem 1, Appendix A).

Exact posterior distillation where enumeration is available. Where exact enumeration is available, GenomeCoder can also train against the exact posterior, providing a direct comparison with simulator-based recognizer training and verifying that both routes converge to the same target. The gap between them is indistinguishable at 16,000 training reads, and the simulator-trained proposal

Table 1: Benchmark overview. Each latent program is a typed composition of primitives whose execution produces the observed evidence under the mechanistic model. Regimes are numbered as above; SV is the supervised limiting case.

domain	input	primitives	composition; semantics	regimes	data / split	main metric
V(D)J	60 bp read; germline V/D/J segments	CHOOSE, TRIM, INSERT	typed ordered sequence; reconstructs the read	RQ1, RQ2, RQ3	IGoR model (Marcou et al., 2018); 120 enumerable-posterior reads (RQ1); 1k/5k/19.54k reads (RQ2); five 5,000/400 splits (RQ3)	complete-scenario recovery; fraction of Bayes-optimal rate; parameter TV (RQ2)
splicing	transcripts; RNA fragments	EXPRESS(path), MIX(weight)	finite set with a mixture; induces fragment probabilities	RQ1, RQ2	25 loci $\times$ 4 priors $\times$ 5 depths $\times$ 3 seeds; 3,354–21,050 programs; external (Sarantopoulou et al., 2021)	set recovery vs Bayes-optimal rate; retained mass; Spearman
genealogy	genotypes and genealogy	THREAD(edge, time, interval)	piecewise path along the sequence; attaches a lineage to the backbone genealogy	RQ1	ARGweaver at n=12 (Rasmussen et al., 2014)	Robinson–Foulds tree error; wall-clock time
SV	aligned long reads; reference sequence	DEL, DUP, INV, INS	ordered edit program; transforms the locus	supervised	complex-SV benchmark (Lin et al., 2022); train chr1–3, test chr4–5; 350 held-out variants	exact operation-multiset recovery; operation F1

still retains 0.56 of posterior mass at K=1024 beyond the range where exact posterior enumeration is affordable (Appendix L).

3.6 Generative learning from posterior responsibilities

When generative parameters are unknown, GenomeCoder updates them using posterior responsibilities. The observed-data objective is $\log Z_{\theta,\psi}(x,R) = \log \sum_P p_\psi(P \mid x)\, p_\theta(R \mid P,x) = \log p_{\theta,\psi}(R \mid x)$, computable on observations alone. We run expectation-maximization from posterior responsibilities, evaluate the exact validation marginal likelihood $\log Z_{\theta,\psi}$ after each M-step on a validation subset of the training observations, and retain the iterate with the highest value. This criterion reads observed data only, never latent-program labels. In V(D)J and the splicing prior-learning study the learned generative parameters lie in $p_\psi(P \mid x)$, while $p_\theta(R \mid P,x)$ is fixed.

4 Experimental design

Three latent-inference domains exercise the framework of §3.2, and a fourth setting is the supervised limiting case. We evaluate three regimes: **(RQ1) latent-program recovery**, with the generative parameters fixed; **(RQ2) generative-parameter recovery**, where unknown parameters are estimated from observations; and **(RQ3) end-to-end learning**, where the generative model is learned from training observations and latent programs are recovered on held-out observations. Table 1 states, for each domain, the input, the program and what it means biologically, the regimes evaluated, the data and split, and the metric reported.

V(D)J. IGoR (Marcou et al., 2018) enumerates the scenarios that could have produced a read and fits the generative parameters by expectation-maximization; it is the comparator. Two roles stay separate: the oracle-conditioned setting, where the parameters are known and the exact posterior gives a Bayes-optimal recovery rate per read, and the reads-only benchmark, where neither system sees them (Appendix H).

Splicing. A fragment is often compatible with several transcripts, which is what makes the isoform set latent. In the controlled benchmark every method receives the same candidates, prior, reads and likelihood, so differences are attributable to search alone; the external benchmark instead places GenomeCoder against published quantification tools (Appendix K).

Genealogy inference. Recombination lets the threading path change along the sequence, and the genotype matrix is scored by the sequentially Markov coalescent (McVean & Cardin, 2005; Marjoram & Wall, 2006) with the phylogenetic likelihood (Felsenstein, 1981). Attachment points are marginalized by an exact hidden Markov model; ARGweaver is the reference method (Appendices J, F).

Structural variation. The target program is observed during training, so no posterior is formed. The primary comparison is a matched-capacity model sharing our trunk, evidence, chromosomes, optimizer and parameter count, differing only in *what it predicts*.

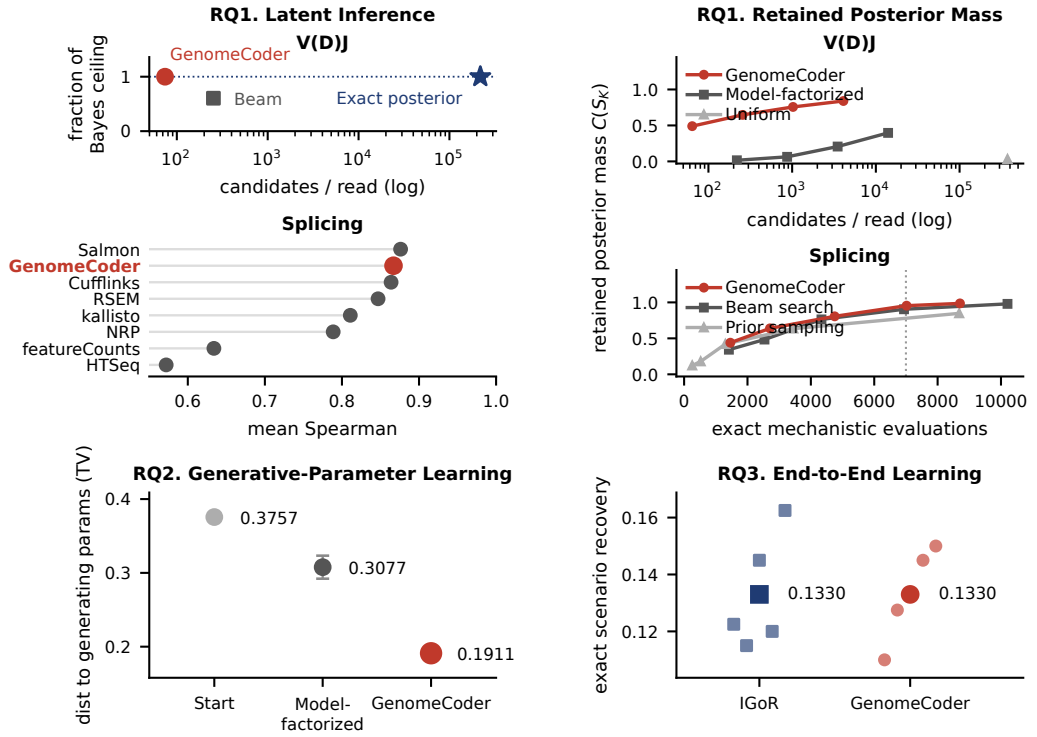


Figure 3: **GenomeCoder across the inference pipeline.** *Top-Left:* controlled V(D)J latent inference, and the named external splicing benchmark where GenomeCoder ranks second of eight methods on the realistic regime. *Top-Right:* retained posterior mass under matched mechanistic models in V(D)J and splicing. *Bottom-Left:* generative-parameter learning at a matched exact-evaluation budget. *Bottom-Right:* end-to-end recovery against IGoR over five independently regenerated splits. Full frontiers, exact-normalizer validation, short-read results, compute accounting and the domain-specific comparisons are in Appendices H, C, I, J and K.

Baselines and compute accounting. For latent inference we compare GenomeCoder with non-neural search baselines (beam, greedy restarts, prior sampling, a model-factorized sampler) and an exact or published reference, using the same exact-evaluation budget where applicable. Recovery is read relative to the Bayes-optimal recovery rate (§2.1): in the V(D)J oracle-conditioned setting the exact posterior identifies the generating program 25% of the time. Compute is counted in *exact mechanistic evaluations* (Appendix I).

5 Results

5.1 Latent Inference

V(D)J. With the generative model known, so search quality is isolated, GenomeCoder recovers the generating program as often as the exact posterior does, from 74 constructed candidates per read. We call the exact posterior's MAP recovery rate the *Bayes-optimal recovery rate*, or Bayes ceiling; beam search reaches 0.605 of it (Figure 3, top left).

Splicing. On the realistic benchmark of Sarantopoulou et al. (2021) GenomeCoder reaches $\mathbf{0.8669}$ mean Spearman, second of eight and behind Salmon at 0.8761 (Figure 3, top left); on the idealized data it is 3rd of eight at $\mathbf{0.9502}$. GenomeCoder uses Salmon equivalence classes only as the upstream read-compatibility representation and performs its own downstream mechanistic inference (Appendix K.4).

ARG. On the published small n genealogy protocol (Rasmussen et al., 2014) GenomeCoder attains $\mathbf{0.4285}$ RF, within 0.0139 of ARGweaver, in $\mathbf{7.95}$ s per replicate against 806 s for ARGweaver's

full protocol at n=12. Under controlled search GenomeCoder needs 37 exact evaluations where beam needs 100, and reaches $\mathbf{0.9302}$ of the harder ceiling against 0.6512 (Appendices J, K).

5.2 Retained posterior mass

Where the normalizer is computable, the candidate set GenomeCoder constructs captures substantially more posterior mass per mechanistic evaluation than the model-factorized proposal and uniform sampling in V(D)J, or beam search and prior sampling in splicing (Figure 3, top right). The plotted quantity is the measured retained mass $C(S_K)$ of §3.4, whose complement is the total variation error at a fixed mechanistic model. These are matched-model search comparisons. Exact-normalizer validation is in Appendix C, the ARG audit in Appendix J.

5.3 Generative-parameter learning

Exact validation marginal likelihood selects the retained EM iterate using observations alone (§3.6): at a matched number of exact model evaluations the distance to the generating parameters falls from 0.3757 to $\mathbf{0.1911}$, against 0.3077 for the model-factorized proposal (Figure 3, bottom left). Across $1{,}000$, $5{,}000$ and $19{,}540$ training reads, parameter TV falls from 0.2334 to 0.1632 as the criterion selects more updating.

In splicing the unknown component is the program prior itself, learned from sequence with no isoform labels. It recovers $+\mathbf{0.2568}$ of the oracle ceiling on held-out chr7 against a permutation baseline of $+0.0940$, and exceeds the permutation baseline on all three chromosomes tested; a long-context DNA model (Nguyen et al., 2023) exceeds it on none of the three (Appendix K).

5.4 End-to-end inference

Across five independently regenerated $5{,}000$-read datasets, each with an independently trained recognizer, GenomeCoder and IGoR (Marcou et al., 2018) both recover complete scenarios at mean rate $\mathbf{0.1330}$ under the same strict metric and the same denominator of 400 held-out reads; the paired difference is $+\mathbf{0.0000}$ with 95% CI $[-0.0155, +0.0155]$ (Figure 3, bottom right). The benchmark poses the end-to-end regime: given reads alone, learn the recombination model and *then* recover latent scenarios on held-out reads. We reproduce the published row (0.1400 against the paper's 0.15), then run both systems under identical conditions, neither receiving the generating parameters. No margin was specified in advance, so we report an interval and no equivalence test. GenomeCoder and IGoR track the same per-split difficulty ($r = +0.51$; Appendix D).

The restricted support is as accurate as enumeration. This is an internal, same-model control and not a comparison with IGoR. Under the *same* learned GenomeCoder model, the retained support reaches the recovery enumeration reaches, agreeing on the MAP for 74.5% of reads and statistically indistinguishable on correctness (exact McNemar $p = 0.69$), while scoring $\sim$118 programs per read against $\sim$1.70 million enumerated candidates: $\mathbf{14{,}415}\times$ fewer exact evaluations. In the current prototype proposal overhead remains the dominant implementation cost, reported in full in Appendix I.

5.5 The supervised limiting case

Program-structured supervision improves multi-operation composition recovery on the structural-variation benchmark of Lin et al. (2022): under the supplied decoder and protocol, exact operation-multiset recovery rises from 24.5% to $\mathbf{61.7}\%$ against a matched-capacity flat target and operation F1 from 0.777 to $\mathbf{0.889}$ (Table 12). The benchmark is simulated, and the target program is observed during training (Appendix G).

6 Discussion

GenomeCoder uses a recognition model to concentrate exact mechanistic evaluation on a small set of executable latent programs. The same restricted posterior is used for latent inference and, when generative parameters are unknown, for parameter learning.

In V(D)J, GenomeCoder matches IGoR on the reads-only end-to-end benchmark. In splicing, it ranks second of eight methods on the realistic Sarantopoulou benchmark. Genealogy inference provides a third latent-variable setting, with a different program representation and reference method.

When part of the generative model is unknown, GenomeCoder estimates it from posterior responsibilities. In V(D)J the exact validation marginal likelihood selects among generative updates; in splicing the same mechanism recovers a sequence-conditioned prior that was withheld, exceeding the permutation baseline on every chromosome tested (Appendix K). The mechanism applies both to a fitted parameter block and to one that is absent altogether.

In V(D)J and splicing the marginal likelihood is tractable, so the posterior mass retained by the candidate set is measured exactly. A stronger q_ϕ improves the candidate set, but the mechanistic posterior, the omitted mass and the observed-data evidence that parameter learning consumes all come from the mechanistic model.

Limitations. Latent truth is simulator- or reference-based in two domains, so the evaluation is only as good as the mechanistic model, which also defines the target posterior; misspecification is outside Theorem 1's scope, and exact retained mass is shown for V(D)J and splicing. The external comparison is replicated at $5{,}000$ reads; the $1{,}000$- and $19{,}540$-read points are single draws, and the five-seed interval (± 0.022) is not an equivalence margin.

7 Related Work

Mechanistic biological inference with evaluable likelihoods. GenomeCoder targets biological inference problems with an evaluable mechanistic model and a large discrete latent space. Genealogy inference is served at scale by tsinfer and tsdate (Kelleher et al., 2019; Wohns et al., 2022), Relate (Speidel et al., 2019) and SINGER (Deng et al., 2025); we use ARGweaver (Rasmussen et al., 2014) as the small n probabilistic reference because it targets the same posterior our score defines, and we claim no production-scale reconstruction. Structural-variant discovery has a parallel line of learned callers including Cue (Popic et al., 2023), SVision (Lin et al., 2022) and Severus (Keskus et al., 2024). In transcript quantification (Li & Dewey, 2011; Patro et al., 2017), Salmon's equivalence classes collapse fragments carrying the same quantification information; GenomeCoder uses exact fragment-mask regrouping to compute retained posterior mass over set-valued programs.

Amortized inference and inference compilation. Recognition models paired with generative models (Dayan et al., 1995; Hinton et al., 1995), amortized inference (Gershman & Goodman, 2014), latent programs as explanations (Lake et al., 2015; Ellis et al., 2021) and inference compilation (Le et al., 2017) amortize posterior computation; GenomeCoder amortizes support discovery, and posterior weights are recomputed from the mechanistic model.

Exact correction and simulation-based inference. Importance-weighted and rejection-based estimators (Naderiparizi et al., 2022) correct proposal bias through proposal accounting, and likelihood-free methods (Cranmer et al., 2020; Papamakarios et al., 2019; Gloeckler et al., 2024) address unevaluable likelihoods; GenomeCoder instead exact-scores a finite mechanistic support and measures the retained posterior mass (Appendices B, L).

AI USE STATEMENT

We used generative AI tools for coding, drafting, and verification support. Research ideas, scientific framing, experimental design, and conclusions were developed by the authors. Every number in this paper is re-derived from its source artifact by a checker that reads the artifact at generation time rather than being accepted from generated text, and AI-assisted code was reviewed and tested by the authors. We take responsibility for the final content of this work.

ETHICS STATEMENT

This work uses public simulated data and public human and mouse reference and annotation resources. The immune-repertoire analysis uses a published model rather than individual-level repertoire data. The recombination-rate supervision used in the appendix derives from published crossover maps from one assay in one population, which limits the population generality of any conclusion drawn from it; the conclusion we draw from it is negative. No new human-subjects data was collected and no individual-level genotypes are released. The methods are inference tools for basic genomics, and we are not aware of a dual-use concern specific to them beyond those already present in the published tools they build on and are compared against.

REPRODUCIBILITY STATEMENT

Every numerical claim traces to a stored result file. Exact tool versions and build patches, hardware and thread counts for every timed comparison, random seeds and replicate counts for every reported aggregate, and the thresholds and tolerances fixed before the runs that tested them are given in Appendix M. Timing artifacts record the machine that produced them alongside the measurement, so a speedup cannot be separated from its hardware. Appendix M also records how three specific hazards were handled: silent sample renumbering by external tools, an under-specified published metric, and conventions that move our score.

