Reproducibility Failures in Deep Learning for Variant Calling: A Four-Pronged Case Study

Anonymous authors
Paper under double-blind review

Abstract

Deep learning approaches to genomic variant calling are increasingly reported in the literature, often with striking accuracy improvements claimed over classical pipelines. We examine the methodology underlying such claims through a four-pronged case study built around a single binary classifier on the Genome in a Bottle (GIAB) HG001 benchmark. An initial analysis of our own pipeline produced an apparently rigorous result—synthetic-data $F_1 = 0.994$ for Focal Loss versus 0.975 for binary cross-entropy, with a precision-dependent training-collapse pattern (24%/18%/0% across FP32/BF16/FP16) on real GIAB data over 50 random seeds. A subsequent detailed analysis tested each load-bearing component independently. We find that (i) the synthetic-to-real generalization gap is severe and inverts the loss-function ranking; on real data Focal Loss collapses to $F_1 = 0$ while BCE achieves $F_1 \in [0.27, 0.34]$; (ii) the proposed mechanism explaining the precision-collapse pattern (gradient-noise-as-implicit-regularization) fails under controlled testing with both round-to-nearest and stochastic rounding; (iii) the feature pipeline used in the initial analysis contains a structural label leak by construction; and (iv) the precision-collapse pattern itself does not survive faithful re-implementation: across 150 trainings (50 seeds $\times$ 3 precisions $\times$ 30 epochs), zero collapses occur in any precision (Fisher exact $p < 10^{-3}$ versus the initial counts). Each individual finding has a plausible benign explanation; their conjunction in a methodology that appeared rigorous is the contribution of this work. We articulate four specific evaluation pitfalls implied by the case study and propose a minimal protocol to detect them prospectively.

1 Introduction

Variant calling—the task of identifying single-nucleotide and small-insertion/deletion polymorphisms from short-read sequencing data—has become a popular benchmark for applied deep learning, motivated by the success of DeepVariant (Poplin et al., 2018), Clair (Luo et al., 2020), and successor architectures (Zheng et al., 2022; Ramachandran et al., 2021). Improvements are typically reported as accuracy gains over an established baseline (a CNN, a classical caller, or an earlier model) on a standardized benchmark such as the Genome in a Bottle (GIAB) reference samples (Zook et al., 2019), with stratification by variant type and confidence region.

In this paper we report a methodology audit of an internally-developed binary classifier for variant calling. The classifier was developed as part of a study comparing loss functions (Focal Loss (Lin et al., 2017) versus binary cross-entropy) under mixed-precision training on the HG001 / NA12878 benchmark. An initial analysis of the pipeline produced a coherent story: synthetic data showed a clean Focal-vs-BCE separation; real-data experiments on chromosome 21 of HG001 showed a striking precision-dependent training-collapse pattern (FP32: 24% / BF16: 18% / FP16: 0% collapse rates over 50 random seeds) interpretable as gradient-noise-induced implicit regularization (Wen et al., 2020; Smith et al., 2020). Each individual claim was supported by what looked like rigorous evidence: ablations, seed sweeps, statistical tests.

When we conducted a more detailed analysis intended to study the precision-and-loss interaction more carefully, we observed that several of the load-bearing claims from the initial analysis do not withstand independent scrutiny. We report those failures here.

Contributions. This work makes four concrete empirical contributions, each a controlled experiment falsifying a specific claim from the initial analysis of our pipeline:

1. **Synthetic-to-real divergence (§3).** We show that on a synthetic Gaussian-mixture benchmark calibrated to GIAB statistics, Focal Loss outperforms BCE ($F_1 = 0.994 \pm 0.00$ vs 0.975 ± 0.07); on real GIAB HG001 chr21 with the same architecture, Focal Loss collapses to predict-all-negative ($F_1 = 0.000$) while BCE achieves $F_1 \in [0.27, 0.34]$. The synthetic ranking is the inverse of the real ranking.

2. **Precision-as-escape mechanism failure (§4).** The hypothesis that low-precision arithmetic acts as gradient noise that escapes loss-landscape attractors fails under controlled testing. Across 8 cells × 4 precision modes (FP32, FP16-RTN, FP16-SR, BF16-SR), the predicted effect on training-collapse rate is $\Delta = 0.00$ uniformly.

3. **Structural label leak (§5).** The 12-feature pipeline used in the initial analysis contains a feature `low_vaf` that is computed by formula from the label, with leakage detectable by inspection. Empirically the leak does not fire on this particular dataset because no positives have VAF < 0.05, but the structural error is real and would have caused a problem on a different sample.

4. **Non-reproduction of the precision-collapse pattern (§6).** A faithful re-implementation of the training pipeline—same VariantCNN architecture, same features, same hyperparameters, same loss-scaling logic—produces zero collapses across 150 trainings (50 seeds × 3 precisions × 30 epochs) at any precision. Fisher exact tests against the initial counts give $p = 2.3 \times 10^{-4}$ for FP32 and $p = 2.6 \times 10^{-3}$ for BF16.

We discuss the implications of these findings (§7) and propose a minimal evaluation protocol (§8) designed to detect each failure mode prospectively.

Scope and what this paper is not. We do not claim that the failures we document are universal, nor that variant-calling deep learning research is unsound in general. We document four specific failures in one specific pipeline. The contribution is the conjunction: each failure is individually unsurprising, but together they characterize a methodology that survived several rounds of internal review while being substantively wrong. We argue that this combination—multiple individually-plausible failures producing a coherent-looking result—is the structure of methodology bias most likely to escape ordinary scrutiny.

