

Logit Leakage through Floating-Point Bias Addition

Shaomin Chen

Xiaorui Gong

Abstract—Language model APIs can expose token-specific biases while withholding logits and token probabilities. Existing bias-based extraction recovers relative logits, leaving a common additive component undetermined in ideal arithmetic. We identify a numerical side channel in floating-point bias addition that exposes information about this component through token identities alone. The channel distinguishes score vectors with identical complete unperturbed log probabilities. Our method exploits changes in representable spacing at binary exponent boundaries. Under the stated binary32 arithmetic assumptions and a known reachable score interval, it recovers a logit pair and its unknown tie-breaking priority exactly in at most 52 token queries. Given complete relative logits, one recovered anchor determines the original logit vector. We evaluate recovery across 11 public checkpoints from seven model families and extend the local method with adaptive interval search and anchor selection. In end-to-end experiments with vLLM, SGLang, and llama.cpp, adaptive recovery succeeds on all 128 Qwen3-0.6B inputs per system, including 64 held-out inputs. A separate token-only experiment extracts a pretrained small model’s complete output-head representation up to an invertible hidden-basis transformation. We also establish a sufficient condition for suppressing common-shift disclosure by subtracting the maximum before bias addition. In controlled experiments, this mitigation eliminates all observed distinctions between common-shift variants and preserves token decisions under the tested random biases.

1. Introduction

Language-model services expose configurable decoding interfaces while withholding internal logits and model parameters. Logit bias lets users steer generated text by increasing or decreasing the likelihood of specified tokens, even when the interface returns only the selected token [1]. Understanding what numerical information can be recovered through these controls is part of assessing the information exposed by a prediction interface.

Prediction interfaces support model extraction [2], including recovery of language-model output projections and their dimensions [3], [4]. Morris et al. and Carlini et al. recover relative logits through token-specific biases and observed token choices [3], [5]. In exact real arithmetic, both softmax probabilities and biased-argmax responses are invariant to a common additive shift of the logit vector. These observations determine logit differences while leaving one additive component unidentified for each input.

We show that floating-point bias addition can expose this component through token identities alone. Floating-point spacing varies with magnitude, so adding a chosen bias can produce different rounded comparisons for logit vectors related by a common shift. We construct queries that distinguish such vectors even when their complete unperturbed log probabilities are identical. Under explicit arithmetic and range conditions, we use these differences to recover original-logit anchors. Combined with complete relative logits, an anchor determines the original logit vector, including its common additive component.

Figure 1 summarizes the mechanism and recovery. We address three technical challenges:

- **Identifying leakage beyond relative logits.** Common-shift score vectors produce identical ideal query responses. Establishing additional disclosure requires floating-point queries that distinguish these vectors while their complete unperturbed probability outputs remain identical.
- **Recovering original logits from token responses.** A token comparison combines score gaps, rounding, and an unknown tie-breaking order. Bounded biases restrict the accessible numerical regions, so recovery requires queries that separate these effects.
- **Suppressing leakage while preserving bias control.** Removing dependence on the common additive component can change rounded comparisons and token choices. A mitigation must retain token-specific bias control and be evaluated for both channel suppression and preservation of decoding behavior.

For positive-score recovery, we determine the logit gap and the token tie-breaking priority, then probe the original anchor using the spacing change at an exponent boundary. The probe responses are periodic over part of the candidate range. We first use a small number of queries to determine the anchor’s position within this repeating pattern, then restrict the candidate values accordingly to obtain a monotone comparison for binary search. Binary search then recovers the anchor exactly within the stated interval, and the recovered gap determines the second original logit.

Our central result assumes stable scores, independently converted biases, one correctly rounded addition per coordinate, greedy selection, and a fixed but unknown token tie-breaking order. Given a known reachable score interval, recovery takes at most 50 binary queries for a positive binary32 score pair and its priority, or 47 for negative scores. Selecting the two tokens takes two additional queries. We validate recovery across 11 public checkpoints from seven