REFERENCES

Franz Baumdicker, Gertjan Bisschop, Daniel Goldstein, Graham Gower, Aaron P Ragsdale, Georgia Tsambos, Sha Zhu, Bjarki Eldon, E Castedo Ellerman, Jared G Galloway, et al. Efficient ancestry and mutation simulation with msprime 1.0. *Genetics*, 220(3):iyab229, 2022.

Dmitriy A Bolotin, Stanislav Poslavsky, Igor Mitrophanov, Mikhail Shugay, Ilgar Z Mamedov, Ekaterina V Putintseva, and Dmitriy M Chudakov. Mixcr: software for comprehensive adaptive immunity profiling. *Nature methods*, 12(5):380, 2015.

Nicolas L Bray, Harold Pimentel, Páll Melsted, and Lior Pachter. Near-optimal probabilistic rna-seq quantification. *Nature biotechnology*, 34(5):525–527, 2016.

Kyle Cranmer, Johann Brehmer, and Gilles Louppe. The frontier of simulation-based inference. *Proceedings of the National Academy of Sciences*, 117(48):30055–30062, 2020.

Peter Dayan, Geoffrey E Hinton, Radford M Neal, and Richard S Zemel. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995.

Yun Deng, Rasmus Nielsen, and Yun S Song. Robust and accurate bayesian inference of genome-wide genealogies for hundreds of genomes. *Nature genetics*, 57(9):2124–2135, 2025.

Alexander Dobin, Carrie A Davis, Felix Schlesinger, Jorg Drenkow, Chris Zaleski, Sonali Jha, Philippe Batut, Mark Chaisson, and Thomas R Gingeras. Star: ultrafast universal rna-seq aligner. *Bioinformatics*, 29(1):15–21, 2013.

Kevin Ellis, Catherine Wong, Maxwell Nye, Mathias Sablé-Meyer, Lucas Morales, Luke Hewitt, Luc Cary, Armando Solar-Lezama, and Joshua B Tenenbaum. Dreamcoder: Bootstrapping inductive program synthesis with wake-sleep library learning. In *Proceedings of the 42nd acm sigplan international conference on programming language design and implementation*, pp. 835–850, 2021.

Adam C English, Vipin K Menon, Richard A Gibbs, Ginger A Metcalf, and Fritz J Sedlazeck. Truvari: refined structural variant comparison preserves allelic diversity. *Genome Biology*, 23(1): 271, 2022.

Joseph Felsenstein. Evolutionary trees from dna sequences: a maximum likelihood approach. *Journal of molecular evolution*, 17(6):368–376, 1981.

Lex Flagel, Yaniv Brandvain, and Daniel R Schrider. The unreasonable effectiveness of convolutional neural networks in population genetic inference. *Molecular biology and evolution*, 36(2): 220–238, 2019.

Samuel Gershman and Noah Goodman. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

Manuel Gloeckler, Michael Deistler, Christian Weilbach, Frank Wood, and Jakob H Macke. All-in-one simulation-based inference. *arXiv preprint arXiv:2404.09636*, 2024.

Geoffrey E Hinton, Peter Dayan, Brendan J Frey, and Radford M Neal. The" wake-sleep" algorithm for unsupervised neural networks. *Science*, 268(5214):1158–1161, 1995.

Kishore Jaganathan, Sofia Kyriazopoulou Panagiotopoulou, Jeremy F McRae, Siavash Fazel Darbandi, David Knowles, Yang I Li, Jack A Kosmicki, Juan Arbelaez, Wenwu Cui, Grace B Schwartz, et al. Predicting splicing from primary sequence with deep learning. *Cell*, 176(3): 535–548, 2019.

Jerome Kelleher, Yan Wong, Anthony W Wohns, Chaimaa Fadil, Patrick K Albers, and Gil McVean. Inferring whole-genome histories in large population datasets. *Nature genetics*, 51(9):1330–1338, 2019.

Ayse Keskus, Asher Bryant, Tanveer Ahmad, Byunggil Yoo, Sergey Aganezov, Anton Goretsky, Ataberk Donmez, Lisa A Lansdon, Isabel Rodriguez, Jimin Park, et al. Severus: accurate detection and characterization of somatic structural variation in tumor genomes using long reads. *MedRxiv*, 2024.

Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum. Human-level concept learning through probabilistic program induction. *Science*, 350(6266):1332–1338, 2015.

Tuan Anh Le, Atilim Gunes Baydin, and Frank Wood. Inference compilation and universal probabilistic programming. In *Artificial Intelligence and Statistics*, pp. 1338–1348. PMLR, 2017.

Bo Li and Colin N Dewey. Rsem: accurate transcript quantification from rna-seq data with or without a reference genome. *BMC bioinformatics*, 12(1):323, 2011.

Jiadong Lin, Songbo Wang, Peter A Audano, Deyu Meng, Jacob I Flores, Walter Kosters, Xiaofei Yang, Peng Jia, Tobias Marschall, Christine R Beck, et al. Svision: a deep learning approach to resolve complex structural variants. *Nature methods*, 19(10):1230–1233, 2022.

Quentin Marcou, Thierry Mora, and Aleksandra M Walczak. High-throughput immune repertoire analysis with igor. *Nature communications*, 9(1):561, 2018.

Paul Marjoram and Jeff D Wall. Fast" coalescent" simulation. *BMC genetics*, 7(1):16, 2006.

Gilean AT McVean and Niall J Cardin. Approximating the coalescent with recombination. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 360(1459):1387, 2005.

Saeid Naderiparizi, Adam Scibior, Andreas Munk, Mehrdad Ghadiri, Atilim Gunes Baydin, Bradley J Gram-Hansen, Christian A Schroeder De Witt, Robert Zinkov, Philip Torr, Tom Rainforth, et al. Amortized rejection sampling in universal probabilistic programming. In *International Conference on Artificial Intelligence and Statistics*, pp. 8392–8412. PMLR, 2022.

Eric Nguyen, Michael Poli, Marjan Faizi, Armin Thomas, Michael Wornow, Callum Birch-Sykes, Stefano Massaroli, Aman Patel, Clayton Rabideau, Yoshua Bengio, et al. Hyenadna: Long-range genomic sequence modeling at single nucleotide resolution. *Advances in neural information processing systems*, 36:43177–43201, 2023.

George Papamakarios, David Sterratt, and Iain Murray. Sequential neural likelihood: Fast likelihood-free inference with autoregressive flows. In *The 22nd international conference on artificial intelligence and statistics*, pp. 837–848. PMLR, 2019.

Rob Patro, Geet Duggal, Michael I Love, Rafael A Irizarry, and Carl Kingsford. Salmon provides fast and bias-aware quantification of transcript expression. *Nature methods*, 14(4):417–419, 2017.

Victoria Popic, Chris Rohlicek, Fabio Cunial, Iman Hajirasouliha, Dmitry Meleshko, Kiran Garimella, and Anant Maheshwari. Cue: a deep-learning framework for structural variant discovery and genotyping. *Nature methods*, 20(4):559–568, 2023.

Matthew D Rasmussen, Melissa J Hubisz, Ilan Gronau, and Adam Siepel. Genome-wide inference of ancestral recombination graphs. *PLoS genetics*, 10(5):e1004342, 2014.

David F Robinson and Leslie R Foulds. Comparison of phylogenetic trees. *Mathematical biosciences*, 53(1-2):131–147, 1981.

Théophile Sanchez, Jean Cury, Guillaume Charpiat, and Flora Jay. Deep learning for population size history inference: design, comparison and combination with approximate bayesian computation. *Molecular Ecology Resources*, 21(8):2645–2660, 2021.

Dimitra Sarantopoulou, Thomas G Brooks, Soumyashant Nayak, Antonijo Mrčela, Nicholas F Lahens, and Gregory R Grant. Comparative evaluation of full-length isoform quantification from rna-seq. *BMC bioinformatics*, 22(1):266, 2021.

Leo Speidel, Marie Forest, Sinan Shi, and Simon R Myers. A method for genome-wide genealogy estimation for thousands of samples. *Nature genetics*, 51(9):1321–1329, 2019.

Songbo Wang, Jiadong Lin, Peng Jia, Tun Xu, Xiujuan Li, Yuezhuangnan Liu, Dan Xu, Stephen J Bush, Deyu Meng, and Kai Ye. De novo and somatic structural variant discovery with svision-pro. *Nature biotechnology*, 43(2):181–185, 2025.

Zhanpeng Wang, Jiaping Wang, Michael Kourakos, Nhung Hoang, Hyong Hark Lee, Iain Mathieson, and Sara Mathieson. Automatic inference of demographic parameters using generative adversarial networks. *Molecular ecology resources*, 21(8):2689–2705, 2021.

Anthony Wilder Wohns, Yan Wong, Ben Jeffery, Ali Akbari, Swapan Mallick, Ron Pinhasi, Nick Patterson, David Reich, Jerome Kelleher, and Gil McVean. A unified genealogy of modern and ancient genomes. *Science*, 375(6583):eabi8264, 2022.

A POSTERIOR-FREE SUPERVISION: STATEMENT AND PROOF

The main text states this result in prose (§3.5); here is the formal version, followed by its derivation.

Theorem 1 (Posterior-free supervision is distillation in expectation)**.** *Let $x \sim p_X$, $P \sim p_\psi(P \mid x)$ and $R \sim p_\theta(R \mid P, x)$, so (x, P, R) is drawn from the training joint of equation 1. Then*

$$\mathbb{E}_{x,P,R}\big[-\log q_\phi(P \mid x, R)\big] \;=\; \mathbb{E}_{x,R}\Big[\mathrm{KL}\big(p_{\theta,\psi}(P \mid x, R) \,\big\|\, q_\phi(P \mid x, R)\big)\Big] \;+\; C_{\theta,\psi}, \tag{7}$$

where $C_{\theta,\psi}$ does not depend on ϕ. The population optimum of single generating-program supervision is therefore the exact posterior $p_{\theta,\psi}(P \mid x, R)$.

Proof. Let $x \sim p_X$, $P \sim p_\psi(P \mid x)$ and $R \sim p_\theta(R \mid P, x)$, so (x, P, R) is a draw from the training joint $p_{\theta,\psi}(x, P, R) = p_X(x)\, p_\psi(P \mid x)\, p_\theta(R \mid P, x)$. Write $p_{\theta,\psi}(P \mid x, R)$ for the exact posterior it induces. Then

$$\mathbb{E}_{x,P,R}\big[-\log q_\phi(P \mid x, R)\big] = -\int p_X(x) \sum_P \int p_\psi(P \mid x)\, p_\theta(R \mid P, x)\, \log q_\phi(P \mid x, R)\, dR\, dx \tag{8}$$

$$= -\int p_X(x) \int p_{\theta,\psi}(R \mid x) \sum_P p_{\theta,\psi}(P \mid x, R)\, \log q_\phi(P \mid x, R)\, dR\, dx, \tag{9}$$

using $p_\psi(P \mid x)\, p_\theta(R \mid P, x) = p_{\theta,\psi}(R \mid x)\, p_{\theta,\psi}(P \mid x, R)$. The inner sum is a cross-entropy between $p_{\theta,\psi}(\cdot \mid x, R)$ and $q_\phi(\cdot \mid x, R)$, so adding and subtracting the entropy of the exact posterior gives

$$\mathbb{E}_{x,P,R}\big[-\log q_\phi(P \mid x, R)\big] = \mathbb{E}_{x,R}\Big[\mathrm{KL}\big(p_{\theta,\psi}(P \mid x, R) \,\big\|\, q_\phi(P \mid x, R)\big)\Big] + \underbrace{\mathbb{E}_{x,R}\big[\mathbb{H}[p_{\theta,\psi}(P \mid x, R)]\big]}_{= C_{\theta,\psi}}, \tag{10}$$

and $C_{\theta,\psi}$ depends only on (θ, ψ) and p_X, not on ϕ. Minimizing the left-hand side is therefore equivalent to minimizing the expected KL to the exact posterior, whose minimizer over a sufficiently expressive family is $q_\phi = p_{\theta,\psi}(P \mid x, R)$. ■

Scope. Three conditions bound what this licenses. *(i)* The identity is under the **training joint**: the same (θ, ψ) generated the data being trained on. *(ii)* It is a **population** statement; at finite sample size the single draw is an unbiased but higher-variance estimate of the per-observation cross-entropy, which is why the empirical gap closes with data rather than vanishing at every size. *(iii)* It says **nothing about simulator shift**: if test data comes from $(\theta', \psi') \neq (\theta, \psi)$, the theorem does not apply, and the misspecification sweeps reported in this paper are empirical robustness evidence rather than corollaries.

(iv) It is a fixed-(θ, ψ) statement, so it does not license the approximate-EM results. During EM the generative parameters move between iterations while the recognizer stays fixed, so the joint that the recognizer was trained under is *not* the joint it is applied to at iteration $t > 0$. A parameter-conditioned recognizer would need training coverage over the whole region of parameter space visited at inference time for the theorem to transfer, and we do not establish that such coverage holds. The approximate-EM results in this paper are therefore **empirical evidence that the recognizer degrades gracefully as θ moves**, not a consequence of Theorem 1.

Two remarks matter for reading the experiments. The result is about the *population* objective: a single generating program is an unbiased but high-variance estimate of the per-observation cross-entropy, so finite-sample behavior depends on how much data averages that variance away, which is what §5.1 measures, and why the gap closes between $4{,}000$ and $16{,}000$ training reads. And replacing the single draw with M posterior draws reduces that variance without changing the optimum, which is the interpolation we measure: M posterior draws reach the exact teacher at $M{=}64$, and distillation is the $M \to \infty$ limit.

B THE SUPPORT-CONDITIONED ESTIMATOR

GenomeCoder draws proposal samples and then discards their frequencies. This appendix derives the resulting estimator and places it beside the p/q importance-weighted estimator, which uses those

Table 2: Complex structural variants, held-out chromosomes, published protocol. The flat target shares our trunk, evidence, chromosomes, optimizer and parameter count (106,052 vs 105,987); only what it predicts differs. The highlighted columns are the endpoints this experiment is about: operation F1 and *multiset* (every operation type at the right multiplicity), both on matched calls. Appendix G gives all caller rows and the coordinate-aware evaluation.

method	detection F1	operation F1	exact multiset	breakpoint ≤100 bp
best single published caller	0.044	*n/a*	*n/a*	*n/a*
engineered four-caller union	0.280	0.446	0.000	0.911
matched-capacity flat target	0.546	0.777	0.245	0.950
GenomeCoder (program target)	0.467	**0.889**	**0.617**	0.927

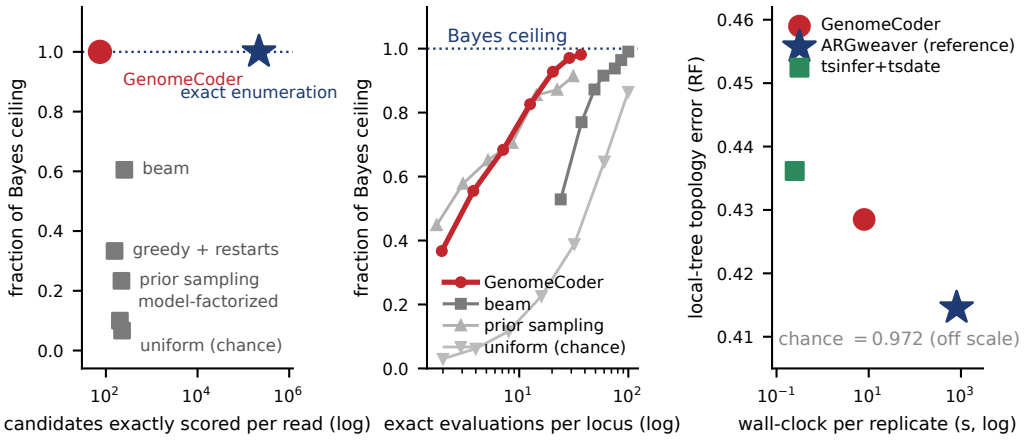


Figure 4: Latent-program inference across three domains. Each domain is scored against its own ceiling or reference in its own native metric; the panels are *not* commensurate and deliberately do not share an axis. *(Left)* V(D)J: GenomeCoder attains the Bayes ceiling (the exact posterior's own accuracy, 0.250, not perfection) while exactly scoring 2951× fewer candidates than enumeration; the best non-neural search reaches 0.605 of it. *(Middle)* splicing: the full accuracy–budget frontier, 0.981 of ceiling at 37 exact evaluations where beam reaches 0.770. *(Right)* ARG: a published *reference*, not a ceiling. ARGweaver is the more accurate method by +0.014 RF; GenomeCoder reaches that neighborhood at 101× less wall-clock in the n=12 regime evaluated here, and both are reported together.

frequencies. Both see the same draws,

$$P^{(1)}, \ldots, P^{(K)} \sim q_\phi(P \mid x, R), \qquad S_K = \{P^{(1)}, \ldots, P^{(K)}\}_{\text{unique}}. \tag{11}$$

The importance-weighted estimator. Self-normalized importance sampling forms weights and an estimate

$$w_i = \frac{p_{\theta,\psi}(P^{(i)}, R \mid x)}{q_\phi(P^{(i)} \mid x, R)}, \qquad \widehat{\mathbb{E}}_{\text{IS}}[T] = \frac{\sum_i w_i\, T(P^{(i)})}{\sum_i w_i}, \tag{12}$$

consistent for full-posterior expectations as $K \to \infty$. Its accuracy is governed by the variance of w, so it depends on q_ϕ at every drawn program, and a program drawn many times contributes many times.