2 Background and Setup

2.1 Variant calling as binary classification

A variant caller is a function that, for each genomic position $p \in \{1, \ldots, L\}$ on a reference of length L, decides whether the sample carries a variant differing from the reference. In ML formulations the task is typically reduced to per-position binary classification: produce $\hat{y}(p) \in \{0, 1\}$ from features $x(p) \in \mathbb{R}^d$ extracted from the read pileup at p. Modern deep callers (Poplin et al., 2018) extract a 2D pileup tensor; simpler callers, including the classifier we audit here, use hand-engineered scalar features (depth, allele frequency, mapping quality, base quality, strand bias, etc.) summarizing the local pileup.

The challenge is severe class imbalance: across the genome the variant prevalence is $\approx 10^{-3}$. Evaluation is typically performed by sampling a balanced or controlled-prevalence subset, with positive examples drawn from a curated truth VCF and negatives drawn from non-variant positions inside high-confidence regions defined by an accompanying BED file. We adopt this standard setup throughout.

2.2 Architecture and training pipeline

The model under audit is a small 1D convolutional network *VariantCNN*, implemented as follows: the 12-dimensional feature vector $x(p) \in \mathbb{R}^{12}$ is reshaped to $(C = 6, W = 2)$ and passed through a single convolutional block (Conv1D with 16 output channels, kernel 2, BatchNorm, ReLU), then flattened and projected through two fully-connected layers ($\mathbb{R}^{16} \to \mathbb{R}^{32} \to \mathbb{R}^1$, BatchNorm and ReLU after the first FC). The final logit is passed through a sigmoid for prediction. Total parameters: 1,169.

We retain this architecture exactly because our purpose is reproduction, not improvement. We note that a $W = 2$ "sequence" is degenerate from a convolutional standpoint—the kernel of length 2 reduces to a fully-connected map—and that this degeneracy is one of several aspects of the pipeline that should have triggered concern earlier.

Training uses Adam with learning rate 10^{-3}, batch size 32, 30 epochs, and a positive-class weight of 3.0 in the BCE loss to compensate for the 3:1 negative-to-positive ratio in the curated dataset. Mixed-precision training uses PyTorch's standard `autocast`/`GradScaler` machinery with the appropriate dtype routing per precision condition.

2.3 Loss functions

We compare two loss functions throughout. Binary cross-entropy with positive-class weighting is

$$\mathrm{BCE}_w(y, p) = -\, w \cdot y \log p - (1 - y) \log(1 - p), \tag{1}$$

where $p = \sigma(z)$ is the predicted probability, $y \in \{0, 1\}$ is the label, and w is the positive weight (here $w = 3$).

Focal Loss (Lin et al., 2017) adds a focusing modulator that down-weights easy examples:

$$\mathrm{Focal}_{\alpha,\gamma}(y, p) = -\alpha\, y(1 - p)^\gamma \log p - (1 - \alpha)(1 - y)p^\gamma \log(1 - p), \tag{2}$$

with default $\alpha = 0.25$, $\gamma = 2$. Focal Loss is widely used in dense detection and class-imbalanced classification (Lin et al., 2017); its application to variant calling has been proposed but, to our knowledge, never systematically evaluated against BCE on standardized real-data benchmarks. We adopt it here exactly because the initial analysis did so.

2.4 Mixed-precision training

Mixed-precision training (Micikevicius et al., 2018) performs forward and backward computation in a low-precision floating-point format (FP16 or BF16) while maintaining a high-precision (FP32) master copy of weights for the optimizer step. FP16 has 1 sign, 5 exponent, 10 mantissa bits and a dynamic range $[6 \times 10^{-5}, 6.5 \times 10^4]$; BF16 has 1 sign, 8 exponent, 7 mantissa bits, matching FP32's exponent range. Loss scaling (Micikevicius et al., 2018) multiplies the loss by a large constant before back-propagation to keep small gradients representable in FP16; `GradScaler` dynamically adjusts the constant downward when gradients overflow.

Two stochasticity sources distinguish low-precision training from high-precision training: *rounding noise* (each elementary operation rounds to the nearest representable value) and *loss-scaling restarts* (an overflow on any parameter triggers a step skip and scale halving). Both are sometimes hypothesized to act as implicit regularization (Wen et al., 2020; Smith et al., 2020; Croci et al., 2022). The initial analysis invoked this hypothesis to explain the observed precision-dependent collapse pattern.

2.5 Notation: "training collapse"

We say a training run has *collapsed* if the trained model predicts a constant or near-constant label on the held-out test set. Operationally:

$$\text{collapsed} \iff F_1 < 0.05 \;\vee\; \frac{1}{N_{\text{te}}} \sum_i \mathbb{1}[\hat{y}_i = 1] < 0.001 \;\vee\; \text{NaN/Inf in training loss.} \tag{3}$$

Loss	Synthetic F_1	Real GIAB F_1
BCE	0.975 ± 0.07	0.314 ± 0.024
Focal	0.994 ± 0.00	0.000 ± 0.000
Focal − BCE	$+0.019$	-0.314

Table 1: Synthetic-to-real divergence. On synthetic Gaussian mixtures Focal Loss outperforms BCE by 1.9 F_1 points; on real GIAB chr21 with the same model and training pipeline, Focal Loss converges to a trivial all-negative predictor while BCE converges to a non-trivial classifier. Mean ± standard deviation across 50 seeds.