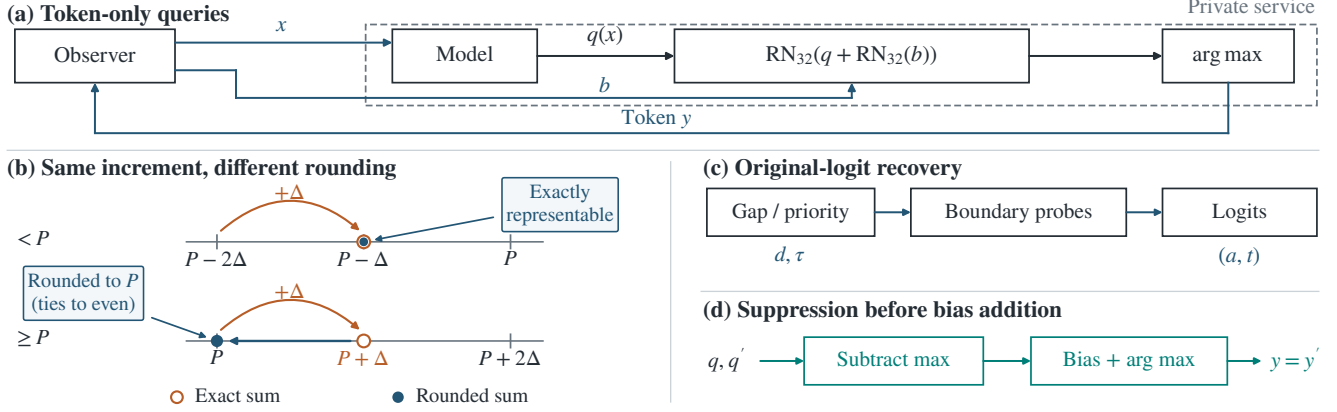


Figure 1. Overview of the numerical channel and recovery. (a) The observer chooses token biases b for fixed input x and observes the selected token; RN_{32} denotes binary32 rounding. (b) At the positive boundary $P = 2^e$, spacing increases from Δ to 2Δ . At the illustrated point below P , the increment is exact. At P , the exact sum $P + \Delta$ lies halfway between P and $P + 2\Delta$ and rounds to P under ties-to-even. (c) Gap and priority recovery followed by boundary probes recovers the score pair under the stated arithmetic and range assumptions. (d) Subtracting the maximum before bias suppresses common-shift disclosure when stored variants yield identical centered vectors.

model families and extend the local method with interval search and anchor selection. End-to-end adaptive recovery through vLLM, SGLang, and llama.cpp succeeds on all 128 Qwen3-0.6B inputs per system, including 64 held-out inputs.

The recovered component has a direct role in output-head extraction. For a bias-free linear head under the rank conditions in Appendix G, centered logits determine CWG , where C subtracts the vocabulary mean, W is the output head, and G is an invertible hidden-basis transformation. Original-logit anchors supply the missing shared row component and permit recovery of WG . We demonstrate this composition by extracting a pretrained model’s complete 512×64 output-head representation from token responses across three independent input sets. On vocabulary rows excluded from basis alignment, weight RMSE ranges from 3.60×10^{-7} to 1.20×10^{-6} (Section 5.3).

Finally, we analyze subtracting the maximum before bias addition as a mitigation. When common-shift variants yield identical preprocessed vectors, subsequent bias queries disclose no difference between them. Controlled experiments verify this condition and measure output preservation under independently generated sparse biases.

We make three contributions:

- 1) **Logit leakage through floating-point bias addition.** We characterize how rounding exposes a common additive component that remains unidentified by ideal probabilities and biased-argmax responses.
- 2) **Constructive recovery and output-head extraction.** We give local exact-recovery algorithms and query bounds, validate them through generation and serving interfaces, and combine recovered anchors with relative logits to extract an output-head representation.
- 3) **Processing-order analysis and mitigation.** We establish a sufficient condition for suppressing

common-shift disclosure through pre-bias max subtraction and measure its effect on token decisions.

2. Background and Problem Formulation

2.1. Logits, Probabilities, and Common Shifts

At one generation position, input x determines the vector $q(x) \in \mathbb{R}^V$ entering bias addition, where V is the vocabulary size. We use *scores* as a general term for numerical values used in token selection. A linear output head produces *logits* $Wh(x) + \beta$ before softmax [3], [6]. When bias addition acts directly on these outputs, we distinguish the *original logits* from the resulting *biased logits*. If temperature scaling or clipping precedes bias addition, our numerical analysis instead applies to the *processed scores* entering that operation [7].