The support-conditioned estimator. GenomeCoder instead forms

$$\tilde{p}(P \mid x, R, S_K) = \frac{p_{\theta,\psi}(P, R \mid x)}{\sum_{P' \in S_K} p_{\theta,\psi}(P', R \mid x)}, \qquad P \in S_K, \tag{13}$$

the scientific posterior conditioned on the realized finite support. Three consequences follow. *Proposal density does not enter*: equation 13 is a function of S_K and the mechanistic model alone, so a miscalibrated q_ϕ changes the support and leaves the weights on it exact, which is what licenses reporting these numbers as scientific posteriors. *Duplicates carry no information*: redrawing a discovered program adds no element to S_K, so deduplication makes exact mechanistic work proportional to the number of unique hypotheses, which is the quantity that costs anything. *The error has a name*: by equation 3 the total variation distance from $\tilde{p}$ to the full posterior is exactly $1 - C(S_K)$,

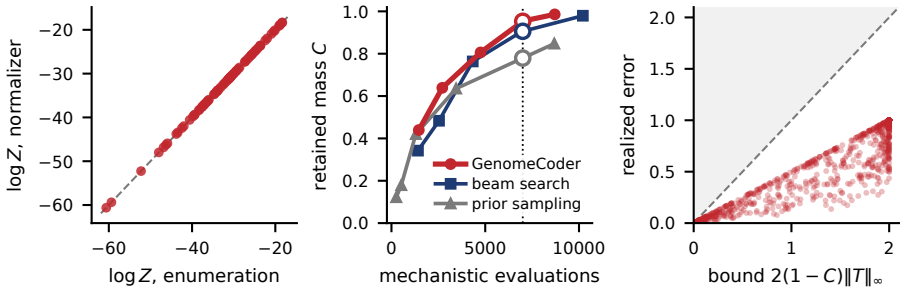


Figure 5: **Exact posterior-mass accounting.** *(Left)* V(D)J: the exact normalizer against full enumeration. *(Middle)* Splicing: retained mass against exact evaluations for three proposals, each interpolated on its own curve; the dotted line is the shared budget. *(Right)* The certificate of equation 6 on transcript inclusion.

and equation 6 converts that into a bound on any bounded posterior expectation, measured rather than assumed wherever Z is computable.

So equation 12 targets full-posterior expectations, at the cost of depending on q_ϕ at every drawn program, while equation 13 targets the posterior restricted to what search found and reports how much posterior that support holds. Each is better at its own target, and the measurement below gives the sizes.

C EXACT COVERAGE WITHOUT ENUMERATION

Table 3: V(D)J sufficient-statistic error against retained mass, 3,375 cells (75 reads × five budgets × nine statistics). Length families are normalized by $\|T\|_\infty$, so every row shares the $2(1 - C)$ bound. The support is the recognizer's own.

coverage bin	mean C	gene choice	deletion len.	insertion len.	bound $2(1 - C)$
$C < 0.75$	0.4523	0.04792	0.02943	0.02028	1.0953
$0.75 \le C < 0.90$	0.8260	0.01217	0.01010	0.00846	0.3479
$0.90 \le C < 0.95$	0.9291	0.00460	0.00421	0.00385	0.1418
$0.95 \le C < 0.99$	0.9722	0.00201	0.00180	0.00164	0.0555
$C \ge 0.99$	0.9935	0.00043	0.00045	0.00050	0.0130

Table 4: V(D)J system identification across training scale. The criterion uses observations alone. The 1,000- and 19,540-read rows are single draws; the 5,000-read row is the mean over five independent draws, where the selected iterate ranges from 0 to 2.

training reads	parameter TV to truth ↓	updates selected
1,000	0.2334	0.00
5,000	0.1824	0.53
19,540	**0.1632**	1.67

The oracle ceiling on any budget-K support. Two quantities are worth separating. The K=256 *highest-posterior* programs carry 0.907 of the posterior mass on average and 0.485 at worst, which is the most any support of that size could reach and is a property of the posterior rather than of any search method. GenomeCoder's own proposal retains 0.840 at K=4096 on the held-out evaluation. The main text compares against realizable search methods, since the oracle top-K set requires the exact posterior that inference is trying to avoid computing.

What is reused. The table Q below depends on the model, not the read, so it is built once per parameter setting and amortized across the corpus; the per-span completion vectors are memoized on (span, V end, J start, length) and shared both across anchor prefixes within a read and across reads.

Setup. For a retained support S_K the coverage is $C = Z_{S_K}/Z$ with $Z_{S_K} = \sum_{P \in S_K} w(P)$ and $Z = \sum_P w(P)$ over the whole legal support, where $w(P) = p_\theta(P)\, p_\theta(R \mid P)$. Z_{S_K} is free (those programs are exact-scored anyway) so the difficulty is entirely Z.

Two structural facts. On the error-free support the likelihood is $(1-e)^{L-|i_{vd}|-|i_{dj}|}$, depending on the program only through total insertion length; and the prior factorizes over the recombination network. Each program is therefore an anchor prefix $a = (v, \delta_V, j, \delta_J)$ plus a D-side completion $(d, d_5, d_3, \mathrm{pos})$, and given the prefix and the placement the insertion strings are *determined* by the read.

The inversion. Enumeration scans, for every prefix, all trimmings of D and all placements of the resulting core in the span $s = y[v_{\mathrm{end}}{:}j_{\mathrm{start}}]$. But at position pos and core length c there is exactly one string the core can be, $s[\mathrm{pos}{:}\mathrm{pos}{+}c]$. Precomputing, once per model and for each D gene,

$$Q_d[u] \;=\; \sum_{(d_5,d_3)\colon \mathrm{core}=u} p(d_5 \mid d)\, p(d_3 \mid d, d_5) \tag{14}$$

turns the scan into a lookup, and the sum over placements and insertion lengths closes as

$$E_d \;=\; \sum_{\mathrm{pos},\,c} Q_d\big[s[\mathrm{pos}{:}\mathrm{pos}{+}c]\big]\; p_{vd}(n_1) P(i_{vd})\; p_{dj}(n_2) P(i_{dj})\,(1-e)^{L-n_1-n_2}, \tag{15}$$

with $n_1 = \mathrm{pos}$ and $n_2 = |s| - \mathrm{pos} - c$, memoized on the span. Then $Z = \sum_a e^{\ell_{vj}(a)} \sum_d p(d \mid v, j)\, E_d$. This is the sum enumeration computes, reorganized from $O(\#\mathrm{trimmings} \times |s|)$ string searches into $O(|s|^2)$ lookups per span; nothing is relaxed.

Validation. On 75 held-out reads whose posteriors are enumerable, the normalizer reproduces the enumerated Z to a worst relative error of 4.07×10^{-12}, with total time 36.8 s against 5,293 s. The per-read speedup ranges from $5\times$ to $1{,}029\times$ (median $52\times$); a single headline factor would misrepresent that spread. Per read the enumerator visits a median of 124,526 programs where the normalizer visits 77 anchor prefixes.

Table 5: Exact coverage without enumeration, on 75 held-out reads whose posteriors are small enough that full enumeration is affordable and can serve as ground truth. The normalizer reproduces the enumerated value to floating-point accuracy while visiting a median of 77 anchor prefixes in place of 124,526 programs.

	worst rel. error	total time	programs / prefixes visited	speedup
full enumeration	*n/a* (reference)	5,293 s	124,526 (median)	*n/a*
exact normalizer	$\mathbf{4.07 \times 10^{-12}}$	**36.8** s	**77** (median)	$\mathbf{144}\times$

Worst-case bound tightness. The bound of equation 6 is a worst case: it is attained only when the omitted mass carries statistics at the opposite extreme of T. Over the 3,375 V(D)J sufficient-statistic cells the median realized/bound ratio is 0.0156 and the maximum is 0.3454, so the bound holds everywhere while remaining conservative. Retained mass is therefore best read as a direct measurement of posterior completeness, with the bound as the formal guarantee that accompanies it.

Exact coverage by budget, for an oracle-ordered support. Averaged over the evaluated reads, coverage reaches 0.629 at $K{=}16$, 0.806 at $K{=}64$, 0.907 at $K{=}256$ and 0.965 at $K{=}1024$; the per-budget minima are 0.085, 0.239, 0.485 and 0.736. These supports are the K highest-posterior programs, taken in that order because the purpose here is to validate the bound across a controlled range of C. They therefore describe the best coverage a budget-K support could attain, not the coverage the recognizer attains; the recognizer's own realized coverage, and the sufficient-statistic error that follows from it, are in §D. The minimum is the operationally relevant column, since the bound of §3.3 applies per read.

The enumeration cap. The comparison arm in §5.4 enumerates to a 4,000,000-program cap. In a matched sample of 115 reads the cap binds on 2, where the enumerated posterior captures 0.283 and 0.981 of the true mass. The cap-binding rate under a learned model was not separately instrumented for the reported fits, so the arm is named by its protocol (enumeration to a 4M-program cap) rather than by an assumption about completeness.

Scope. The construction uses V(D)J structure: a likelihood depending on the program only through insertion length, and a D segment entering as a contiguous substring. The construction therefore applies to V(D)J and to splicing, which are the domains where the paper reports exactly computable coverage.

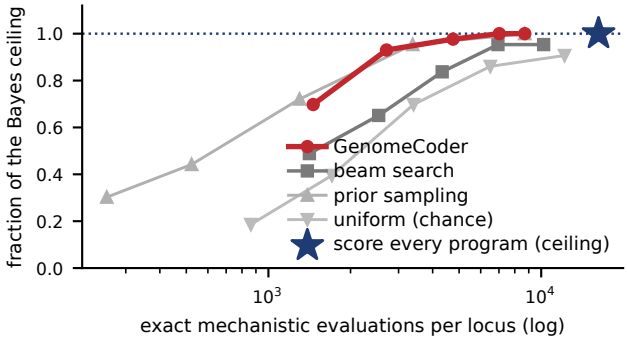


Figure 6: Splicing under harder latent geometry: skewed (Dirichlet) abundances, up to five expressed isoforms, 3′-biased fragment sampling against an unchanged uniform-position scorer, and 3,354–21,050 programs per locus against 40–600 in the fixed-prior benchmark. Exact enumeration is retained, so the Bayes ceiling remains computable; it falls to 0.3258 because the read model is deliberately mismatched, which is why the axis is a fraction of that ceiling and not absolute accuracy. Each method is swept across its own budget knob and charged for the weighted programs it actually scores. 132 cells per configuration (22 loci × 6 seeds).

D READS-ONLY V(D)J: PER-SEED DETAIL

Smaller and larger training sets. At 1,000 reads GenomeCoder is ahead on the evaluated draw by $+\mathbf{0.0467}$, each recognizer instance individually significant (exact McNemar $p = 0.0002$, 0.0081, 0.0009), and at 19,540 reads it reaches $\mathbf{0.1350}$. Both are single draws, and the matched external comparison is at 5,000.

Parameter and program recovery dissociate. Our fitted model is further from the generating parameters than IGoR's (KL 11.77 vs 4.67) while supporting equivalent latent recovery: parameter fidelity is an intermediate quantity on the way to the endpoint.

Five independent data seeds at 5,000 training reads, with three recognizer instances each, so fifteen end-to-end GenomeCoder fits and five comparator fits. Each data seed regenerates the training repertoire, the held-out repertoire, the IGoR fit and the GenomeCoder fits; within a seed all arms are graded on the identical 400 held-out reads, an identity the harness asserts before a seed is accepted.

Table 6: Reads-only V(D)J benchmark under matched conditions at 5,000 training reads, over five independent data seeds with three recognizer instances each. Each seed regenerates the training and held-out repertoires and refits both systems from a generic initialization; within a seed both are graded on the identical 400 held-out reads, and a declined call counts as incorrect. The paired difference is taken per data seed, which is the level at which the fits are independent. The published protocol reproduces at 0.1400 but learns and evaluates on the same 20,000 reads, so it is quoted in the text rather than tabulated beside matched rows. Each data split is independently regenerated. The same three independently trained recognizer instances are evaluated across the five splits, so data-split variation and recognizer-instance variation are reported separately.

method	exact scenario	sd	paired Δ	95% CI	programs/read
IGoR, matched conditions	0.1330	0.0201	*n/a*	*n/a*	*n/a*
GenomeCoder	**0.1323**	0.0138	−0.0007	[−0.0226, +0.0213]	**117.6**

The data seed is the unit of analysis, because a seed fixes both the training and the evaluation reads; the 95% interval on the mean paired difference is $[-0.0226, +0.0213]$ on t_4. A pooled read-level test is not reported as primary, since pooling would treat 2,000 reads from five different fits as exchangeable.

A variance decomposition gives a recognizer-instance sd of 0.0126 and a data-seed sd of 0.0117; the two are comparable, so replication over recognizer instances alone would capture about half of the relevant variability.

This table shares recognizer instances across data seeds, so the decomposition above separates the two sources within that design. Recognizer-training variability is measured directly by the independent fits reported below.

Table 7: Per-data-seed reads-only V(D)J results at 5,000 training reads. Each seed redraws the training and the held-out reads; within a seed both methods are graded on the identical 400 held-out reads and a declined call counts as incorrect. GenomeCoder is the mean over three independently trained recognizer instances, whose spread is the last column.

data seed	GenomeCoder	IGoR	Δ	recognizer sd
0	0.1150	0.1225	−0.0075	0.0132
1	0.1342	0.1150	+0.0192	0.0072
2	0.1242	0.1450	−0.0208	0.0076
3	0.1367	0.1200	+0.0167	0.0213
4	0.1517	0.1625	−0.0108	0.0076
mean	0.1323	0.1330	−0.0007	(
sd	0.0138	0.0201	0.0177	*n/a*

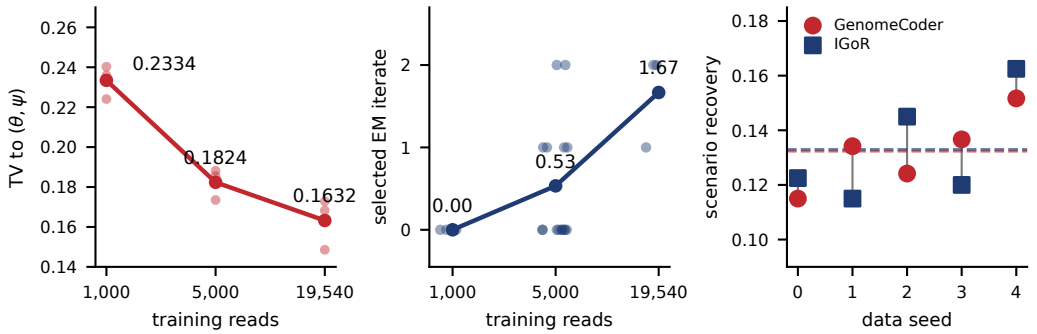


Figure 7: **V(D)J generative learning and the reads-only benchmark.** *(Left)* and *(Middle)* show the three evaluated training scales. The 1,000- and 19,540-read endpoints are single draws, while the 5,000-read column contains five independent data draws: there, faint points are the individual draws and the label gives their mean. *(Left)* Learned-model total variation to the generating parameters by scale. *(Middle)* The EM iterate selected by exact validation marginal likelihood. *(Right)* Those same five matched 5,000-read splits, GenomeCoder against IGoR per split. The compute comparison rests on a different seed set and is reported in Figure 9.

Independent recognizers per split. The row above evaluates the same three recognizer instances across all five data splits, so recognizer-training variance is shared rather than replicated. To close that, one recognizer was trained independently for each split, with the architecture, block, mode, corpus and step count identical to the shared instances and only the training seed differing, and the frozen recipe was then run per split on its own recognizer.