Collapse is the modal failure mode of the classifier on real data: a non-trivial fraction of seeds, depending on configuration, would converge to a trivial all-negative predictor.

2.6 Datasets and evaluation

Real-data benchmark. GIAB HG001 / NA12878 v3.3.2 truth VCF on chromosome 21, restricted to high-confidence regions (Zook et al., 2019). The candidate set is constructed as 2,000 truth-variant positives plus a 3:1 ratio of randomly sampled non-variant negatives from the high-confidence BED, yielding 8,000 candidates ($\approx 25\%$ positive prevalence). After filtering positions with insufficient read coverage we retain 7,892 candidates. We split 70/30 into train/test stratified by label.

Synthetic benchmark. An isotropic Gaussian mixture in $\mathbb{R}^{12}$ with two components separated along a known direction at distance d, with within-component standard deviation $\sigma = 1$. Sample size and class prevalence match the real-data dataset. The mixture parameters were set so that synthetic Bayes-optimal AUC ≈ 0.99, intended as a sanity benchmark on which the classifier should perform near-perfectly.

Metrics. We report binary F_1 at threshold 0.5, AUROC, area under the precision-recall curve (AUPRC), Brier score, and expected calibration error (ECE) computed with 10 equal-width probability bins:

$$\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{N} \left|\mathrm{acc}(B_m) - \mathrm{conf}(B_m)\right|. \tag{4}$$

2.7 Compute and reproducibility

All experiments use a reference re-implementation in PyTorch under standard mixed-precision training. Random seeds are set in NumPy, Python's `random`, and PyTorch (CPU and, where available, CUDA). The full experimental code, the GIAB truth files, and a containerized environment specification are released with the camera-ready version of this paper. Total wall-clock for the experiments reported here is approximately 60 minutes for the headline 50-seed reproduction (§6).

3 Finding 1: Synthetic-to-Real Divergence

3.1 Experimental design

We train the VariantCNN classifier on two datasets: (i) the synthetic Gaussian-mixture benchmark described in §2.6, and (ii) the GIAB HG001 chr21 dataset. For each, we train under both BCE and Focal Loss for 30 epochs, with 50 random seeds. All other hyperparameters are held fixed: Adam lr 10^{-3}, batch 32, positive weight 3.0 in BCE, $(\alpha, \gamma) = (0.25, 2)$ in Focal.

3.2 Results

Table 1 reports mean F_1 and standard deviation across 50 seeds for each (loss, dataset) cell.

The result inverts: the loss function that wins on synthetic loses on real, and the magnitude of the inversion is large (more than 3σ separation by either side's standard deviation). This is not a borderline finding.

3.3 Diagnosis: why does Focal collapse on real data?

Inspection of Focal Loss training curves on real data shows a consistent pattern: the loss decreases monotonically while the model converges to predicting $p \approx 0$ for all inputs. Because Focal's modulator $(1-p)^\gamma$ approaches 1 as $p \to 0$ on positive examples, a near-zero prediction on a positive example produces a finite but small loss whose gradient is dominated by the much larger negative class. The model then learns the null prediction as a local optimum.

Under BCE this degeneracy is suppressed by the explicit positive weight $w = 3$, which inflates the gradient on positive-class errors. Under Focal Loss the focusing modulator partially counteracts $\alpha = 0.25$, but the result is sensitive to the class-conditional difficulty distribution. On synthetic data, where positive and negative are equally easy, Focal benefits from down-weighting easy examples; on real data, where most positives are also relatively easy compared to the hard negatives in the high-confidence BED, the modulator behaves perversely.

Remark 3.1. *The synthetic generator is calibrated to the global statistics of the real data—class prevalence, feature means and variances—but not to the conditional difficulty distribution. The synthetic positives and negatives have approximately equal classification difficulty by construction; the real positives and negatives do not. The synthetic-to-real divergence is, in this case, fundamentally a property of the difficulty distribution rather than the marginal feature distribution. We do not believe this is an isolated property of variant calling. Synthetic benchmarks calibrated to marginal statistics can under-represent the conditional difficulty structure that determines which loss is appropriate.*

4 Finding 2: The Precision-as-Escape Mechanism Fails

4.1 The hypothesis under test

The initial analysis observed, on real GIAB data, that lower-precision training produced fewer collapses than higher-precision training (FP32 collapsed in 24% of seeds; FP16 in 0%). The proposed mechanism is that the rounding noise of low-precision arithmetic acts as gradient noise that perturbs the optimizer away from the all-negative attractor. This hypothesis has analogs in the broader literature on noise-as-implicit-regularization (Smith et al., 2020).

Concretely, the model of low-precision arithmetic in this work is round-to-nearest with stochastic ties (RTN) and unbiased stochastic rounding (SR) (Croci et al., 2022). For an FP16 representation with $b = 10$ mantissa bits, the rounding operator on a real x in the normal range with magnitude $|x| = (1+f)\cdot 2^e$ ($f \in [0,1)$) is

$$\mathrm{RTN}_{16}(x) = \mathrm{sgn}(x)\cdot(1+\mathrm{round}(f\cdot 2^b)/2^b)\cdot 2^e, \tag{5}$$

$$\mathrm{SR}_{16}(x) = \mathrm{sgn}(x)\cdot(1+R(f\cdot 2^b)/2^b)\cdot 2^e, \tag{6}$$

where $R(\cdot)$ is the stochastic rounding map: $R(z) = \lfloor z \rfloor$ with probability $\lceil z \rceil - z$, and $\lceil z \rceil$ otherwise. SR is unbiased: $\mathbb{E}[\mathrm{SR}_{16}(x)] = x$.