For a fixed input, write $q = q(x)$ and let $\mathbf{1} \in \mathbb{R}^V$ denote the all-ones vector. For any scalar $s \in \mathbb{R}$,

$$\text{softmax}(q + s\mathbf{1}) = \text{softmax}(q). \quad (1)$$

Writing the mean-centering projection as $C = I - \mathbf{1}\mathbf{1}^\top/V$ gives

$$q = Cq + \mu\mathbf{1}, \quad \mu = \mathbf{1}^\top q/V. \quad (2)$$

Here, μ is the mean score, and Cq contains the deviations from that mean. Probabilities determine the centered component Cq in exact arithmetic, but not μ . An ideal biased-token interface

$$O_q^{\text{ideal}}(b) = \arg \max_i (q_i + b_i) \quad (3)$$

has the same ambiguity. This is the common freedom remaining in the logprob-free extraction setting of Carlini et al. [3].

Let A be a selected token and $a = q_A$ its original score, which we call the anchor. Any recovered anchor fixes the common additive component: if an earlier extractor uses

reference token R and returns $d_i = q_i - q_R$, set $r_i = d_i - d_A$. Then $q_i = a + r_i$ and $\mu = a + \sum_i r_i / V$. The local protocol below recovers a score pair and its tie-breaking priority; Section 3.7 extends it to selecting the anchor.

The ambiguity also has an architectural interpretation. Replacing a linear output head W by $W + 1v^\top$ changes the scores by $(v^\top h(x))1$ without changing ideal probabilities. The lost scalar can therefore vary with the input and depend on a shared head component. Softmax-preserving output-head reparameterizations and their quantization effects have been studied for output-head optimization [8]. We ask whether token responses to chosen biases can recover an original logit and thereby determine the missing mean μ when the complete relative logits are known.

2.2. The Numerical Interface

We use the IEEE 754-2019 binary32 format (FP32), which has 24-bit significand precision [9]. For a fixed input, let q be a vector of finite binary32 values. Let RN_{32} denote round-to-nearest, ties-to-even: halfway cases select the adjacent binary32 value with an even least significant significand bit [10]. The observed interface is

$$O_q(b) = \arg \max_i \text{RN}_{32}(q_i + \text{RN}_{32}(b_i)). \quad (4)$$

Each bias is converted independently before one binary32 addition. Scores remain fixed across queries for the same input. A fixed, initially unknown global tie-breaking order selects among tokens with equal maximum rounded scores. This selection is distinct from midpoint rounding; only the selected token identity is returned.

The recovery proofs use the separate bias conversion, rounded addition, and greedy selection in (4). Conversion precision, inserted quantization, and fused operations can alter the comparisons [11]. Section 5.4 extends the comparison protocol to specified errors in the returned tokens while keeping the underlying scores fixed.

A binary interval $[2^{e-1}, 2^e)$ is a *binade*. Binary32 values in its normal range have spacing $\Delta = 2^{e-24}$. At $P = 2^e$, spacing doubles. We use this change to produce score-dependent rounding responses. The experiments use normal values far from overflow and underflow. The abstract extension to very small exponents additionally requires gradual underflow rather than flush-to-zero.

2.3. Threat Model

Protected information and adversary objective. The provider withholds the logits for each input. An external client seeks their original numerical values, including the common additive component left unidentified by ideal probabilities and biased argmax queries. Recovering one original anchor fixes this component when relative logits are known. Our local algorithm additionally recovers the target score t and the pairwise tie-breaking priority; its exact-recovery criterion requires agreement with all three values at the

specified decoder stage. Figure 1(a) separates the client-visible interface from the private computation.

Capabilities and observations. The client can repeat its own fixed input, request one greedy output token, and assign independent additive biases to selected token IDs within a public magnitude limit H . It adapts each bias to the preceding token responses. The selected token identity and tokenization are assumed observable. No returned probabilities, scores, score gaps, weights, hidden states, execution timings, co-location, model modification, or access to another user's request is required. The client does not choose the service's tie-breaking order, arithmetic implementation, or processing order.

Numerical assumptions and prior knowledge. The service follows the stable-score operator in (4), with an unknown fixed global tie-breaking order. Greedy selection alone does not establish score stability; Section 5.4 evaluates execution changes and extensions for noisy responses. The client knows this arithmetic model and a stated reachable score interval. The stated interval is required side information for the recovery theorem. In the single-window generation evaluation, separate calibration inputs select a public interval before testing; the test inputs do not supply scores to the observer. Precision, range, and stability are checked privately after predictions are frozen, and all fixed test inputs remain in the accounting.

The experiment evaluator can inspect private scores to measure success; the recovery controller cannot. In a search over multiple public intervals, uniqueness is defined relative to the prespecified search domain.