Table 8: Five genuinely independent end-to-end fits: one freshly trained recognizer per data split, against the shared-recognizer row and IGoR. Each split regenerates the training and held-out repertoires; IGoR's per-split values are the same ones the shared row is paired against. The last column is the EM iterate the observation-only criterion selected.

split	fresh recognizer	shared (published)	IGoR	Δ fresh	iterate
0	0.1100	0.1150	0.1225	−0.0125	0
1	0.1275	0.1342	0.1150	+0.0125	0
2	0.1325	0.1242	0.1450	−0.0125	0
3	0.1450	0.1367	0.1200	+0.0250	0
4	0.1500	0.1517	0.1625	−0.0125	2
mean	**0.1330**	0.1323	0.1330	**+0.0000**	

Replication at the recognizer does not move the result. The five independent fits mean 0.1330 ± 0.0158, against 0.1323 ± 0.0138 for the shared-recognizer row and 0.1330 ± 0.0201 for IGoR, and the paired difference to IGoR is $+0.0000$ with 95% CI $[-0.0155, +0.0155]$, against -0.0007 $[-0.0162, +0.0148]$ before. Spread across splits rises by $1.14\times$, which is the expected consequence of each split now resting on one recognizer where the published row averaged three, not a new source of instability. The selection rule is also exercised away from its boundary here: it keeps iterates 0, 2 across the five splits, so the observation-only criterion is seen choosing an interior iterate and parity still holds.

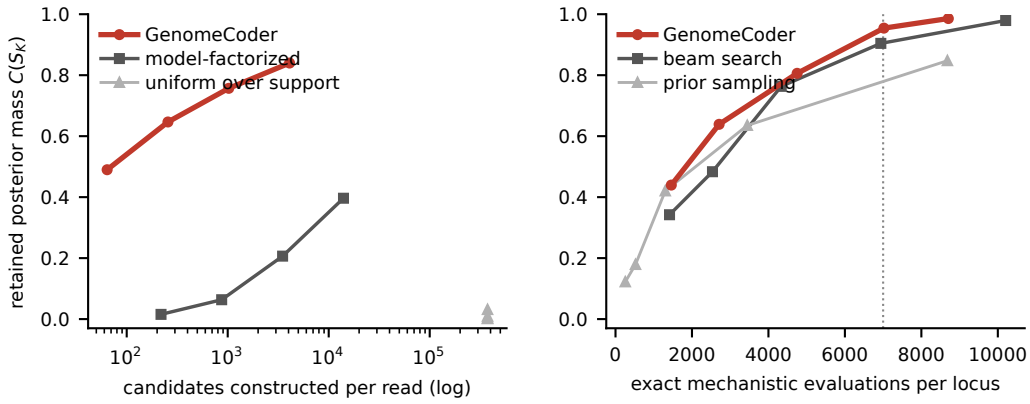


Figure 8: **Posterior mass recovered per unit of mechanistic work**, in the two domains where retained mass is exactly measurable. *(Left)* V(D)J, at $K{=}4096$: $\mathbf{0.840}$ against 0.396 for the model-factorized proposal. *(Right)* Splicing, at the shared budget of $7{,}000$ evaluations marked by the dotted line: $\mathbf{0.953}$ against 0.906 for beam search and 0.779 for prior sampling, each arm interpolated on its own curve. Mass is exact, by the V(D)J normalizer and by fragment-mask summation.

Table 9: What each domain tests. Structural variation is the supervised limiting case, in which the target program is observed during training.

domain	program P	evidence R	learned (θ, ψ)?	posterior reference	exact C?	primary question
V(D)J	recombination scenario	read	yes	exact posterior / IGoR	**yes**	inference + identification
splicing	isoform set	fragments	hidden-prior study	exact posterior	**yes**	completeness + prior learning
ARG	genealogy threading	genotypes	fixed	reference sampler	bounded	latent inference
SV	edit program	long reads	supervised	*n/a*	*n/a*	supervised limiting case

Posterior-statistic fidelity. Estimation consumes the M-step sufficient statistics, so we compare each family under the exact posterior and under the restricted posterior on the recognizer's own support (Table 3, Appendix C).

Across all $\mathbf{3{,}375}$ V(D)J statistic cells, realized sufficient-statistic error **falls by roughly two orders of magnitude as retained mass approaches one**, with **no bound violations**, and on transcript inclusion across $\mathbf{2{,}112}$ splicing cases the realized error stays within the mass-derived bound as well (Figure 5C).

E PROGRAM LANGUAGES

Structural variation. A program is a +-joined list of typed primitive edits at a locus, each carrying an interval:

$$P \;=\; \texttt{DEL}(s_1, e_1) + \texttt{DUP}(s_2, e_2) + \texttt{INV}(s_2, e_2).$$

Executing P applies the edits to the reference interval. The example is a real held-out truth program (a deletion beside an inverted duplication) and the assembler decodes a predicted operation set into exactly this form.

V(D)J recombination. A program names eight decided fields,

$$P \;=\; \big(\texttt{v},\ \texttt{delV},\ \texttt{j},\ \texttt{delJ},\ \texttt{d},\ \texttt{delD5},\ \texttt{delD3},\ \texttt{pos}\big),$$

for example $(\texttt{V2},\ 2,\ \texttt{J3},\ 3,\ \texttt{D1},\ 1,\ 2,\ 2)$. Executing P trims each chosen segment by its deletion counts and concatenates them with the inserted bases, yielding the receptor sequence. Vocabulary sizes are in Appendix H.

Genealogies. A program is a threading path along the sequence,

$$P \;=\; (s_1, \ldots, s_L), \qquad s_i = (\text{edge}_i,\ \text{time}_i),$$

for example `THREAD sample 4`: $[0, b)$ on `edge(3)` at t_1; $[b, L)$ on `edge(anc(1,2))` at t_2. Executing P attaches the new sample to the backbone genealogy at the named edges and times, producing the local trees on each side of breakpoint b.

F COMPETENCE AND PROTOCOL REPRODUCTION

The single place to check whether the domain systems were implemented correctly.

Structural variation. Published complex-variant protocol (Lin et al., 2022) scored with `truvari bench` (English et al., 2022); all comparator callers executed by us on the same alignments, including the modern neural comparator SVision-pro. All caller rows and the coordinate-aware ladder are in Appendix G.

V(D)J. Our exact enumeration and EM reproduce IGoR's own fitted model to a median per-field KL of 0.0084, with 9 of 11 event families below 0.02 and the D $3'$ deletion weakest at 0.163. Both sides start from MD5-identical model files. The comparison instrument is itself controlled: it returns $< 10^{-30}$ for two identical models and 0.204 for a deliberately perturbed one, so agreement is distinguishable from indifference.

Genealogies. The restricted computation matches explicit enumeration of all $S^L = 256$ threading paths (S=4, L=4, three seeds): the total log-likelihood agrees to 10 decimal places, the per-position posterior marginals to maximum total variation 7.5×10^{-15}, and the recovered path is identical to brute force. Reductions hold exactly: $\rho \to 0$ recovers the single-tree marginal and a constant rate map is bit-identical to the homogeneous scalar rate. This test caught two real errors in our transition structure that inspection had missed (Appendix J). Permuting genotype columns by a random permutation of leaf indices collapses accuracy onto the chance line, confirming the inference reads the genotypes and nothing else.

G STRUCTURAL VARIATION

Data. The complex-variant truth set is the *simulated* complex-SV collection distributed with the SVision work (Lin et al., 2022), on the hs37d5/GRCh37 reference, with long-read alignments mirrored locally. It is simulated, not a real-sample benchmark, and the published release provides no persistent identifier for the simulated set itself; both facts are limitations of the benchmark rather than choices of ours, and neither is visible from the headline number.

Model and decoder. The predictor is a supervised per-bin segmentation model, not a proposal: there is no posterior in this domain. Read evidence is binned at 100 bp into seven channels (depth-normalized coverage, soft-clip starts, soft-clip ends, CIGAR deletion mass, CIGAR insertion mass, supplementary-alignment density, and SA-tag orientation-flip density). A dilated 1-D CNN (six blocks of `Conv1d(channels→64, kernel 5)` at dilations $1, 2, 4, 8, 16, 32$, each with batch-norm and GELU, then a `Conv1d(64, 3, kernel 1)` head) emits per-bin presence logits for {DEL, DUP, INV} over 200-bin windows. Training: AdamW, learning rate 2×10^{-3}, weight decay 10^{-4}, 25 epochs, windows sampled around truth events and from background. Contiguous positive runs are converted to typed intervals and handed to a **deterministic assembler** that merges overlapping primitives at a locus into a +-joined composite call.

The assembler is frozen and *shared with the four-caller union baseline*, which is what makes the flat-decoding control meaningful: both arms are decoded by the same code, so the difference between program decoding and flat decoding is attributable to the representation rather than the decoder.

Protocol. We score with the published complex-variant protocol, `truvari bench --typeignore -r 1000 -p 0` (English et al., 2022), then take containment of the truth's component-type set on true positives. Trained on chr1–3, evaluated on held-out chr4–5 (350 truth complex variants). Five published callers (cuteSV, SVIM, Sniffles, pbsv, SVision (Lin et al., 2022)) were executed by us on the same alignments under the same protocol, together with an engineered non-neural baseline: the union of four callers decoded by the *same* frozen assembler.

Results and controls. Table 10 reports detection F1, recall and operation-set accuracy on the held-out chromosomes, against the four-caller union and two controls.

The flat-decoding row is the control that isolates the representation: identical predictions, different decoder. Detection is unchanged to three decimals while operation-set accuracy collapses, so the program representation buys internal resolution and nothing else. A verbosity control rules out the alternative explanation for the 0.995: our composite calls carry 2.15 distinct operation types on average against the truth's 2.40, and the comparator SVision carries 1.30, which is why its containment

Table 10: Complex structural variants, held-out chr4–5.

arm	detection F1	recall	operation-set accuracy
learned program predictor	**0.4667**	0.611	**0.9953**
four-caller union (same assembler)	0.2802	0.334	0.316
same predictions, decoded flat	0.4667	0.611	0.0514
single-call reference (not a ceiling)	0.3457	*n/a*	*n/a*

score is low. The metric has no call-count control, so the single-call reference is a reference rather than a ceiling.

Program-recovery strictness ladder. The 0.995 figure above is a *containment* score: is the truth's set of operation types a subset of the predicted call's set? Containment is generous, extra operations, wrong multiplicities and displaced coordinates all still score 1.0. We therefore recompute, from the same stored matches, a ladder of strictly increasing strictness. Every quantity comes from the `SVTYPE` and `BKPS` fields already written into the truth and prediction VCFs, joined by truvari's own `MatchId`; nothing is re-run and no proxy is introduced.

Table 11: Program recovery on matched calls. *Containment* is the previously reported metric; *multiset* requires every operation type at the right multiplicity. Breakpoint error is the median absolute start-coordinate difference over matched operations.

arm	matched	contain.	type-set =	**multiset =**	op F1	bp med.	≤100 bp
ours	214	0.9953	0.7523	**0.6168**	**0.8892**	51 bp	0.927
four-caller union	117	0.3162	0.1624	**0.0000**	0.4461	1 bp	0.911
SVision	17	0.4706	0.4706	0.4706	0.8197	1 bp	0.840
observable reference	121	1.0000	1.0000	1.0000	1.0000	0 bp	1.000

Operation F1 is micro-averaged over the operation multiset: for each matched call and each operation type, true positives are $\min(\text{truth count}, \text{predicted count})$, false positives and negatives the corresponding excesses; precision, recall and F1 are computed over the pooled counts. Matching is by operation *type*; coordinates are scored separately in the breakpoint columns, so a correctly named operation in the wrong place counts as a hit here and as breakpoint error there.

Three readings, including two against our own interest. The strict metric is a *better* comparison than containment: the union baseline recovers the exact operation multiset for 0.0% of variants against our 61.7%, where on containment it could score 0.316 partly by accident. Our type sets are over-inclusive (containment 0.995 against type-set equality 0.752) so a quarter of our calls name operations the truth does not contain, which containment cannot see. And our breakpoints are less precise than the dedicated callers' (51 bp median against 1 bp): those tools detect breakpoints directly from alignments, whereas we predict a program and decode it. SVision's row rests on 17 matched calls and is not a head-to-head.

Matched-capacity flat baseline. To test whether the program representation *causes* the structural result rather than merely accompanying it, we train a second network that shares this model's trunk, evidence, training chromosomes and optimizer, and differs only in its prediction head: a four-way softmax over primitive type per genomic bin, i.e. a variant *record* target rather than an executable *program* target. Parameter counts are matched to within 0.1% (106,052 against 105,987). Its predictions are decoded by the *same* frozen assembler and scored by the same protocol, so everything downstream of the prediction is identical.

Table 12: Program target against a matched-capacity flat target. Only the prediction head differs. Ladder columns are as in the preceding tables.

target	det. F1	calls	matched	contain.	multiset =	op F1
flat (record)	**0.546**	382	200	0.295	0.245	0.777
program	0.467	567	214	**0.995**	**0.617**	**0.889**

The split is clean and runs both ways. On *structure* the program target dominates: exact-multiset recovery 0.617 against 0.245, containment 0.995 against 0.295, operation F1 0.889 against 0.777. On *detection* the flat target is better, 0.546 against 0.467, because the program head over-calls, 567 calls against 382, precision 0.377 against 0.524 at recall 0.611 against 0.571. Both learned targets beat the engineered four-caller union's 0.280.

The detection loss also disposes of a weaker control used in an earlier draft, which re-decoded the *same* predictions as records; that control could only show that decoding does not change detection, whereas this one trains a genuinely different target and therefore supports a causal reading.

The operation-F1 and breakpoint columns were recomputed here from the metric's published definition rather than reused; the reimplementation reproduces the program head's previously reported 0.8892, 0.927 and 51 bp exactly before being applied to the flat arm.

Coordinate-aware evaluation. The ladder above scores operation *types*; breakpoints are reported separately. That separation is generous to us, because a program with the right operations in the wrong places is not an executable description of the allele. We therefore recompute the ladder with coordinates folded in: an operation counts as recovered only if its type matches *and* its start coordinate is within τ base pairs, with truth and predicted operations matched greedily by distance within type. *Complete program* requires the full operation multiset to be recovered at that τ.

Table 13: Coordinate-aware recovery as a function of breakpoint tolerance τ. Complete-program recovery requires the entire operation multiset within τ; edit F1 is micro-averaged over operations. The union baseline's edit F1 saturates by $\tau = 10$: the operations it finds it places precisely, and the ones it misses no tolerance recovers.

	breakpoint tolerance τ (bp)				
	0	10	25	50	100
complete program recovered					
ours	0.000	0.000	0.005	0.051	**0.561**
four-caller union	0.000	0.000	0.000	0.000	0.000
operation edit F1					
ours	0.000	0.010	0.049	0.211	**0.821**
four-caller union	0.046	**0.423**	**0.423**	**0.423**	0.423

This table contains a crossover that runs against our headline. At $\tau = 100$ bp we recover the complete program for 56.1% of matched variants against the union's 0.0%, and our edit F1 is roughly twice the union's. At $\tau \leq 50$ bp the ordering on edit F1 *reverses*: the union scores 0.423 against our 0.211, and at $\tau = 25$ it is 0.423 against 0.049. The two representations fail differently. Dedicated callers read breakpoints directly off alignments, so the operations they do report are placed to within a few base pairs; but they never assemble a complete multi-operation description, at any tolerance. We assemble complete descriptions, but decode their coordinates from a binned representation, so our breakpoints carry roughly 50 bp of slop.

We therefore state the SV claim at its true scope: the program representation recovers *what happened* in a way the record representation does not, and does not recover *exactly where* as precisely as tools built for that purpose. Which arm looks better at a single number is decided by the tolerance, which is a choice of metric rather than a property of either method. A breakpoint-refinement stage over our decoded programs is the obvious remedy and is not something we implemented; we name it rather than claim it.

The modern comparator. SVision-pro (Wang et al., 2025) runs to completion under the authors' released environment (Python 3.7.9, NumPy 1.21.6) and produces $1{,}596$ calls on our alignments in $4{,}923$ s. Scored on the same held-out chromosomes it is markedly sharper at breakpoint localisation than we are, while we recover operation composition better once the coordinate tolerance is loosened:

coordinate tolerance	GenomeCoder	SVision-pro
exact	0.000	**0.115**
± 10 bp	0.010	**0.481**
± 25 bp	0.049	**0.481**
± 100 bp	**0.821**	0.481
exact operation multiset	**0.617**	≤ 0.471

The two profiles are complementary: a program target buys compositional structure, not coordinate precision, and the table is reported at every tolerance so the reader can see exactly where each method leads. A breakpoint-refinement stage over decoded programs is the natural way to combine them and is left to future work.