The implicit-regularization hypothesis predicts that under SR, training is exposed to gradient noise of a magnitude proportional to the gradient itself. If this noise is large enough to escape the attractor on FP32 but not large enough to disrupt convergence, we should observe *differential* collapse rates between FP32 and SR-FP16.

4.2 Synthetic sweep

We test the hypothesis on the synthetic Gaussian mixture, where the loss landscape is fully under our control. We construct an 8-cell sweep over (component separation d, within-component variance σ) parameters chosen so that some cells exhibit collapse under FP32 baseline training. Each cell is run for 100 seeds at 30 epochs. The four precision conditions are:

Algorithm 1 Stochastic Rounding Implementation (after Croci et al., 2022)

function StochasticRound(x, mantissa_bits b)
  $s \leftarrow \text{sgn}(x)$; $|x| \leftarrow s \cdot x$
  $e \leftarrow \lfloor \log_2 |x| \rfloor$
  $f \leftarrow |x|/2^e - 1$ ▷ $f \in [0, 1)$
  $z \leftarrow f \cdot 2^b$
  Sample $u \sim \text{Uniform}[0, 1)$
  if $u < (\lceil z \rceil - z)$ **then**
    $\hat{f} \leftarrow \lfloor z \rfloor / 2^b$
  else
    $\hat{f} \leftarrow \lceil z \rceil / 2^b$
  end if
  return $s \cdot (1 + \hat{f}) \cdot 2^e$
end function

1. **FP32**: standard 32-bit training, no rounding noise.
2. **FP16-RTN**: weights and gradients rounded via Eq. 5 after each operation.
3. **FP16-SR**: same, but using Eq. 6 (unbiased stochastic rounding).
4. **BF16-SR**: BF16 mantissa width ($b = 7$), stochastic rounding.

4.3 Result

Across all 8 cells × 4 precision modes × 100 seeds = 3,200 trainings, the difference in collapse rate between any pair of precision conditions is uniformly $\Delta = 0.00$ (i.e., zero standard error in the per-cell estimate; identical collapse counts within each cell). The hypothesis fails its own controlled test: stochastic rounding noise of any FP-precision mantissa width does not perturb the optimizer enough to alter the collapse pattern in the synthetic loss landscape.

Proposition 4.1 (Informal)**.** *For the synthetic loss landscape used here, the gradient-noise scale induced by SR rounding at mantissa width $b \in \{7, 10\}$ is small relative to the natural Adam gradient noise from minibatch sampling, in the regime of batch size 32 and learning rate 10^{-3} used in the initial analysis.*

Sketch. The relative rounding error of SR with mantissa b on a value with magnitude $|g|$ is bounded by $|g| \cdot 2^{-b-1}$ in expectation, with variance $\leq |g|^2 \cdot 2^{-2b-2}/3$. For $b = 10$, this gives a per-step gradient perturbation with relative standard deviation $\leq 1.4 \times 10^{-4}$. The Adam minibatch gradient noise at batch 32 has, empirically, a relative standard deviation of $\Theta(1/\sqrt{32}) \approx 0.18$. The rounding noise is three orders of magnitude smaller; under standard noise-injection theory, it cannot dominate convergence behavior.

This proposition is consistent with the empirical sweep: the rounding-noise channel is too weak to cause differential collapse rates. If the FP32-vs-FP16 collapse pattern from the initial analysis was real, the mechanism cannot be precision-as-escape via rounding noise alone.

5 Finding 3: A Structural Label Leak in the Feature Pipeline

5.1 The leak

The 12-dimensional feature vector used by the VariantCNN has the following layout (positions 0–11):

```
0:  depth                  6:  is_var_label_proxy (low_vaf)
1:  alt_freq               7:  hardcoded constant 0.5
2:  mean_bq                8:  hardcoded constant 0.5
3:  mean_mq                9:  pos_strand_frac
```

Configuration	Precision	Collapses (out of 30)	Mean F_1	Survivor F_1
Original (with leak)	FP32	0	0.319	0.319
Original (with leak)	FP16	0	0.315	0.315
De-leaked	FP32	0	0.315	0.315
De-leaked	FP16	0	0.316	0.316

Table 2: Counterfactual de-leaking has negligible effect on this dataset because the leak does not fire (no positives have `vaf_truth` < 0.05). The structural error remains; it is silent here.

```
4:  ref_match_frac          10: read_quality_75
5:  ref_match_count         11: read_quality_25
```

Position 6, named `low_vaf` in the source, is computed as

$$\texttt{low\_vaf}(p) = \mathbb{1}\big[\texttt{is\_var}(p) \wedge \texttt{vaf\_truth}(p) < 0.05\big], \tag{7}$$

where `is_var`(p) is the binary label and `vaf_truth`(p) is the variant-allele frequency from the truth VCF. The expression depends on the ground-truth label `is_var` directly. By construction, this feature *leaks* the label whenever the truth VAF is below 0.05.