3. Logit Leakage and Recovery

3.1. Leakage Mechanism

Equation (3) makes every ideal query invariant under $q \mapsto q + s1$. Common-shift variants therefore yield identical transcripts even for adaptive, randomized observers: conditioned on the observer's randomness, identical responses lead to identical subsequent queries. Floating-point addition need not preserve this equivalence.

We compare complete *unperturbed* probability outputs with active numerical queries. An interface that returns probabilities after the same bias operations may also expose the additional information.

A distinguishing query. Consider two binary32 vectors

$$q = (30, 28), \quad q' = (29.875, 27.875). \quad (5)$$

Their difference is a common shift. Their max-centered vectors are both $(0, -2)$, and their unperturbed probabilities are equal. Give the anchor bias 2 and the target bias $4 + 3 \cdot 2^{-21}$. All supplied values are representable in binary32.

For q , the anchor becomes 32. The target's exact sum is $32 + 3 \cdot 2^{-21}$, which rounds to 32 because binary32 spacing above 32 is 2^{-18} . If the anchor has priority, it wins. For q' , the anchor becomes 31.875. The target rounds to $31.875 + 2^{-19}$, because spacing below 32 is 2^{-19} , and wins

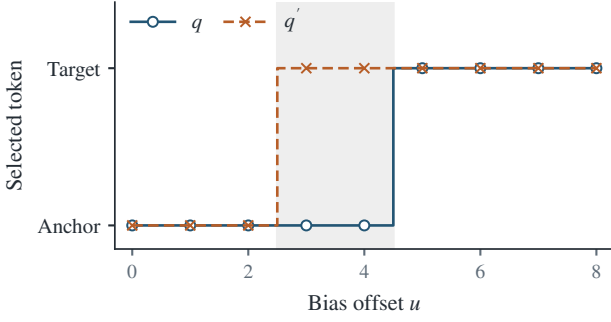


Figure 2. Token responses for $q = (30, 28)$ and $q' = q - \frac{1}{8}\mathbf{1}$. The anchor bias is 2; the target bias is $4 + u2^{-21}$ for integer offsets u . The vectors have identical unperturbed probabilities. Shading marks offsets with different winners. The anchor wins ties between rounded scores; connecting lines guide the eye.

strictly. Thus one numerical query distinguishes the two score vectors although their passive outputs are the same.

This example isolates the source of information. In exact arithmetic, the target would win for both vectors at the stated bias. Rounding distinguishes the two vectors, whose original relative scores are identical. Section 6.2 evaluates how preprocessing affects these differences over complete vocabularies.

3.2. Recovery Setting

Pair discovery and binary responses. An initial unbiased query selects the anchor, and a second query suppresses it by an allowed negative bias to select a target. The anchor is the original greedy winner, so $a \geq t$. Local recovery gives the anchor only nonnegative biases. No unmodified third token can then displace it under the fixed global tie-breaking order, making the subsequent response binary. Pair discovery is counted separately from the recovery bound.

Our recovery construction requires enough bias to move the two scores across a chosen binary exponent boundary $P = 2^e$, where the spacing between adjacent binary32 values increases from Δ just below P to 2Δ just above it. Queries must also resolve the unknown tie-breaking priority: which token wins when the rounded scores are equal. We first specify a reachable score interval; Section 3.6 addresses searches over multiple intervals and bounded-bias nonidentifiability.

Let $P = 2^e$ with $-125 \leq e \leq 127$, assume gradual underflow and no overflow in probe arithmetic, and set $\Delta = P/2^{24}$. Define

$$K = \left\lfloor \frac{\min(H, P/2)}{\Delta} \right\rfloor, \quad H_0 = K\Delta, \quad F = P - H_0. \quad (6)$$

For the anchor score a and target score t , assume $K \geq 4$ and

$$a \geq t > F, \quad a \in [\ell, u] \subset (F, P - 2\Delta), \quad (7)$$

where ℓ, u are Δ grid points. Both scores lie in the same binade, where binary32 values have spacing Δ . Their gap $d = a - t$ is therefore a multiple of Δ .

The interval in (7) provides enough bias to move the scores across P while keeping every recovery bias between 0 and H_0 . The margin below P also permits the adjacent probes used to resolve the priority and anchor residue.

3.3. Recovery Algorithm

Algorithm 1 implements the positive-score recovery sequence in Figure 1(c). FirstWinningIndex finds the first target-winning bias index; ResolveTie identifies the gap and priority. ResidueCandidates selects anchors with the observed residue for BinarySearch, using the monotone response $\text{Probe}_{d,\tau}$. Write $O(b_0, b_1)$ for biases on tokens 0 and 1, with zero bias elsewhere. All subroutines use the public range parameters.