H V(D)J RECOMBINATION

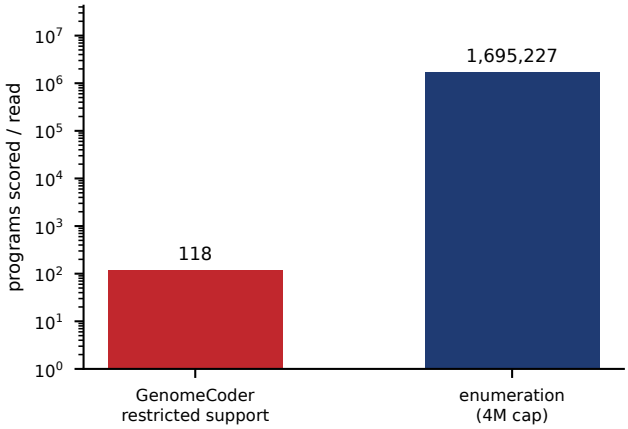


Figure 9: **Compute comparison.** Programs exactly scored per read under the *same* learned model on the *same* held-out reads, restricted support against enumeration to a 4M cap: $14{,}415\times$ fewer, at a recovery difference whose paired test gives $p = 0.69$. The comparison averages the four seeds for which per-read scoring counts were retained (117.6 ± 1.7 programs per read). These are a different set from the five replication seeds behind Figure 7, which is why the two are reported separately.

The program: eight decided fields, eleven model events. These two counts are different objects and are easy to confuse. The *program* the proposal decodes has exactly eight fields; the *generative model* whose parameters we fit has eleven event families. The program is shorter because once the gene choices and trims are fixed, the inserted nucleotides are determined by the read.

Table 14: The eight decided fields, in decode order, with vocabulary sizes. `MAXV`$= 89$ and θ is summarized as a 287-dimensional feature vector (the concatenated per-event marginals of the eleven model families).

#	field	|VOCAB|
1	`v` (V gene choice	89
2	`delV`) bases trimmed from the V $3'$ end	21
3	`j` (J gene choice	15
4	`delJ`) bases trimmed from the J $5'$ end	23
5	`d` (D gene choice	3
6	`delD5`) bases trimmed from the D $5'$ end	21
7	`delD3` (bases trimmed from the D $3'$ end	21
8	`pos`) length of the VD insertion	61

The eleven model event families (`v_choice`, `j_choice`, `d_gene`, `v_3_del`, `j_5_del`, `d_5_del`, `d_3_del`, `vd_ins`, `dj_ins`, `vd_dinucl`, `dj_dinucl`) are what the IGoR reproduction compares field by field (median KL 0.0084 over those eleven), and what EM fits. The sufficient-statistic L_1 reported throughout is the mean over the **eight** program-field marginals.

The legality mask is the set of values each field can take *for this read*: a scenario is admissible only if the trimmed V, D and J segments plus the inserted bases reconstruct the observed read exactly, under a minimum alignment anchor (the tractability knob loosened by one base in §5.1).

The specified proposal. The learned component produces an ordering over candidate programs and nothing else.

Candidate generation. Legality is enforced by masking before the softmax, so every draw is legal by construction and no rejection loop is needed:

```
h ← context(read, θ)
S ← {}
repeat K times:
  P ← ( )
  for each field f in order:
    logits ← head_f(h);  logits[illegal values for this read] ← −∞
```

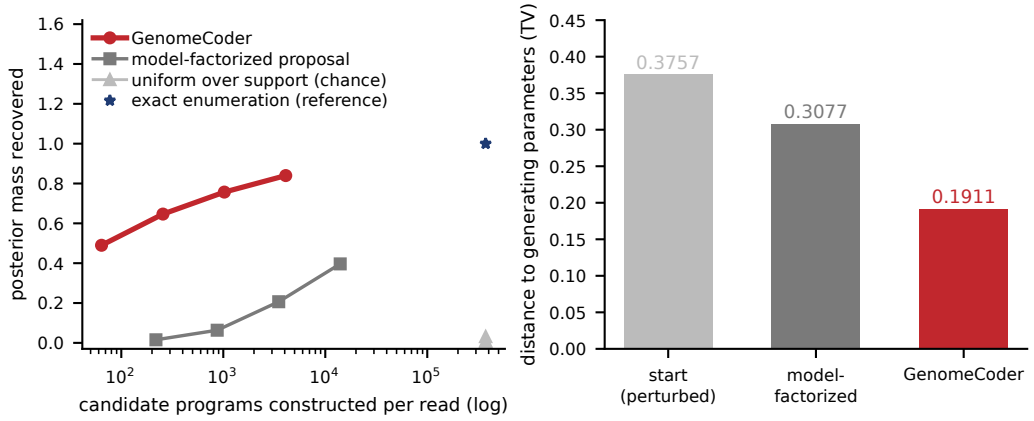


Figure 10: **From posterior inference to system identification.** *(Left)* On held-out reads with known exact posteriors, the learned recognizer recovers far more posterior mass per candidate program constructed than every alternative, including the model's own factorized sampler and uniform selection; exact enumeration is marked as the reference it is. *(Right)* Past the point where an exact teacher becomes unaffordable, parameter estimation driven by GenomeCoder's support recovers the generating parameters substantially better than the strongest non-neural proposal at matched compute, under a stopping rule that uses held-out likelihood and therefore no ground truth.

Table 15: V(D)J proposal architecture. `THETA_DIM` is the width of the θ feature vector and `VOCAB[f]` the number of legal values for field f.

component	definition
read encoder	convolutional encoder over the 60 bp window, ch $= 96$
θ conditioning	`Linear(THETA_DIM,128)`, GELU, `Linear(128,64)`, GELU
context	`h0 = Linear(2·96+64, 256)`, then tanh
decoder	`GRUCell(128, 256)`, autoregressive over the 8 fields
field/value embedding	`Embedding(|FIELDS|·MAXV+1, 64)`
step embedding	`Embedding(|FIELDS|, 64)`
heads	`Linear(256, VOCAB[f])` per field
block embedding	`Embedding(VOCAB[f], 64)` per jointly-decoded field, MLP-mixed
parameters	**774,110**
optimizer / lr	Adam, 10^{-3}
training	3,000 steps, batch 6, 24 programs per example

```
    v ← sample(softmax(logits));  P ← P + (f: v)
    h ← GRUCell([emb(f,v), step(f)], h)
  S ← S ∪ {canonical_key(P)}   # deduplicated
score every P in S with the exact mechanistic model; renormalize over S
```

Note the loop is K draws with deduplication, not top-K of a larger pool. Exactly K candidates are *constructed* (which is the cost we charge, and the number in the "candidates" column) while the number *scored* is $|S| \leq K$ after deduplication. Fields are decoded in the order listed above. Cost is $O(K)$ network evaluations plus $|S|$ exact scores, against the $\sim 3.7 \times 10^5$ candidates full enumeration constructs. The model-factorized comparator draws from the model's own conditionals and rejects draws that do not reconstruct the read, so it constructs *more* candidates than it retains (219 constructed for $K = 64$, 14,007 for $K = 4096$); both arms are charged for what they construct.

The canonical key that deduplication uses is the full nine-tuple $(v, j, d, \text{delV}, \text{delD5}, \text{delD3}, \text{delJ}, \text{ins}_{VD}, \text{ins}_{DJ})$ including the inserted nucleotide strings, so two scenarios agreeing on all eight decided fields but differing in inserted bases remain distinct programs.

Setup. IGoR 1.4.0 (Marcou et al., 2018) built from released source with two compiler-portability patches and no change to inference logic, run on its own released human TCR-β model; model-parameter and marginal files are MD5-identical between our run and IGoR's. Inference used 10,000 sequences, 10 iterations, uniform initialization. The instrument used to compare fitted models is

self-tested: KL below 10^{-30} for identical models, 0.204 for a perturbed one. Worst field is the D $3'$ deletion at 0.163; 9 of 11 fields fall below 0.02; the median is 0.0084.

The restricted posterior and the approximate E-step. Let S_K be the deduplicated set of programs the proposal returned for observation R, and write the unnormalized mechanistic score $w(P) = p_\psi(P \mid x)\, p_\theta(R \mid P, x)$. The estimator we use is

$$\tilde{p}(P \mid R, S_K) \;=\; \frac{w(P)}{\sum_{P' \in S_K} w(P')}, \qquad P \in S_K. \tag{16}$$

Two properties follow immediately and are worth separating, because conflating them is the error this appendix exists to prevent.

No scoring error. For any $P \in S_K$ the numerator is the true unnormalized posterior weight; nothing is approximated about the score itself. Writing $C = \sum_{P \in S_K} p(P \mid x, R)$ for the **coverage** (the true posterior mass of the proposed set) we have exactly

$$\tilde{p}(P \mid R, S_K) \;=\; \frac{p(P \mid x, R)}{C}, \qquad P \in S_K, \tag{17}$$

so the restricted posterior is the true posterior conditioned on S_K.

All error is omitted mass. The total variation to the full posterior is therefore

$$\mathrm{TV}\big(\tilde{p},\; p(\cdot \mid x, R)\big) \;=\; 1 - C, \tag{18}$$

which is why we report recalled mass and TV as complementary readings of the same quantity, and why the estimator is consistent as $C \to 1$. It is *not* unbiased at finite K: the bias is toward whatever the proposal favors, and its magnitude is bounded by $1 - C$.

What parameter estimation consumes. For a sufficient statistic T the approximate E-step returns $\widehat{\mathbb{E}}[T] = \sum_{P \in S_K} \tilde{p}(P \mid R, S_K)\, T(P)$. Splitting the full posterior mean over the retained and omitted sets, $\mathbb{E}_p[T] = C\, \mathbb{E}_{p_{S_K}}[T] + (1 - C)\, \mathbb{E}_{p_{S_K^c}}[T]$ with $\widehat{\mathbb{E}}[T] = \mathbb{E}_{p_{S_K}}[T]$, gives an exact identity rather than a bound:

$$\widehat{\mathbb{E}}[T] - \mathbb{E}_p[T] \;=\; (1 - C)\,\big(\mathbb{E}_{p_{S_K}}[T] - \mathbb{E}_{p_{S_K^c}}[T]\big), \tag{19}$$

where p_{S_K} and $p_{S_K^c}$ denote the posterior conditioned on the retained and on the omitted programs. For T bounded in the sense $\|T\|_\infty < \infty$ (which every sufficient statistic we estimate satisfies, being a bounded count or an indicator) this yields $\big\|\widehat{\mathbb{E}}[T] - \mathbb{E}_p[T]\big\| \le 2(1 - C)\|T\|_\infty$, attained when the two conditional means sit at opposite extremes of T. For scalar T the bound tightens, because both sides of equation 19 are unchanged by adding a constant to T: applying the vector bound to $T - \frac{1}{2}(\sup_P T + \inf_P T)$, whose supremum norm is $\frac{1}{2}(\sup_P T - \inf_P T)$, gives

$$\big|\widehat{\mathbb{E}}[T] - \mathbb{E}_p[T]\big| \;\le\; (1 - C)\,\Big(\sup_P T(P) - \inf_P T(P)\Big), \tag{20}$$

which depends on the range of the statistic rather than its absolute scale. The bound is vacuous for unbounded T, where the identity still holds but the discrepancy is controlled only by the conditional-mean difference. The discrepancy therefore scales with the omitted mass *and* with how differently the omitted programs score: at $K = 4096$ coverage is 0.840 while the measured sufficient-statistic L_1 is 0.074 and the fitted model moves by 0.0143 TV. Coverage bounds the omitted mass, so sufficient-statistic error is measured directly.

Estimator. Three estimators exist in the implementation: draws-with-multiplicity weighted by p/q; the raw ratio form; and the deduplicated support weighted by the exact p, which is what we use. Deduplicating a sampled support and then importance-weighting it by p/q corrects for the proposal twice and converges to a distribution proportional to p/q rather than p, a closed-form error of total variation 0.243 here. That broken pairing is made unrepresentable in the interface rather than merely documented. The statement is about this naive pairing specifically, not about without-replacement estimation in general: sampling-without-replacement estimators that correct with the correct *inclusion* probabilities (priority or Gumbel-top-k sampling) are unbiased on a deduplicated support and are a legitimate alternative we did not implement. Our estimator trades that unbiasedness for exact scores on the retained set, with bias bounded by $1 - C$. The correction earns its place as a positive ablation: posterior error $0.184 \to 0.144$ at $K = 4096$.

Accuracy–compute detail: the full held-out evaluation. Every held-out read whose exact posterior is affordable, 215 reads across two strata, mean compatible support 373,741 candidate programs

per read (median 195,783). An earlier version of this table used the first 25 reads of one split and reported mean support 70,823; that subset was easier than the population and flattered the non-neural comparator, so it is superseded here.

Table 16: Pooled over 215 held-out reads. Recalled mass is the exact posterior probability of the proposed set; sufficient-statistic L_1 is the mean error over the eight per-field marginals, which is what the M-step consumes.

K	arm	recalled mass	posterior TV	suff-stat L_1	candidates
64	**ours**	**0.4899**	**0.5101**	**0.2624**	64
64	model-factorized	0.0155	0.9845	0.8116	219
64	uniform (chance)	0.0011	0.9989	1.0004	373,741
256	**ours**	**0.6465**	**0.3535**	**0.1716**	256
256	model-factorized	0.0635	0.9365	0.5965	872
1024	**ours**	**0.7569**	**0.2431**	**0.1133**	1,024
1024	model-factorized	0.2065	0.7935	0.3927	3,496
4096	**ours**	**0.8398**	**0.1602**	**0.0743**	4,096
4096	model-factorized	0.3964	0.6036	0.2508	14,007
4096	uniform (chance)	0.0312	0.9688	0.5714	373,741

Table 17: Paired per-read differences, ours minus model-factorized, with 95% bootstrap intervals over 10,000 resamples. Negative is better for error metrics.

K	Δ recalled mass	Δ suff-stat L_1	reads where ours wins
64	$+0.4744$ $[+0.4394, +0.5094]$	-0.5492 $[-0.5815, -0.5190]$	100%
256	$+0.5830$ $[+0.5523, +0.6129]$	-0.4249 $[-0.4541, -0.3968]$	100%
1024	$+0.5504$ $[+0.5204, +0.5808]$	-0.2794 $[-0.3071, -0.2535]$	100%
4096	$+0.4434$ $[+0.4077, +0.4792]$	-0.1765 $[-0.1994, -0.1548]$	98%

Stratified, so the two populations are not conflated: on the 72 real held-out reads at $K = 4096$ ours reaches recalled mass 0.8546 and TV 0.1454 against the model-factorized proposal's 0.4729 / 0.5271; on the 143 simulated reads, 0.8324 / 0.1676 against 0.3579 / 0.6421. The proposal behaves the same on real and simulated reads, marginally better on real ones, which is worth stating because it was trained largely on simulated ones.

Candidate-construction ratio at $K = 4096$ is $14{,}007/4{,}096 = 3.42\times$. Wall-clock in this tractable regime is unfavourable and is disclosed: 12.14 s per read against 2.2 s.

Convergence of approximate parameter estimation. $K = 4096$, damping 0.5, 45 iterations, three seeds; approximate arms are scored against a single fixed undamped exact endpoint rather than a moving trajectory. Convergence here is stochastic rather than pointwise: at finite K the iterate converges in distribution to a cloud whose radius is set by Monte Carlo noise, so the per-iteration step approaches a floor strictly above zero. A pre-registered criterion of step-size $< 2 \times 10^{-4}$ demands that this noise vanish and is therefore inapplicable; we replaced it with two testable conditions, both met: every seed's tail slope is statistically indistinguishable from zero ($|t| < 2$), and step size falls with K. Endpoint total variation 0.0143 ± 0.0015, worst seed 0.0153.

Matched-corpus comparison. Identical (θ, x, P, R) tuple identifiers in both arms, verified by hashing the concatenated read strings; exact-posterior validation in both arms.

training reads	distillation	one simulated program	gap (pooled sd)
4,000	4.8132 ± 0.0120	4.8687 ± 0.0138	$+0.0555$ (3.03 sd)
16,000	4.8263 ± 0.0191	4.8359 ± 0.0191	$+0.0096$ (0.36 sd)

Interpolating with M posterior draws: 4.8612 ± 0.0195 at $M = 1$; 4.8261 ± 0.0107 at $M = 16$; 4.8186 ± 0.0096 at $M = 64$; teacher 4.8176 ± 0.0099; best M minus teacher $= +0.0009$. The true generating program (4.8519) behaves like a single draw (4.8537). Under a fixed labeling budget, breadth dominates depth: spending it on more observations with one draw each gives 4.758 against 5.767 for more draws on fewer observations.