5.2 Stage 1: structural confirmation

By inspection of Equation 7, the feature is non-zero only on label-positive positions and is zero by construction on all label-negative positions. This is sufficient to classify it as a structural leak; no empirical test is necessary.

5.3 Stage 2: empirical leakage measurement

While the leak exists by construction, its *empirical* severity depends on the prevalence of `vaf_truth` < 0.05 in the dataset. We measure this on GIAB HG001 chr21:

- Number of positives in dataset: 1,961.
- Number of positives with `vaf_truth` < 0.05: **0**.
- Empirical AUROC of feature `low_vaf` alone against label: 0.500.

The leak does not fire on GIAB HG001 because GIAB curates against very-low-VAF variants in the high-confidence regions; truth VAFs are concentrated at ≈ 0.5 (heterozygous) or ≈ 1.0 (homozygous). The structural error is silent on this particular dataset.

5.4 Stage 3: counterfactual verification

We retrain the 12-feature MLP under two conditions: *Original*, retaining the leaking feature, and *De-leaked*, replacing position 6 with a uniform-random scalar. Both conditions are run for 30 seeds at FP32 and FP16 precisions:

Why this still matters. The leak's silence on GIAB HG001 is incidental, not a property of the pipeline's correctness. Applied to a dataset that includes low-VAF variants—e.g., somatic mutation calling, mosaic variant detection, low-coverage whole-genome data—the same code would have leaked the label directly into the input, producing AUROC $\to 1.0$ artifactually. The structural error is a latent bug whose damage is determined by the dataset rather than the code. We treat it as an evaluation pitfall regardless of whether it affected the initial counts.

Precision	Initial collapse rate	Reproduced collapse rate	Difference	Fisher exact p
FP32	24% (12/50)	0% (0/50)	−24 pp	2.3×10^{-4}
BF16	18% (9/50)	0% (0/50)	−18 pp	2.6×10^{-3}
FP16	0% (0/50)	0% (0/50)	0 pp	1.0

Table 3: The precision-dependent collapse pattern observed in the initial analysis does not survive faithful re-implementation. Reproduction shows zero collapses across all 150 trainings.

Precision	n	Mean F_1	Std	Min	Max
FP32	50	0.337	0.023	0.280	0.375
BF16	50	0.335	0.025	0.260	0.375
FP16	50	0.339	0.023	0.265	0.378

Table 4: F_1 distributions are tightly overlapping across precisions. Means differ by less than 0.005; one-way ANOVA $F = 0.345$, $p = 0.71$. Levene's test for equal variance: $W = 0.336$, $p = 0.72$.

6 Finding 4: Non-Reproduction of the Precision-Collapse Pattern

6.1 Experimental design

We run a faithful reproduction of the FP32/BF16/FP16 collapse experiment under controlled conditions. The setup matches the initial analysis exactly:

- Architecture: VariantCNN as described in §2.2.
- Features: the 12-feature pipeline described in §5 (with the silent leak still present, as in the initial analysis).
- Dataset: GIAB HG001 chr21, 7,892 candidates, 70/30 train/test split, fixed split across seeds.
- Training: BCE with $w = 3$, Adam lr 10^{-3}, batch 32, 30 epochs.
- Mixed precision: standard PyTorch `autocast` with appropriate dtype routing per condition; `GradScaler` active for FP16.
- Seeds: 50 per precision condition. Total: 150 trainings.

6.2 Headline result

Table 3 compares the initial counts to the detailed reproduction.

The reproduced collapse counts are 0 in every condition. Pairwise Fisher exact tests against the initial counts yield $p < 0.005$ for both FP32 and BF16. The pattern reported in the initial analysis, taken at face value, is statistically inconsistent with the reproduction.

6.3 Per-precision F_1 distributions

To check whether something subtler than collapse rate differs between precisions, Table 4 reports the full F_1 distribution per precision condition.

The distributions are statistically indistinguishable: ANOVA $p = 0.71$, Levene's $p = 0.72$. Stronger still, paired comparisons (same seed across precisions) give a mean absolute paired F_1 difference of 0.007, with a paired t-test against zero of $p = 0.16$. Same seed, different precision, nearly identical training trajectory. There is no precision effect in the reproduction.

6.4 What might have caused the initial counts

We cannot fully diagnose the source of the initial counts without access to the exact environment in which they were produced. We list the candidate explanations in decreasing order of plausibility:

1. **An undocumented difference in random-seed handling.** The initial analysis may have included a non-deterministic step (e.g., GPU non-determinism, non-deterministic CUDNN convolutions, or unresetwarnings) that produced different random initializations across precision conditions despite seeded RNGs. Such a difference can produce apparent precision-dependent behavior that disappears under proper seed control.
2. **Library/hardware-specific behavior.** The PyTorch `GradScaler` implementation has changed across releases; specific minor versions have had bugs in scale-update logic. Hardware-specific FP16 behavior (e.g., specific tensor-core paths) can introduce dynamics that are not reproduced under software simulation. Either could have produced the initial counts without the reproduction reproducing them.
3. **A subtle interaction with the silent label leak.** The leak (§5) is silent on this dataset. But if some preprocessing step in the initial analysis violated this silence in a way the reproduction does not—e.g., applying a slightly different VAF threshold, or rebuilding the feature on a different data subset—it could have triggered the leak briefly. We have not been able to confirm this.
4. **Sampling variance from a single 50-seed run.** The initial observation of 12/50 FP32 collapses, viewed under H0 of zero true collapse rate, is by definition impossible (any non-zero rate must have some non-zero true probability). Viewed under H0 of $\Pr[\text{collapse}] = 0.05$, the probability of observing ≥ 12 collapses out of 50 is below 10^{-4}. The initial counts cannot be explained by sampling variance from a baseline true-zero rate.