Algorithm 1. Exact recovery of a positive score pair.

Input: Token oracle O , original winner 0, target 1, and public P, H, ℓ, u satisfying (7); Δ, K, F follow (6).

Output: Scores (a, t) before bias addition and tie-breaking priority τ .

```

1  $j \leftarrow \text{FirstWinningIndex}(O, \Delta, K)$ 
2  $d_0 \leftarrow (j - 1)\Delta$ 
3  $\tau \leftarrow \text{ResolveTie}(O, d_0)$ 
4  $d \leftarrow d_0 + \tau\Delta$ 
5  $p_0 \leftarrow \max\{\ell, F + d + \Delta\}$ 
6  $\mathcal{S} \leftarrow \text{ResidueCandidates}(O, d, \tau, p_0, u)$ 
7  $a \leftarrow \text{BinarySearch}(\mathcal{S}, \text{Probe}_{d,\tau})$ 
8 return  $(a, a - d, \tau)$ 
```

Recovering the gap and tie-breaking priority. Label the original winner as token 0 and the target as token 1. Their pairwise tie-breaking priority is $\tau = 0$ if token 0 wins when their rounded scores are equal, and $\tau = 1$ otherwise. When only the target receives bias $k\Delta$, the first winning grid point j determines exactly two possibilities:

$$\begin{aligned} \tau = 0 : \quad d_0 &= (j - 1)\Delta, \\ \tau = 1 : \quad d_1 &= j\Delta = d_0 + \Delta. \end{aligned} \quad (8)$$

The target loses at $k = 0$ by the original-anchor definition and wins at $k = K$ because $t + H_0 > P > a$. FirstWinningIndex finds j by integer binary search in at most $\lceil \log_2 K \rceil$ queries; the endpoint responses require no calls. Around the comparison threshold, the sums are representable grid values, so the search determines the two gap hypotheses exactly.

Three rounding probes identify the implementation's tie-breaking priority without assuming a token-ID order. Define

$$p = \max\{\ell, F + d_0 + \Delta\}. \quad (9)$$

For $r = p, p + \Delta, p + 2\Delta$, submit

$$b_0 = P - r, \quad b_1 = P - r + d_0 + \Delta. \quad (10)$$

Under the second hypothesis in (8), the two exact sums coincide and the target has priority, so all three replies are token 1. Under the first hypothesis, the responses follow the

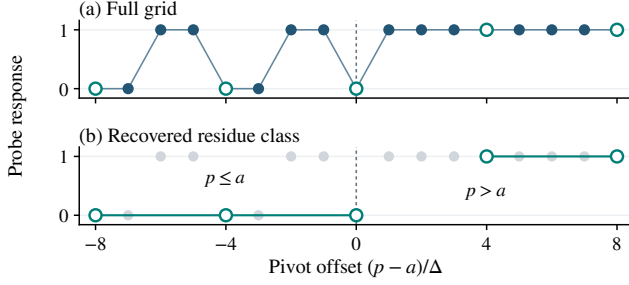


Figure 3. Eliminating the periodic response. (a) Full-grid probes are periodic for $p \leq a$ and constant for $p > a$. (b) The recovered residue class (open markers) gives a monotone comparison; gray points are excluded. The change at $p = a$ corresponds to floating-point spacing changing from 2Δ to Δ . Lines connect discrete queries. The horizontal origin uses the unknown a for illustration.

rounding pattern derived next and cannot all be token 1. ResolveTie returns $\tau = 1$ exactly when all three replies are token 1, thereby resolving the gap ambiguity.

The equality case $a = t$ is included. It implies $\tau = 0$ because token 0 was the original winner, and the first winning target bias is Δ .

Identifying the anchor residue. Suppose d and τ are known. For a pivot $p \in \Delta\mathbb{Z}$ with $F + d + \Delta \leq p \leq P$, give the anchor bias $P - p$ and the target bias $P - p + d$. Their exact biased scores are equal:

$$x = P + (a - p) = P + k\Delta, \quad k = (a - p)/\Delta. \quad (11)$$

Add one further Δ to whichever token has lower priority. After incorporating the increment, convert each bias independently to binary32 and add it to its score with one rounded addition.