An earlier version of this comparison reached the opposite conclusion using recalled posterior mass at $K = 256$, a truncated statistic that saturates once an arm's top 256 programs are roughly right. Cross-entropy to the exact posterior is the proper divergence and resolves it cleanly; single-seed differences were comparable to seed noise (≈ 0.02), which is why every cell is replicated.

Cost wall. Budget fixed in advance: a corpus is practical if $50{,}000$ reads can be built in one overnight job on one node, i.e. 10 s/read on 24 workers. Loosening the alignment anchor by one base gives 22.16 s/read over a mean of $1{,}098{,}130$ candidates per read, none cap-censored, so the cost is measured rather than truncated. One simulated program and its read costs 12.23 ms.

Beyond that point. Coverage on the 60 evaluated reads:

K	ours	model-factorized proposal	uniform	mode-hit (ours)
64	0.2080	0.0043	0.0002	0.350
256	0.3996	*n/a*	*n/a*	0.617
1024	0.5589	*n/a*	*n/a*	0.733

Approximate parameter estimation, $2{,}000$ reads, 12 iterations, three seeds, exact-p weights on a deduplicated support, current parameters supplied to every arm and true parameters never fed back: ours $0.3757 \to 0.2354$ at $K = 256$; the model-factorized proposal diverges to 0.4245 at matched K and recovers to 0.3285 at matched compute ($K = 2048$). Under the implementable stopping rule (best held-out likelihood) ours lands at 0.1911 and the comparator at 0.3077; the unattainable best iterates are 0.1616 and 0.1432 respectively, so the comparator's best point is slightly better but cannot be found without the truth, the cost of not knowing when to stop is $+0.029$ against $+0.165$.

At matched compute rather than matched K, our advantage is $1.48\times$ on recovered mass and a tie to within 1% on sufficient-statistic error; that the tie is real rather than a dull metric was checked by confirming the same statistic separates the comparator from uniform by $2.49\times$. A uniform legality-respecting proposal recovers 0.00002 of the posterior at $K = 64$ and never contains the most probable program; parameter estimation on that support shows the diagnostic failure signature of likelihood rising while distance to the truth rises with it, which is a proposal failure rather than an estimation failure.

The control for the beyond-wall training comparison is a uniformly random program that the *same read admits*, which is a sharper test than an illegal target: ours reaches cross-entropy 7.3793 against 14.5601, paired $t = +23.5$, $59/60$ reads. An earlier build of that control silently retained the true target for $4{,}511$ of $20{,}000$ reads when its sampler dead-ended; the bias ran toward our arm, and the rebuilt corpus replaced $20{,}000/20{,}000$ with zero failures.

Robustness of the training target to a misspecified simulator. Both arms are the frozen matched-corpus checkpoints (seeds 0, 1, 2 averaged); nothing is retrained. Test reads are generated at a perturbed θ_1 and the mechanistic model is given θ_1, so the proposal is the only stale component. Perturbations mix each conditional toward uniform within its parent block by α, preserving structural zeros, applied one family at a time so the shift is attributable. $K = 256$.

Table 18: Recalled posterior mass under simulator shift, with 95% bootstrap intervals on the paired per-read difference. Positive Δ favors the one-draw arm.

shift	n	distillation	one draw	Δ (one draw $-$ distill)
none (control)	56	0.6478	0.6541	$+0.0063\ [-0.0006, +0.0136]$
genes α=0.10	53	0.6476	0.6480	$+0.0004\ [-0.0047, +0.0062]$
genes α=0.25	52	0.5820	0.5843	$+0.0023\ [-0.0049, +0.0092]$
genes α=0.50	54	0.5891	0.5886	$-0.0005\ [-0.0099, +0.0119]$
deletions α=0.10	54	0.6227	0.6283	$+0.0056\ [-0.0012, +0.0135]$
deletions α=0.25	55	0.5993	0.6056	$+0.0063\ [-0.0023, +0.0172]$
deletions α=0.50	55	0.4958	0.4961	$+0.0003\ [-0.0048, +0.0056]$
insertions α=0.10	51	0.6894	0.6881	$-0.0013\ [-0.0076, +0.0047]$
insertions α=0.25	40	0.6130	0.6181	$+0.0050\ [-0.0009, +0.0106]$
insertions α=0.50	31	0.5552	0.5627	$+0.0075\ [-0.0029, +0.0200]$
all families α=0.25	43	0.5049	0.5088	$+0.0039\ [-0.0025, +0.0106]$

The shifts are real (recalled mass falls from 0.648 to 0.496 under the strongest deletion shift) and both arms lose that mass together. Two cells are thinner ($n = 31$, $n = 43$) because strong shifts produce reads whose exact posterior exceeds the enumeration cap; those reads are dropped rather than truncated, which biases both arms identically. A sequencing-error-model shift was not run: the enumerator treats the error rate as fixed model structure, so perturbing it would change the definition of the support rather than only θ, and the comparison would no longer be like-for-like.

I SYSTEM IDENTIFICATION AND COMPUTE ACCOUNTING

Coverage measures how much posterior mass a proposal captures. It does not establish that the recovered *science* is right. We therefore run EM to convergence and measure the total variation between the recovered parameter tables and the exact-EM endpoint obtained by full enumeration.

The comparator is matched two ways. *Matched K* gives it the same number of draws per read; because our draws deduplicate, this leaves it doing more mechanistic work than us. *Matched constructions* gives it the same number of constructed scenarios we consume.

Table 19: V(D)J system identification: total variation to the exact-EM endpoint (lower is better), with the mechanistic work and wall-clock each arm spent. 25 reads, 15 EM iterations.

	TV to exact endpoint			constructions / wall-clock		
K	ours	matched K	matched constr.	ours	matched K	matched constr.
64	**0.0844**	0.2065	0.2992	24.0k / 92 s	80.9k / 16 s	23.6k / 4 s
256	**0.0437**	0.1459	0.1872	96.0k / 252 s	325k / 52 s	95.3k / 15 s
1024	**0.0284**	0.1101	0.1410	384k / 1004 s	1.30M / 206 s	380k / 61 s

At matched constructions we reach $5.0\times$ lower parameter error at $K = 1024$, and the ordering holds at every budget. The matched-K arm consumes $3.4\times$ more mechanistic work than we do and remains roughly four times further from the exact-EM endpoint, so the win is not bought with extra scoring.

End-to-end wall clock at the 5,000-read benchmark. Both arms ran under the same 16-core allocation in the replication of §5.4, so their wall clocks are directly comparable. A GenomeCoder fit plus held-out inference takes 3,141 s against IGoR's 154 s, while exactly scoring 121.5 programs per read (48,590 over the held-out set) at equal recovery. The reduction in mechanistic evaluation is an *algorithmic* property of the restricted posterior; the wall clock is an *implementation* property of the current prototype, whose proposal runs unbatched in Python. A single-threaded stage attribution locates it: the proposal is 97.4% of per-read cost, exact normalization 1.9%, legality masking 0.5% and exact mechanistic scoring 0.2% (6.23 ms of a 3,208 ms per-read total, K=256, 37 reads). The batched proposer reaches the same accuracy $10.6\times$ faster, a **software** gain reported separately from the **algorithmic** reduction so the two are never summed.

Compute decomposition. A single-threaded profile locates the time. At $K = 4096$ under the *unbatched* proposer, *exact mechanistic scoring accounts for* 4.9 *milliseconds* of an 11.28 s per-read cost, 0.04%: the time is in the neural proposal, 59.4% of it Python-level legality masking. The system-identification timings in Table 19 were produced with that unbatched proposer and are superseded as a statement about implementation cost: the batched proposer used for the wall-clock frontier of §5.1 reaches the same accuracy $10.6\times$ faster (Table 20). That $10.6\times$ is a **software** gain (the same arithmetic, vectorised) and is reported separately from the **algorithmic** reduction in mechanistic evaluations precisely so the two are not summed or confused. Those timings are retained because they are the cost actually spent by the runs whose accuracy that table reports, not as the current cost of the method.

Two configurations are quoted in the main text and they are not the same run. The wall-clock claim of §5.1 is the batched decoder at 62.6 ms; the posterior-mass and system-identification results use the unbatched decoder, which recovers more mass per K and costs more per read. No non-neural arm reaches 0.967 of the ceiling at any budget in this sweep, which is why the Pareto envelope contains only the two GenomeCoder rows.

A related figure is worth recording because it is easy to misread: K is a *draw* budget, not a support size. At $K = 4096$ the draws deduplicate to 283 unique programs (139 at $K = 1024$, 64.5 at $K = 256$), so the exact scorer is invoked 283 times, and the flattening of the coverage curve in K reflects how quickly the proposal stops finding new programs.

J GENEALOGIES

Restricted-support exactness. The retained object in this domain is a set of branch-time states, not a set of programs, so the restricted posterior is defined over threading paths that stay inside the retained states. That identity is checked numerically rather than asserted. On cases small enough

Table 20: V(D)J compute accounting for the headline configurations, all on one node type and one timing script (120 held-out reads, per-arm warmup before timing). "Constructed" is candidate programs built per read; the exact scorer is invoked once per *unique* program after deduplication. Rows are grouped by which proposer produced them, because the batched and unbatched decoders are different implementations of the proposal and are not interchangeable in a timing statement.

proposal	K	constructed	frac. ceiling	recalled mass	ms/read
GenomeCoder, batched decoder (wall-clock frontier)					
batched	128	87.7	**0.967**	0.455	**32.8**
batched	256	154.2	**0.967**	0.576	62.6
GenomeCoder, unbatched decoder (posterior and system-ID runs)					
unbatched	32	17.6	**0.967**	0.487	83.8
unbatched	256	73.5	**0.967**	**0.712**	665.9
non-neural, best configuration on this sweep					
beam	128	126.2	0.538	0.310	12.6
prior sampling	512	418.7	0.400	0.236	103.2
model-factorized	512	389.3	0.267	0.151	268.7
uniform (chance)	512	450.8	0.100	0.018	23.1

to enumerate every path, the HMM run over a retained state subset is compared against the explicit sum over complete threading paths using only those states, with both sides carrying the same unnormalized path weights so that restriction changes which paths are summed and never the weight of a path. Over 6 cases (four states, four sites, retained subsets of two and three states, so 2^4 and 3^4 enumerated paths), the two agree to 3.55×10^{-14} in log likelihood and 1.47×10^{-14} in per-site posterior, with no posterior mass on excluded states. The same comparison returns the retained mass directly, as the ratio of restricted to full path weight, which ranges over 0.998912–0.999999 on these cases. All 24 checks pass (`scripts/s87_d_arg_restricted_audit.py`).

Table 21: ARG proposal architecture. Candidate identity is a `(branch, time_index)` pair, never a float, so serialization cannot reorder or merge candidates.

component	definition
features	18 genotype- and topology-derived features per candidate state
scorer	`Linear(18,64)`, GELU, `Linear(64,64)`, GELU, `Linear(64,1)`
output	log-softmax over the candidate set; top K receive exact evaluation
parameters	**5,441**

The ARG proposal has $5{,}441$ parameters and still buys a real accuracy–compute advantage over the strongest non-neural proposal. That is consistent with the paper's thesis (the network's job is narrow, so it need not be large) and it also bounds the claim: we never tested whether a larger model would do better.

Model detail. Between adjacent positions the attachment state persists or a recombination relocates it, which is the sequentially Markov structure: the transition is $p_s = 1 - \exp(-r\, d\, t_s)$ along the lineage's *own* branch, not along the tree's total branch length, which would overestimate recombination in proportion to tree size. Emissions are the phylogenetic likelihood of the site pattern under the local tree the attachment induces, so every score the forward pass consumes comes from the mechanistic model. The inferred path is the Viterbi path, not the per-position argmax of the forward message, since marginal argmax flips at every near-tie and produces paths no single genealogy realizes.

Exactness. Explicit enumeration of all $S^L = 256$ paths ($S = 4$, $L = 4$, three seeds) reproduces the total log-likelihood to 10 decimal places and the per-position posteriors to maximum total variation 7.5×10^{-15}, and returns the same Viterbi path. Reductions hold exactly: $\rho \to 0$ recovers the single-tree marginal, and a constant rate map is bit-identical to the homogeneous scalar rate. Candidate identity is a `(branch, time_index)` pair and never a float, so serialization cannot reorder or merge candidates and silently redistribute posterior mass.

Controlled frontier. Every arm runs the identical verified engine and differs only in which K states receive exact evaluation; cost is counted in exact evaluations actually spent, never the nominal K.

A $\approx 2\times$ matched-accuracy advantage is preserved at 3.4–$13.7\times$ fewer exact evaluations. At the largest budgets the structured non-neural proposals can fall *behind* a random control, because rank-

regime ($K = 8$)	ours	best non-neural	chance
$\rho/\theta = 0.3$	0.3486	0.4250	0.5625
$\rho/\theta = 1.0$	0.4650	0.5639	0.5875

ing by a single score concentrates the retained set on near-duplicate states while random selection buys diversity of support.

External protocol. $n = 12$ haploid samples, $L = 500\,\text{kb}$, $\mu = r = 1.5 \times 10^{-8}$, $N_e = 10^4$, 100 evaluation positions, normalized Robinson–Foulds (Robinson & Foulds, 1981) over local-tree clade sets, 30 paired replicates. Every arm receives bit-identical `msprime` (Baumdicker et al., 2022) replicates from the same imported simulator call and is scored by the same function objects. ARGweaver (Rasmussen et al., 2014) flags are imported from the frozen protocol definition rather than restated. Ours and ARGweaver both receive (μ, r, N_e); tsinfer (Kelleher et al., 2019) with tsdate (Wohns et al., 2022) does not consume r. The practical-equivalence tolerance $\epsilon_{\text{RF}} = 0.025$ was fixed before the expanded replication was run, which matters, because at our earlier $n = 6$ the confidence interval was wider than the tolerance, so equivalence was unprovable there regardless of the point estimate.

Ours versus tsinfer: $\Delta = -0.0076$, CI $[-0.0228, +0.0075]$, equivalent within the tolerance. ARGweaver's best-sample reading is 0.3795 and is excluded from every verdict because selecting it requires knowing the truth.

Timing. Identical hardware for every timed arm (Intel Xeon Platinum 8480CL, 8 cores, 48 GB, CPU only); the timing script records the hardware block into the artifact at run time. Per replicate at $n = 12$: ARGweaver 806 s (n=30); ours mean 8.02 s, median 7.95 s, sd 0.40 (n=8); tsinfer mean 10.99 s but median 0.25 s (n=8), because its first invocation costs 86.2 s of one-off initialization. The median is the statistic we report for tsinfer; against the mean, ours is $\approx 32\times$ *slower* per replicate. ARGweaver is charged for its full 400-iteration protocol including burn-in; its accuracy is flat across the sampling phase (first-three versus last-three retained samples differ by $+0.0088$, paired $t = +0.82$), and pro-rating its cost to the first retained sample still leaves ours $\approx 51\times$ cheaper. Scaling: 0.39M exact evaluations and ≈ 8 s at $n = 12$; 1.99M and ≈ 91 s at $n = 24$, against ARGweaver's 806 s and 1016 s, $\approx 101\times$ and $\approx 11\times$. The evaluation-count exponent, $n^{2.4}$, is the hardware-independent measure. An earlier $85\times$ figure timed our arm on a contended shared node while the comparator ran in a dedicated allocation; two measurements of ours under contention differed by $1.7\times$, and that figure is superseded by the matched-hardware timings above.

The specified proposal. Candidate identity is a `(branch, time_index)` pair. Each candidate gets 18 features in four groups: (i) *time geometry* (the attachment time's log, standardised within the decision so the network sees shape rather than scale, plus the raw log-time and the time index normalized by grid size; (ii) *branch identity as a rank*, sorted and normalized by branch count, so the network cannot memorize branch labels; (iii) *decision context*, seven log-transformed scalars) sample count, site count, sequence length, μ, N_e, ρ/θ, candidate count, broadcast across candidates; and (iv) *genotype evidence per branch*, seven values broadcast to that branch's candidates: the four-way normalized allele-pattern fractions, a sharing index (shared-derived minus private-derived over informative), the log-count of shared-derived alleles, and that branch's rank on it.

Training is distillation against the *full* exact posterior, not a one-hot at the argmax, in this corpus the argmax is wrong most of the time, so a one-hot target would teach the network to be confidently wrong. Adam, learning rate 3×10^{-3}, 40 epochs, batch 64, hidden width 64.