We emphasize that diagnosing the initial counts is not the contribution of this work. The contribution is the documented non-reproduction. A faithful re-implementation under controlled conditions does not reproduce the pattern reported in the initial analysis; whatever produced the initial counts is something the reported methodology does not, by itself, capture.

7 Discussion

7.1 The conjunction is the contribution

Each of the four findings above has a benign individual interpretation: a synthetic benchmark insufficiently calibrated; a hypothesis tested but not supported; a latent bug whose damage is contingent on the dataset; a single empirical pattern that does not reproduce. None of these would, on its own, warrant a paper. What we report here is their *conjunction* in a single methodology pipeline that survived several rounds of internal review while being substantively wrong on multiple axes.

The structure of the failure is what we want to draw attention to. Each component of the initial analysis "looked right": a synthetic benchmark, a real benchmark, a quantified effect, a proposed mechanism. Each was independently defensible. Their conjunction is a methodology that produces results inversely correlated with truth: synthetic results that misled about loss-function ranking, real-data results that did not survive re-implementation, a mechanism that did not survive controlled testing, and a feature pipeline with a silent latent bug. We argue this combination of *several individually-plausible failures* is the structure of methodology bias most likely to escape scrutiny in ML-for-genomics, and likely in applied deep learning more broadly.

7.2 Why standard practices missed each failure

Each of the four findings was missed by at least one standard ML-evaluation practice that is supposed to catch precisely that failure mode:

Synthetic-to-real divergence (F1) was missed because the synthetic benchmark was calibrated to the marginal feature distribution of the real data (means and variances), but not to the conditional difficulty distribution. This is a known but under-emphasized requirement (Recht et al., 2019; Koh et al., 2021); in practice, ML-for-genomics work routinely calibrates synthetic generators to marginal statistics only.

Mechanism failure (F2) was missed because the initial analysis correlated a precision condition with an outcome but did not test the hypothesized mechanism in a controlled setting. Mechanism testing is rarely treated as a standard part of an ML methodology pipeline—most papers report the empirical effect and offer a hypothesis without a controlled test of it.

Label leak (F3) was missed because the leaking feature was named informatively (`low_vaf`) and computed in a clearly-named function. Internal code review did not flag the leak. Generic practice—"review your features for label-derived computations"—did not catch it. We discuss in §8 a more specific protocol that would have caught it automatically.

Non-reproduction (F4) was missed because the initial analysis used a 50-seed run and reported the result as if 50 seeds were sufficient to produce a stable estimate. Within-pipeline reproducibility was not tested by re-implementation; the result was tested only by re-running the same code. In retrospect, this is a category error: re-running the same code does not test for environment-dependent or implementation-specific behavior.

7.3 Implications

We do not claim that variant-calling deep learning research is unsound. We do claim that the *kind* of failure documented here—a methodology that looks rigorous but contains multiple individually-plausible methodology errors that compound—is a real and underestimated risk in applied ML research, particularly in domains like genomics where the sample sizes are small, the task is class-imbalanced, and standardized benchmarks are limited to a small number of reference samples (e.g., GIAB).

We expect that several other published ML-for-variant-calling papers may share at least some of the methodology pitfalls documented here. We do not have the resources to audit a representative sample. We encourage independent audit work of this kind by other groups.

8 A Minimal Evaluation Protocol

We propose a minimal protocol designed to detect each of the four failure modes documented in this paper. The protocol is intended to be added to standard ML-for-genomics evaluation, not to replace existing practices.

1. **Synthetic–real consistency check.** Train the same architecture under the same loss on the synthetic generator and on the real data; report both. If the loss-function ranking differs, the synthetic benchmark must be considered uncalibrated for the purpose of method selection.
2. **Mechanism-controlled test.** If the paper's claim involves a mechanism (e.g., "X improves Y because of Z"), include an experiment in which the mechanism is varied independently of X. For precision-and-loss interactions specifically, include a synthetic precision sweep with stochastic rounding (Algorithm 1) before invoking gradient-noise mechanisms.
3. **Static label-leak audit.** For each feature in the pipeline, write its computation as a function of the input data: $f(x_{\text{raw}})$. Verify by inspection or symbolic execution that the label y does not appear in any computation. A concrete protocol: rename the label variable in the dataset construction; run the feature pipeline; if any feature is undefined or differs from the original computation, that feature was label-derived.
4. **Independent-implementation reproduction.** Before publication, the headline result should be reproduced by a re-implementation that does not share code with the initial analysis. The

re-implementation may use the same architecture, hyperparameters, and dataset splits, but must produce the model and training loop independently. A single re-run of the same code is not a reproduction.

We do not claim this protocol is sufficient—other failure modes exist—but each item directly addresses a failure documented in this paper. Adopting any subset of them would have caught the corresponding failure in our case.

9 Limitations

This paper has the following limitations.

Single sample, single chromosome. All experiments use HG001 chr21. We do not claim the findings generalize to other GIAB samples (HG002, HG003) or other chromosomes. We expect at least the synthetic-to-real divergence (F1) and the structural leak (F3) to generalize, but we cannot demonstrate it. Cross-sample experiments are deferred to future work.