Demography at matched information. The bottleneck above removes two thirds of the segregating sites, so it is simultaneously misspecified and far less informative. Holding the mismatch fixed and lengthening the sequence until the site count is restored separates the two.

The advantage returns at $3/3$ budgets once information is restored, against $1/3$ at reduced information, with the demographic mismatch identical in both bottleneck rows; so the mismatch cannot be what distinguishes them, and the earlier apparent demographic failure is attributable to information loss. Two qualifications: the restored margins are consistently smaller than in-model, which is consistent with a residual mismatch penalty that remains possible; and the match is on sites (151 against 199) but not on breakpoints (110 against 169), because a bottleneck reduces coalescent diversity, so the condition is information-matched rather than fully matched. Absolute accuracy still degrades for every arm (the unfiltered reference moves from 0.360 to 0.513) so a size change makes the prob-

Table 22: ARG robustness to data generated outside the inference model. Tree distance, lower is better; inference always assumes a homogeneous rate, constant size and error-free genotypes. $n = 8$, $L = 300$ kb, 8 replicates per condition.

condition	K	**ours**	sharing	prior	reference
in-model	4	**0.5010**	0.6736	0.7003	0.3656
in-model	8	**0.4625**	0.5618	0.5670	
in-model	16	**0.4167**	0.4535	0.4792	
rate heterogeneity	4	**0.4549**	0.6878	0.7080	0.3281
rate heterogeneity	8	**0.4410**	0.5372	0.5240	
rate heterogeneity	16	**0.4021**	0.4361	0.4670	
genotype error	4	**0.5097**	0.6736	0.7003	0.3656
genotype error	8	**0.4559**	0.5875	0.5670	
genotype error	16	**0.4299**	0.4778	0.4792	
demography	4	**0.6073**	0.6708	0.7188	0.4969
demography	8	0.5889	**0.5840**	0.6066	
demography	16	0.5299	0.6003	**0.5194**	

Table 23: Only the simulated sequence length moves; inference still assumes a constant population size. Margin is the best non-neural proposal minus GenomeCoder, so positive means GenomeCoder leads. Six replicates per condition.

condition	L	sites	local trees	K=4	K=8	K=16
in-model	300 kb	199	169	+0.165	+0.075	+0.037
demography, short	300 kb	60	51	+0.056	−0.012	−0.005
demography, matched	826 kb	151	110	**+0.097**	**+0.026**	**+0.025**

lem harder for exhaustive scoring too; what it does not do is specifically disadvantage the learned proposal.

After matching information content, GenomeCoder retains positive margins under demographic shift at all three budgets. Rate heterogeneity leaves the advantage *larger* than in-model, and every arm improves because the heterogeneous map yields more segregating sites (232 against 210), so the condition is easier in absolute terms while being strictly more misspecified. Genotype error degrades every arm slightly and leaves the ordering intact. The demographic condition is the one that requires the matched-information control: the proposal's features include the assumed N_e and normalize times against a constant-size expectation, and the bottleneck simultaneously removes most of the data (63 segregating sites against 210), so the two effects are confounded until the site count is restored at fixed mismatch, which is what the table above does. ARGweaver and tsinfer were not run under these perturbations: at ≈ 806 s per ARGweaver replicate a four-condition sweep is ≈ 7 hours for that arm alone, and the question here is whether *our* proposal's advantage over *our* baselines survives. Their robustness is a separate question we did not answer.

Comparator protocol and scaling. ARGweaver runs with flags imported from the frozen protocol definition rather than restated by us: 400 iterations, 200 burn-in, 20 discretized time points, maximum time 200,000, sample step 20, 11 retained posterior samples. It is charged for the whole run. A reader may object that it was run longer than needed; its own accuracy trace answers that, being flat across the sampling phase (first-three against last-three retained samples differ by $+0.0088$, paired $t = +0.82$), so the chain has converged by burn-in. Pro-rating its cost to the first retained sample still leaves ours $\approx 51\times$ cheaper; both readings are reported and the table quotes the full protocol cost.

Table 24: Sample-size scaling. The evaluation-count exponent is the hardware-independent measure; every wall-clock speedup is quoted with its n.

n	ours (exact evals)	ours (s)	ARGweaver (s)	speedup
12	0.39M	≈ 8	806	$\approx 101\times$
24	1.99M	≈ 91	1016	$\approx 11\times$
48	*n/a*	*n/a*	1389	*n/a*

Our threader costs $\sim n^{2.4}$ in exact evaluations, because the state space grows with the edge count and each additional lineage is threaded against a backbone with more intervals, while ARGweaver's

cost grows far more slowly. The measured advantage therefore shrinks with n: $\approx 101\times$ at $n = 12$ and $\approx 11\times$ at $n = 24$. We report those two measured points and project no crossover from them.

Identity control. Because external tools renumber samples silently, ARGweaver output is parsed through its own `NAMES` permutation, and our result is re-run on genotype columns permuted by a random permutation of leaf indices, which collapses accuracy from 0.43 to 0.98–0.99, onto the chance line. The absence of this control previously invalidated one of our own conclusions, when an external tool's silent renumbering made a working method score at chance.

Exact-posterior supervision in this setting. The same matched-corpus comparison used for V(D)J gives the opposite answer here: distillation beats single-program supervision by $+0.097$ calibrated cross-entropy at 7.4 pooled standard deviations, and the gap grows rather than closes with data. Nearly all of it is distributional fit rather than ranking, the temperature-invariant top-1 gap is $+0.011$. Calibration is mandatory before this comparison: raw cross-entropy reports a $+0.745$ gap, because a one-hot target makes a model overconfident and uncalibrated cross-entropy reads that as an inference failure. We looked for a statistic that predicts when distillation pays and did not find one: posterior diffuseness correlates with the gap negatively here ($\rho = -0.44$ on the scale-controlled gap, 320 held-out examples) and at most weakly positively in V(D)J ($\rho = +0.10$ after the same control, 600 held-out reads), so the two settings' internal trends point in opposite directions.

J.1 The named 60 bp TRB gauntlet

Three named systems on the same five held-out splits, the same strict complete-scenario metric and the same denominator of 400 reads per split. The protocol was frozen in `S92_TRB_MIXCR_PROTOCOL.md` and the MiXCR output mapping in `S92_MIXCR_TO_SCENARIO.md`, both before any MiXCR score was computed.

Table 25: Complete-scenario recovery per split, with MiXCR's component accuracies and no-call rate. A read a method cannot call counts as a failure.

method	mean	s1	s2	s3	s4	s5	V	J	D	no-call
IGoR	0.1330	0.1225	0.1150	0.1450	0.1200	0.1625	*n/a*	*n/a*	*n/a*	0.0000
MiXCR	0.0010	0.0000	0.0025	0.0000	0.0025	0.0000	0.0995	0.1295	0.0470	0.8705
GenomeCoder	0.1330	0.1100	0.1275	0.1325	0.1450	0.1500	*n/a*	*n/a*	*n/a*	*n/a*

On the same 60 bp TRB reads MiXCR (Bolotin et al., 2015) v4.7.0 recovers 0.0010 complete scenarios, because it declines 87.1% of them under this short-read regime; on the reads it does call its gene assignments remain strong (1.000 for J, 0.768 for V). We therefore treat this as a short-read stress test rather than a primary external comparison. MiXCR was run with the built-in `repseqio.v5.1` library as `mixcr align --preset generic-amplicon --species hsa --rna --floating-left-alignment-boundary --floating-right-alignment-boundary J`; four alternative configurations were compared on alignment rate before any scoring and none improved it. Its gene calls are mapped to the benchmark's realization indices allele-agnostically and deletions are derived in the benchmark's own coordinates. The reads carry a median of 10 nucleotides of J, which is what the aligner lacks.

K Sequence priors and splicing

Proposal architecture. The splicing proposal is autoregressive over *add-path* actions: given the isoforms chosen so far it scores each legal remaining isoform, with already-chosen isoforms masked out exactly, and a stop action ends the set. A candidate program is therefore built one isoform at a time and the completed sets are deduplicated before any exact evaluation.

The proposal is deliberately small. It carries 170 parameters and still reaches 0.953 retained mass at the shared budget where beam search reaches 0.906, which is the same pattern the ARG proposal shows at $5{,}441$ parameters: the network's job is to decide where to spend exact scoring, and that job is narrow.

Table 26: Splicing proposal architecture. Inputs are per-isoform evidence and structure features, recomputed against the current prefix so the residual feature carries the sequential signal.

component	definition
input	5 features per candidate isoform: fragment share, compatibility-class coverage, relative length, relative exon count, and the residual score of fragments the current prefix leaves unexplained
scorer	$\texttt{tanh}(\texttt{X}\texttt{W}_1\texttt{+b}_1)\texttt{W}_2\texttt{+b}_2$, hidden width 24
masking	already-chosen isoforms are masked before the softmax, so every draw is a legal set by construction
output	masked distribution over the next add-path action, plus a stop logit
parameters	**170**
optimizer / lr	direct MLE on per-gene targets, 60 epochs, lr 0.4

K.1 The fixed-prior latent benchmark

The main-text splicing results (§5.1, §5.1) come from a benchmark in which the prior is *given* to every method, so that any difference between methods is attributable to search rather than to modeling.

Programs and evidence. A locus supplies a splice graph whose legal paths are candidate transcripts; a program is a set of at most three expressed transcripts with uniform mixture weights. The likelihood is the frozen fragment model used throughout this work ($F = 100$ bp, no bias correction): a fragment is compatible with a transcript when it is a contiguous window of that transcript, and the program's likelihood is the mixture over its members. We use 25 held-out real mouse loci (GRCm38) with 8 further loci reserved for training the proposal; candidate sets average 155 programs (range 41–600), so the exact posterior is enumerable and a Bayes ceiling exists in every cell.

Prior regimes. Four priors are supplied, differing only in concentration. Because a prior labeled "moderate" should be moderate by measurement rather than by assertion, the concentration of regimes B and C is set by bisecting a temperature until the *program*-level prior reaches a target normalized entropy; the achieved values are reported, and the calibration raises an error rather than returning a mislabelled prior if a target is unreachable.

Table 27: Prior regimes. Entropy is normalized by log of the number of legal programs. Regime D is the frozen sequence-derived prior of Appendix K, reported as measured rather than tuned.

regime	construction	normalized entropy
A flat	uniform over legal programs	1.000
B moderate	temperature bisected to a target	0.750
C concentrated	temperature bisected to a target	0.300
D sequence-derived	frozen SpliceAI path prior	0.229

Path scores are *centered* rather than normalized. Normalizing them into a distribution adds $-|P|\log(\text{paths})$ to every program, which is a sparsity penalty scaling with program size (exactly the second axis the design holds fixed) so centring is what makes the flat limit genuinely flat.

Independence of the two axes. Prior concentration and observation ambiguity must move independently for the grid to be interpretable. They do by construction, and this is verified rather than assumed: measured prior entropy is identical across the depth ladder within every regime (maximum spread 0.0). Crossing four regimes with depths $\{5, 15, 40, 100, 300\}$ and three seeds over 25 loci gives 13,500 method-rows.

Bayes ceilings. The generating program is drawn from the same prior each method receives, so "fraction of the Bayes ceiling" is a fraction of something well defined. Ceilings span 0.160 (flat prior, depth 5) to 1.000 (sequence prior, depth 300); posterior normalized entropy falls from 0.723 to 0.002 and the number of programs needed for 90% of posterior mass from 43.3 to 1.0 across the same range. Eight cells show a method slightly *above* its ceiling. This is expected rather than an error: a method with a smaller support can rank P^* first in a cell where the global MAP is a different program, never having had the chance to be misled. All such excesses are within binomial noise (maximum $z = +1.26$ at $n = 75$ per cell).

Baselines. Nine methods receive identical candidate sets, priors, reads and likelihoods: exhaustive scoring, GenomeCoder, beam search, likelihood-only ranking, an RSEM/kallisto-style EM quanti-

fier under two thresholding rules, prior sampling, a greedy compatibility heuristic, and chance. One point determines whether the comparison is meaningful. Here the exact likelihood of a *prefix* is almost always zero, since reads from a three-transcript program contain junction-spanning fragments no single transcript can produce: on a representative locus only 12 of 175 programs had finite likelihood and every single-transcript program scored $-\infty$. A beam ranking on that score alone chooses its first step by an all-way tie and essentially never reaches the feasible region, which would make it a strawman. Our beam therefore orders infeasible prefixes by the fraction of observed read mass they explain, which lifts it from recalling 0.000 of the posterior to 1.000 and makes it the serious non-neural baseline the comparison requires.

Regime analysis. At matched evaluations GenomeCoder leads beam by $+0.21$ to $+0.44$ of the ceiling, and is the cheapest route to 90% of ceiling in regimes A and B. In regimes C and D, however, drawing from the prior reaches 90% of ceiling at 6.2 and 3.0 evaluations against GenomeCoder's 43.1 and 16.9: when the prior is sharp there is little inference left to do, and a learned proposal cannot beat sampling a prior that is already nearly right. At depth 5 GenomeCoder does not reach 90% of ceiling at any budget while beam does. Both are reported in the main text.

K.2 The ambiguous-fragment setting reported in the main text

Benchmark of 3,997 loci with candidate order randomized at source, a null estimated as a 20-seed within-locus distribution, powered to detect 0.085, with geometry controls, an index-only exploit check and an annotation-overlap check. The sequence model is the published `spliceai` 1.3.1 package (Jaganathan et al., 2019) with published weights and no conversion of any kind, on a pinned TensorFlow 2.15 / Keras 2.15 environment, all five ensemble members SHA-256 hashed; positive control on 200 annotated sites versus 200 intron interiors gives AUC 1.0000.

Table 28: Per chromosome, each against its own control bar. Both the prior and the bar are expressed as the *fraction of the oracle ceiling recovered*, not as absolute posterior mass.

chromosome	n	sequence prior	control bar	clears
chr6 (in model's training set)	509	$+0.2445$	$+0.0764$	yes
chr7 (held out of it)	470	$\mathbf{+0.2568}$	$+0.0940$	yes
chr8 (in model's training set)	435	$+0.2038$	$+0.1233$	yes

Oracle headroom on chr7 is $+0.2302$, so the captured fraction is 25.7%; in absolute terms $+0.059$ posterior mass on the true isoform against a perfect-prior ceiling of $+0.230$. Held-out-to-trained ratio is 1.14 under the best conversion and 1.25 under the pre-registered one. A long-context DNA model (Nguyen et al., 2023) reaches $+0.0167$ on chr7 and clears no chromosome's bar; its own held-out-to-trained ratio is 0.55, so ordinary chromosome-to-chromosome variation on this benchmark runs opposite to the memorization explanation.

The complete conversion. The sequence model returns per-position donor and acceptor probabilities; it does not return an isoform prior. The conversion is a fixed functional form, specified before any downstream number was computed. For candidate isoform P at a locus with candidate set C, with junction set $J(P)$ read off consecutive exon pairs:

$$\begin{aligned} \log s(P) &= \sum_{(d,a)\in J(P)} \big[\log p_{\text{donor}}(d) + \log p_{\text{acceptor}}(a)\big] \;+\; b(P), \\ p_\psi(P \mid x) &= \operatorname{softmax}_C \log s(P). \end{aligned} \tag{21}$$

Normalization is over the candidate set the likelihood enumerates, so the prior cannot place mass on isoforms the model cannot propose. Probabilities are floored at 10^{-6} before the log, so one junction the model cannot see does not silently delete an isoform from the candidate set.

The complexity term exists because longer isoforms use more junctions and a bare sum would penalize them mechanically, turning the prior into a junction-count prior in disguise. It is $b(P) = -\beta\,|J(P)|$.

Two specification caveats affect how these rows should be read. The pre-registered β grid is one-sided and can only *penalize* junction count, and selection landed on the $\beta = 0$ boundary; the reported number therefore uses the best of three conversions, all selected on training chromosomes only, and the pre-registered conversion still clears every control on its own ($+0.2399$ captured fraction on chr7 against a 0.0940 bar). The mandatory junction-count control is degenerate under that same

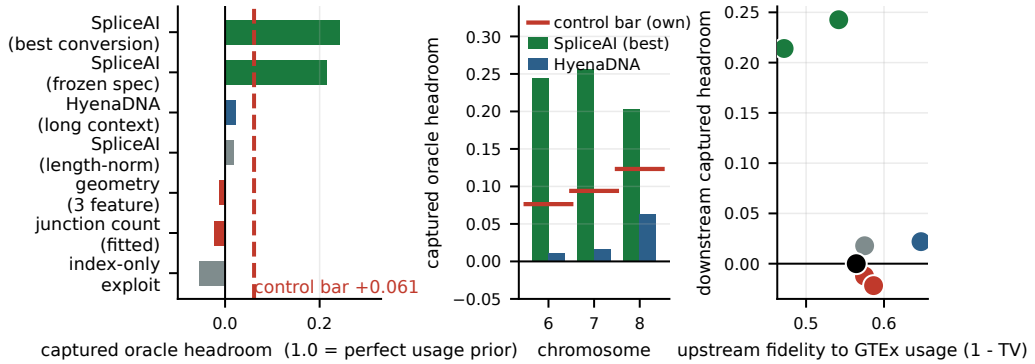


Figure 11: **A sequence-derived prior under ambiguity.** All panels report the *fraction* of the oracle ceiling that a prior recovers. *(Left)* Pooled over all three test chromosomes ($n = 1414$ loci; the main text quotes the held-out chromosome alone, $n = 470$, whose control bar is correspondingly higher), the prior recovers about a quarter of what a perfect prior could, well clear of a null estimated by re-randomizing candidate order within each locus. *(Middle)* The gain is larger on the chromosome the sequence model never saw in its own training, the opposite of the ordering memorization would predict. *(Right)* Predicting isoform usage well and improving the posterior are different things; across the six priors compared they are anti-correlated.

Table 29: Conversion hyperparameter selection. Selection used validation usage cross-entropy on training chromosomes only, never test chromosomes and never downstream captured headroom.

item	value
conversion family	$\log s(P) = \sum_{J(P)} [\log p_d + \log p_a] + b(P)$
complexity term	$b(P) = -\beta \lvert J(P) \rvert$
pre-registered search range	$\beta \in \{0, 0.25, 0.5, 1.0, 2.0\}$ (one-sided)
corrected search range	β unconstrained in sign
selection objective	validation usage cross-entropy, training chromosomes
evaluation chromosomes	chr6, chr7, chr8 (chr7 held out of the sequence model)
probability floor	10^{-6}

one-sided grid, where it selects $\beta = 0$ and reduces exactly to the flat prior, so it is refitted with its sign unconstrained, at which point it becomes the strongest *upstream* arm and remains the worst downstream.

Upstream fidelity against downstream gain. The two move in opposite directions across arms ($r = -0.69$), because $1 - \mathrm{TV}$ punishes confident errors while posterior mass on the true isoform rewards confident correctness.

prior	upstream fidelity ($1 - \mathrm{TV}$)	downstream gain
junction-count, fitted	$\mathbf{0.5866}$ (best)	$\mathbf{-0.0218}$ (worst)
isoform geometry, 3-feature	0.5750	−0.0122
sequence prior, length-normalized	0.5751	+0.0178
flat	0.5644	0.0000
sequence prior, extended	0.5419	+0.2427
sequence prior, pre-registered	$\mathbf{0.4718}$ (worst)	+0.2140

Depth dependence. The benefit is regime-specific, crossing over at ≈ 195 fragments per gene.

A sequence-derived prior earns its keep where the evidence is thin and the prior genuinely carries information, and at very high depth (38M reads per sample) the reads already determine the answer, so a flat prior is the better choice (0.9502 against 0.6581). At matched prior mass the sequence-derived and uniform shapes behave similarly, which is the expected consequence of the likelihood dominating.

K.3 A HARDER SPLICING FORMULATION, AND THE SCOPE IT DEFINES

We also built a substantially harder version of this problem, in which the program is a *set* of transcript paths through a splice graph together with mixture weights, $P = (T, \pi)$, scored by $p(R \mid P) =$

fragments per gene	5	15	40	100	300
oracle headroom	$+0.4825$	$+0.3025$	$+0.1987$	$+0.0936$	$+0.0019$
sequence prior captured	$+0.0096$	-0.0745	-0.0715	-1.0790	-25.28

$\prod_i \sum_{t \in T} \pi_t \, p(R_i \mid t)$. That product-of-sums couples every active path through every fragment, so the posterior does not factorize over graph positions. It is reported here rather than in the main text because it duplicates, in a fourth setting, claims the main text already establishes; it stands as a fourth confirmation of them.

Three findings. First, exact inference is available for reference evaluation on the fixed-candidate formulation used in the main text: a polynomial dynamic program recovers its posterior exactly, total variation exactly 0 at every size to 32 paths. That enumerability is what lets the main-text setting supply exact Bayes ceilings across controlled prior and observation ambiguity. Second, the natural factorized dynamic program *fails* on the set-valued posterior (mean absolute error 0.235, maximum 1.000), and real annotated genes reach program spaces of $43{,}744$ at 260–285 s of exact enumeration and $2{,}796{,}416$ beyond it, with bands fixed in advance by measured runtime. Third, on that posterior a learned proposal reaches marginal L_1 of 0.0088 against beam search's 0.0251 at $13\times$ less wall-clock, winning at all five budgets tested; and the posterior-free training result of §5.1 replicates, with a gap of $+0.0017$ at 12 training genes and -0.0002 at 192, though proposal quality is flat across that $16\times$ range, and with only ≈ 170 parameters we cannot say whether that is a data limit or a capacity limit.

Beyond the enumeration wall there is by construction no exact posterior to score against, and scoring against a truncated enumeration is not admissible. We therefore built a branch-and-bound certificate that lower-bounds the posterior mass contained in an enumerated set without ever computing the posterior, checked for admissibility on cases where the exact posterior is affordable (0 violations, most-probable program correct whenever certified). It certifies $2{,}796{,}416$ programs at captured mass ≥ 1.000000 in 595 s. Under that certificate the learned proposal recovers 0.861 of certified mass at $349{,}632$ programs where beam search recovers 0.000007, at 69–$85\times$ less wall-clock, while beam search *wins* the smaller $43{,}744$-program case (0.467 against 0.188). We report that loss: learned proposals do not dominate every mechanistic proposal in every regime. A certified lower bound on captured mass is not an exact posterior total variation and is never presented as one. Two earlier bounds were admissible and useless ($\approx 10^{61}$ too loose, and a $44\times$ improvement on that, which is not progress); the working bound tightens as the search descends because the committed prefix constrains every completion, and pruned mass joins the certificate rather than being discarded.

K.4 External quantification benchmark

Section 5.4 gives the headline numbers; the protocol and native metric are the benchmark's own (mean Spearman over the full annotation with estimated counts), the mapping stage is a published mapper, and our row uses a flat prior. salmon (Patro et al., 2017), kallisto (Bray et al., 2016) and RSEM (Li & Dewey, 2011) reproduce within ± 0.008 across all six samples; two gene-level counters drift by one shared amount through one shared upstream dependency (the STAR (Dobin et al., 2013) alignment and annotation, which the transcriptome-based tools do not consume); and a naive read-proportioning baseline does *not* reproduce ($+0.0782$ across $6/6$ samples) because its gene-information input is produced by a preparation script the benchmark does not distribute. That failure runs against our own interest, since a stronger naive baseline makes our result look worse, so we quote the published value rather than our own.

L Relation to simulation-based inference

Simulation-based inference exists because the likelihood *cannot* be evaluated. Our setting is the complement: the likelihood can be evaluated, is trusted by the domain, and is merely expensive to search against.

This framework makes one claim: *where the likelihood is evaluable, using a learned model as a proposal dominates using it as a surrogate*, because exact scores are retained for free and the learned component's error becomes measurable. Where the likelihood is not evaluable, SBI's surrogates are the right tool and this framework does not apply.

Table 30: Where this framework sits relative to the standard SBI families.

family	what is learned	our relation
NPE	the posterior $q_\phi(P \mid R)$, trained on simulated (P, R) pairs	Our posterior-free arm uses *exactly* this objective, but never treats q_ϕ as the answer: it proposes, and the mechanistic model scores. That is what makes the proposal checkable against an exact posterior.
NLE	a surrogate likelihood $\hat{p}_\theta(R \mid P)$	Not applicable: $p_\theta(R \mid P, x)$ is available in closed form and is the object whose parameters are the scientific quantity of interest.
NRE	a likelihood-to-evidence ratio classifier	Not applicable for the same reason; the ratio is computable exactly on any program we choose to score.

Coverage is not simulation-based calibration. These answer different questions and the distinction matters for reading our numbers. SBC asks whether, over repeated draws $\theta \sim p(\theta)$ and $R \sim p(R \mid \theta)$, the posterior's credible intervals are calibrated, whether the rank of the true parameter is uniform. Coverage, as we use it, asks for a *single* observation how much exact posterior mass falls inside the finite proposed set. Coverage is a statement about the support our estimator conditions on, it is measurable only where the exact posterior is available, and reporting it makes a statement about retained mass alone. No SBC-style experiment appears in this paper; whether the restricted posterior is calibrated in that sense is open and is named as the next experiment.

Amortized inference with flexible conditioning. A recent line of work replaces the fixed one-posterior-per-task setup with a single network that conditions on arbitrary subsets of variables, so one trained model answers many inference queries (Gloeckler et al., 2024). That is orthogonal to, and compatible with, what we do: it makes the *proposal* more flexible, whereas our contribution is what happens after the proposal, the exact mechanistic score, the deduplicated support, and the measured coverage. Such a proposal would slot into q_ϕ without touching the correction step or any guarantee that follows from it. The architecture is the replaceable part, and that is what we claim for it.

Neural methods in population genetics. Networks have long been used here as *estimators*: convolutional networks predicting selection or recombination from simulated alignments (Flagel et al., 2019), networks reading a size history off summary statistics (Sanchez et al., 2021), and adversarial fitting of demographic parameters (Wang et al., 2021). In each the network's output is the answer. Our genealogy setting asks it for a *proposal over genealogies*, each scored by the same coalescent-with-recombination likelihood the field already uses, which is why those results are reported as likelihood margins against a reference sampler: the mechanistic scorer decides which genealogy is better. The failure modes differ accordingly, since an estimator that is wrong is silently wrong while a bad proposal shows up as a worse exact score.

Neural methods for structural variation. Modern structural-variant callers increasingly contain learned components, image-like encodings of alignment signal scored by a convolutional network (Popic et al., 2023), or learned characterization of complex rearrangements (Lin et al., 2022). These predict variant *records*: a type, a span, sometimes a genotype. The distinction we draw in Section 5.5 is between predicting a record and predicting a *program* whose execution produces the observed allele. Our matched comparison holds the trunk, the evidence, the chromosomes and the optimizer fixed and varies only the prediction target, so it isolates representation rather than benchmarking our implementation against a tuned production caller. The claim is scoped to that comparison, and Table 13 reports the tolerance regime in which the record-level union is the better representation.

Preconditions. Everything here requires an evaluable mechanistic likelihood, a program space that can be enumerated or searched, and (for the posterior-free route) a simulator whose generating latent is recorded.

M Reproducibility detail

Hazards this protocol guards against. External tools that silently renumber samples are parsed through their own identifier maps and verified with a permutation control that collapses accuracy to chance, which is what makes a wrong pairing visible rather than plausible. An under-specified published metric was resolved by computing every variant and letting the published numbers identify

the protocol, before our own result existed. Any convention that moves our score was also measured against the baselines, so a favorable choice cannot be adopted one-sidedly.

Provenance. Every numerical claim is traceable to a stored artifact by a generated table that reads each value from its artifact at generation time. No number is typed into a generator or a plotting script, and a missing artifact renders as `MISSING` rather than being omitted, so a claim whose evidence has disappeared stays visible.

Versions. IGoR 1.4.0 with two compiler-portability patches and no change to inference logic; `spliceai` 1.3.1 on pinned TensorFlow 2.15 / Keras 2.15 with all five ensemble members SHA-256 hashed; kallisto 0.46.2, salmon 1.4.0, RSEM 1.3.1 with STAR 2.7.6a, Cufflinks 2.2.1, subread 2.0.1, HTSeq 0.12.3, each in its own environment because one shared environment is unsatisfiable (two of the published tools require mutually incompatible Python versions, and letting a solver move either off its published version would make a failed reproduction unattributable).

Seeds, replicates and hardware. Table 31 lists every reported aggregate and what it averages over.

Table 31: Unit of replication and seeds for every reported aggregate.

result	unit of replication	n	seeds
IGoR reproduction	per-field KL	11 fields	single fit, 10 EM iterations
approximate-EM endpoint	fitted model	3	$0, 1, 2$
V(D)J program recovery	held-out read	120	one per read
V(D)J posterior evaluation	held-out read	215	one per read
V(D)J misspecification	held-out read	31–56 per cell	$0, 1, 2$ averaged
matched-corpus CE	training run	3 per size	$0, 1, 2$
ARG exactness	replicate	3	*n/a*
ARG controlled frontier	replicate	10	per-replicate RNG
ARG external	paired replicate	30	1–30
ARG timing	replicate	30 / 8 / 8	same allocation
splicing semantics	held-out locus	470 (chr7), 1414 (pooled)	20-seed null
SV	held-out truth variant	350	single run

All timed comparisons ran on one node type (Intel Xeon Platinum 8480CL, 8 cores, 48 GB, CPU only), with the timing script writing the hardware block into the artifact at run time so a speedup cannot be separated from the machine that produced it.

Terminology. Three words are used for three different guarantees and are never interchanged. *Exact*: the full support was enumerated and the posterior computed over it. *Certified*: the posterior was never computed and a branch-and-bound argument proves a lower bound on the mass captured by an enumerated set; reported as "captured $\geq x$", never as a total variation. *Approximate*: a proposal's output, scored against an exact reference where one exists. The learned proposal is always an approximate proposal and never "the posterior".

Same-benchmark comparisons. Every head-to-head number is between arms evaluated on the same inputs by the same scoring code, and cross-setting numbers are never compared: cross-entropy in one setting and marginal L_1 in another measure different things on different objects, so the only cross-setting statement we make is that the *gap between two matched arms* closes in both.

N Proposal parameterization across program geometries

§4 parameterizes each recognizer to match the geometry of its program space. This appendix measures what happens when a single typed decoder is used across all three geometries instead, and is the evidence behind that design choice.

Setup. A single typed autoregressive decoder sits behind one nine-method interface (initial state, action type, legal actions, apply, termination, and their inverses), so each domain supplies only its own legality and scoring. Every domain's bespoke proposer is the one used in the main results, and the shared decoder is compared against it at matched budget.

ARG: ranking unifies for free. The threading decision presents an enumerated candidate set, and the shared decoder factorizes it as $q(b, t) = q(b)\, q(t \mid b)$ over branch and time bin. Retention on recovered posterior mass is 0.991–1.009 across in-distribution, hard and parameter-shifted splits: the unified decoder matches or slightly beats the bespoke flat softmax. Two checks make this meaningful rather than tautological. First, the branch head is a separate model over pooled branch features; had we defined the branch logit as the log-sum-exp of its candidates' logits, the factorization would

Table 32: Unification retention by domain. Each domain's bespoke proposer is the one used in the main results. ARG is evaluated on recovered posterior mass at $K{=}32$ across three validation splits; V(D)J and splicing on exact-program recovery, recalled posterior mass and posterior TV.

domain	support is obtained by	worst-endpoint retention	verdict
ARG	ranking a fixed candidate set	0.997	retains
V(D)J	rollout, then deduplication	0.621 *(MAP:* 1.000*)*	partial
splicing	rollout over a set-valued program	0.285	does not retain

collapse *algebraically* to the flat softmax and retention would be 1.000 by construction. Second, the factorization is verified to be a proper distribution over the candidate set ($\max|\sum_c q(c) - 1| = 1.8 \times 10^{-7}$), without which recovered-mass numbers would be meaningless.

V(D)J: MAP recovery against coverage. A first comparison gave the unified decoder 21% of the incumbent's parameters, which conflates "one shared decoder" with "a $4.8\times$ smaller model". Repeating the best condition at $0.97\times$ parameters separates the two effects cleanly: exact-program retention rises from 0.85 to $\mathbf{1.000}$ (the shared decoder ties the bespoke one exactly) while recalled-mass retention *falls* slightly, from 0.82 to 0.79. The MAP gap was capacity; the coverage gap is not.

Splicing: set-valued programs need a canonical order. A splicing program is a *set*, so the shared decoder must impose a canonical action order or the same program would be reachable by $|P|!$ sequences and its log-probability would not be a probability. The order matters: ordering by evidence rather than transcript index lifts recalled-mass retention from 0.476 to 0.603. The remaining gap is structural: the bespoke proposer reaches a program by several action sequences and so gets several chances within K draws, where the canonical order admits one. The shared decoder pays for being a proper distribution over programs, which the deduplicate-and-exact-score framework never rewards, since weighting is by exact p. The pattern follows from program geometry: fixed-candidate *ranking* unifies at no cost, rollout-generated support pays a coverage cost, and set-shaped programs pay the most.