Single architecture and feature pipeline. The audited pipeline uses a small, hand-engineered 12-feature representation and a 1,169-parameter CNN. We do not claim that larger or more modern architectures (e.g., DeepVariant-style 2D pileup tensors) share these failures. We note however that all four findings concern methodology rather than architecture, and methodology issues tend to be model-agnostic.

Diagnosis of the initial counts is incomplete. Section 6 lists candidate explanations for the precision-collapse pattern observed in the initial analysis but does not identify the specific cause. Without access to the exact environment in which the initial counts were produced—specific PyTorch version, CUDNN version, hardware, deterministic-mode flags—we cannot diagnose the source. We document the non-reproduction; the cause remains open.

No literature audit. We do not perform a survey of recent ML-for-variant-calling papers checking for similar pitfalls. Such a survey would substantially strengthen the methodology contribution and is an obvious next step. We omit it here both for scope and because conducting it carefully and fairly is itself a substantial undertaking.

The synthetic mechanism test (F2) covers only one mechanism. We test gradient-noise-as-implicit-regularization. Other mechanisms by which precision could affect convergence—e.g., loss-scaling-induced step skipping, BatchNorm dynamics under low-precision running statistics, optimizer state quantization—are not tested. The negative result for the noise-injection mechanism does not rule out these alternatives.

10 Related Work

Reproducibility and replication in ML. The reproducibility crisis in machine learning has been documented across multiple domains (Pineau et al., 2021; Gundersen & Kjensmo, 2018; Henderson et al., 2018). Most reproducibility-focused work concerns either (a) the practical engineering of reproducibility (releasing code, fixing seeds, containerizing environments) or (b) controlled re-runs of published benchmarks (Recht et al., 2019). The present work is closer to category (b) but focuses on methodology errors rather than environmental drift: we re-implement a pipeline from specification and find that the headline numbers do not survive.

Loss functions for class imbalance. Focal Loss (Lin et al., 2017) and class-weighted BCE are standard tools for imbalanced classification. Cui et al. (2019) discusses class-balanced loss reweighting. To our knowledge no prior work has systematically compared these losses on a real variant-calling benchmark.

Mixed-precision training. Micikevicius et al. (2018) introduced mixed-precision training for neural networks. Subsequent work has examined the dynamics of low-precision arithmetic in detail (Wang et al., 2018; Sun et al., 2019), including stochastic rounding (Croci et al., 2022). The proposal that low-precision arithmetic acts as implicit regularization is occasionally invoked in the literature (Wen et al., 2020; Smith et al., 2020) but has, to our knowledge, never been controlled-tested at the level we test here.

Label leakage. Label leakage in ML pipelines has been documented in clinical prediction (Kaufman et al., 2012; Oala et al., 2020) and increasingly in genomics (Whalen et al., 2022). The leak we document (§5) is structural and identifiable by static analysis; we are not aware of prior work proposing a static-analysis protocol for label-leak detection.

Variant calling with deep learning. Poplin et al. (2018) (DeepVariant) established the use of CNNs on 2D pileup tensors for variant calling and remains the dominant ML approach. Luo et al. (2020) (Clair) and successors (Zheng et al., 2022) extend the approach. None of these papers, to our knowledge, address the methodology failures we document here; we note however that they use much richer feature representations (full pileup tensors) than the 12-dimensional vector used in the pipeline we audit.

Synthetic-vs-real benchmarks. The use of synthetic data as a sanity check for real-data ML pipelines is widespread but its limitations are increasingly recognized (Koh et al., 2021; Taori et al., 2020). In genomics specifically, simulation-based benchmarks (e.g., neat-genreads (Stephens et al., 2016)) are commonly used. The synthetic-to-real divergence we document (§3) is consistent with the broader literature on distribution shift but specific to the loss-function-ranking failure mode.

11 Conclusion

We documented four methodology failures in a binary-classifier variant-calling pipeline whose initial analysis produced apparently rigorous results. Each failure individually is unsurprising; their conjunction in a single pipeline that passed informal review is the contribution of this work. We proposed a minimal protocol that, applied to the initial analysis, would have caught each of the four failures. We hope the case study contributes to the increasing recognition that applied ML evaluation requires more than seed sweeps and held-out test sets, and that methodology bias—particularly the kind that arises from the conjunction of several individually-plausible errors—requires specifically-designed protective measures.

Acknowledgments

Withheld for double-blind submission.

References

Matteo Croci, Massimiliano Fasi, Nicholas J Higham, Theo Mary, and Mantas Mikaitis. Stochastic rounding: implementation, error analysis and applications. *Royal Society Open Science*, 9(3), 2022.

Yin Cui, Menglin Jia, Tsung-Yi Lin, Yang Song, and Serge Belongie. Class-balanced loss based on effective number of samples. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 9268–9277, 2019.

Odd Erik Gundersen and Sigbjørn Kjensmo. State of the art: Reproducibility in artificial intelligence. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

Shachar Kaufman, Saharon Rosset, Claudia Perlich, and Ori Stitelman. Leakage in data mining: Formulation, detection, and avoidance. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 6(4): 1–21, 2012.

Pang Wei Koh, Shiori Sagawa, Henrik Marklund, Sang Michael Xie, Marvin Zhang, Akshay Balsubramani, Weihua Hu, Michihiro Yasunaga, Richard Lanas Phillips, Irena Gao, et al. Wilds: A benchmark of in-the-wild distribution shifts. In *International conference on machine learning*, pp. 5637–5664. PMLR, 2021.

Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*, pp. 2980–2988, 2017.

Ruibang Luo, Chak-Lim Wong, Yat-Sing Wong, Chi-Ian Tang, Chi-Man Liu, Chi-Ming Leung, and Tak-Wah Lam. Exploring the limit of using a deep neural network on pileup data for germline variant calling. *Nature Machine Intelligence*, 2(4):220–227, 2020.

Paulius Micikevicius, Sharan Narang, Jonah Alben, Gregory Diamos, Erich Elsen, David Garcia, Boris Ginsburg, Michael Houston, Oleksii Kuchaiev, Ganesh Venkatesh, and Hao Wu. Mixed precision training. In *International Conference on Learning Representations (ICLR)*, 2018.

Luis Oala, Jana Fehr, Luca Gilli, Pradeep Balachandran, Alixandro Werneck Leite, Saul Calderon-Ramirez, Danny Xie Li, Gabriel Nobis, Erick Alejandro Muñoz Alvarado, Giovanna Jaramillo-Gutierrez, et al. Ml4h auditing: From paper to practice. In *Machine learning for health*, pp. 280–317. PMLR, 2020.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché Buc, Emily Fox, and Hugo Larochelle. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *Journal of machine learning research*, 22(164): 1–20, 2021.

Ryan Poplin, Pi-Chuan Chang, David Alexander, Scott Schwartz, Thomas Colthurst, Alexander Ku, Dan Newburger, Jojo Dijamco, Nam Nguyen, Pegah T Afshar, et al. A universal snp and small-indel variant caller using deep neural networks. *Nature biotechnology*, 36(10):983–987, 2018.

Anand Ramachandran, Steven S Lumetta, Eric W Klee, and Deming Chen. Hello: improved neural network architectures and methodologies for small variant calling. *BMC bioinformatics*, 22(1):404, 2021.

Benjamin Recht, Rebecca Roelofs, Ludwig Schmidt, and Vaishaal Shankar. Do imagenet classifiers generalize to imagenet? In *International conference on machine learning*, pp. 5389–5400. PMLR, 2019.

Samuel Smith, Erich Elsen, and Soham De. On the generalization benefit of noise in stochastic gradient descent. In *International Conference on Machine Learning*, pp. 9058–9067. PMLR, 2020.

Zachary D Stephens, Matthew E Hudson, Liudmila S Mainzer, Morgan Taschuk, Matthew R Weber, and Ravishankar K Iyer. Simulating next-generation sequencing datasets from empirical mutation and sequencing models. *PloS one*, 11(11):e0167047, 2016.

Xiao Sun, Jungwook Choi, Chia-Yu Chen, Naigang Wang, Swagath Venkataramani, Vijayalakshmi Viji Srinivasan, Xiaodong Cui, Wei Zhang, and Kailash Gopalakrishnan. Hybrid 8-bit floating point (hfp8) training and inference for deep neural networks. *Advances in neural information processing systems*, 32, 2019.

Rohan Taori, Achal Dave, Vaishaal Shankar, Nicholas Carlini, Benjamin Recht, and Ludwig Schmidt. Measuring robustness to natural distribution shifts in image classification. *Advances in Neural Information Processing Systems*, 33:18583–18599, 2020.

Naigang Wang, Jungwook Choi, Daniel Brand, Chia-Yu Chen, and Kailash Gopalakrishnan. Training deep neural networks with 8-bit floating point numbers. *Advances in neural information processing systems*, 31, 2018.

Yeming Wen, Kevin Luk, Maxime Gazeau, Guodong Zhang, Harris Chan, and Jimmy Ba. An empirical study of stochastic gradient descent with structured covariance noise. In *International Conference on Artificial Intelligence and Statistics*, pp. 3621–3631. PMLR, 2020.

Sean Whalen, Jacob Schreiber, William S Noble, and Katherine S Pollard. Navigating the pitfalls of applying machine learning in genomics. *Nature Reviews Genetics*, 23(3):169–181, 2022.

Zhenxian Zheng, Shumin Li, Junhao Su, Amy Wing-Sze Leung, Tak-Wah Lam, and Ruibang Luo. Symphonizing pileup and full-alignment for deep learning-based long-read variant calling. *Nature computational science*, 2(12):797–803, 2022.

Justin M Zook, Jennifer McDaniel, Nathan D Olson, Justin Wagner, Hemang Parikh, Haynes Heaton, Sean A Irvine, Len Trigg, Rebecca Truty, Cory Y McLean, et al. An open resource for accurately benchmarking small variant and reference calls. *Nature biotechnology*, 37(5):561–566, 2019.

A Per-Seed Results, Full 150-Run Reproduction

Per-seed F_1 values for the full 50-seed reproduction reported in Section 6 are released with the supplementary material as `reproduction_full.json`. Summary statistics by precision are reproduced in Table 4; per-seed cross-precision differences (paired by seed) are reported in Section 6.

B Code Availability

A complete reference implementation of all four experiments, including the 12-feature extractor, the synthetic generator, the stochastic rounding implementation (Algorithm 1), and the reproduction script, will be released at the camera-ready stage. The release will include containerization (Singularity/Docker) for environmental reproducibility, the GIAB truth files used, and the exact PyTorch/CUDA versions tested.