Define $\text{Probe}_{d,\tau}(p)$ as the indicator that token $1 - \tau$ wins this query, and write $f(k)$ for its response at $k = (a - p)/\Delta$. Each evaluation makes one oracle call. Below P , both x and $x + \Delta$ are representable and $f(k) = 1$. At or above P , ties-to-even rounding operates on a grid of spacing 2Δ , yielding

$$f(k) = \begin{cases} 1, & k < 0, \\ 0, & k \geq 0, k \bmod 4 \in \{0, 3\}, \\ 1, & k \geq 0, k \bmod 4 \in \{1, 2\}. \end{cases} \quad (12)$$

The nonnegative pattern is 0, 1, 1, 0 (Figure 3(a)). In particular, it contains no run of three ones. This establishes the priority test above because its initial pivot satisfies $p \leq a$ under either gap hypothesis.

Set $p_0 = \max\{\ell, F + d + \Delta\}$. Using at most three queries, ResidueCandidates either identifies $a = p_0$ or decodes the residue $\rho = ((a - p_0)/\Delta) \bmod 4$. It returns $\mathcal{S} = \{p_0\}$ in the former case and $\mathcal{S} = [p_0, u] \cap (p_0 + \rho\Delta + 4\Delta\mathbb{Z})$ otherwise. A singleton interval needs no query. Appendix A gives the response table, the zero-offset exception, and the payload bounds.

Once the residue is known, restrict candidate anchors to that residue class, spaced 4Δ apart. Both a and a candidate

pivot p lie in this class, so $(a - p)/\Delta \in 4\mathbb{Z}$. Equation (12) therefore reduces to

$$f((a - p)/\Delta) = 1 \iff a < p. \quad (13)$$

Figure 3 illustrates this restriction to a monotone comparison. BinarySearch queries $\text{Probe}_{d,\tau}$ on $\mathcal{S}$, retaining candidates below the pivot after response 1 and those at or above it after response 0. It recovers a in at most $\lceil \log_2 |\mathcal{S}| \rceil$ queries, including zero for a singleton; then $t = a - d$.

3.4. Correctness and Query Complexity

Theorem 1 (Local pair recovery). *Under (4) and (7), with token 0 the original winner and a fixed global tie-breaking order, Algorithm 1 recovers (a, t, τ) exactly. If the legal anchor interval after determining the gap contains $N = (u - p_0)/\Delta + 1$ grid points, it uses at most*

$$\lceil \log_2 K \rceil + 6 + \lceil \log_2 \lceil N/4 \rceil \rceil \quad (14)$$

binary recovery queries. For binary32 this is at most 50.

Proof. The target-only search gives the two gap hypotheses exactly. The three probes in (10) identify the correct hypothesis by (12). At most three more probes identify the anchor residue, including the $k = 0$ exception. The remaining class contains at most $\lceil N/4 \rceil$ points, and (13) permits ordinary binary search. The special case of a singleton interval terminates earlier. Every pivot keeps the anchor bias nonnegative and all payloads within H_0 ; the original-anchor property excludes third-token responses. Finally, $K \leq 2^{23}$ and $N \leq K$, giving $23 + 3 + 3 + 21 = 50$. $\square$

For binary precision $p \geq 3$, the same full-binade construction gives a worst-case upper bound $2p + 2$, under the corresponding representability assumptions.

The full-binade domain in (7) contains $(K - 2)^2$ admissible pair-and-priority states. For $a = F + i\Delta$, with $1 \leq i \leq K - 2$, targets $t = F + j\Delta$, $1 \leq j \leq i$, admit $2i - 1$ states: two priorities for each $j < i$ and only the original-anchor priority for $j = i$. A binary decision tree of depth T has at most 2^T leaves. Thus any deterministic algorithm that always recovers both scores and their priority has worst-case binary query cost at least

$$\lceil \log_2 (K - 2)^2 \rceil, \quad (15)$$

where $K = 2^{p-1}$; for binary32, the bound is 46.

Existing multiclass extraction methods use more informative responses to recover many relative coordinates [3], with costs measured per coordinate.

3.5. Negative Scores and Mixed Precision

Logits can be negative. A negative exponent boundary reverses the direction of the spacing change and admits a separate construction without shifting the service's scores. Let $S = 2^e$, $\delta = S/2^{23}$, $K = \lfloor \min(H, S)/\delta \rfloor$, $H_0 = K\delta$,

and $F = -S - H_0$. Assume $K \geq 2$, representable $\delta/2$, no overflow or flushing of required subnormals, and

$$a \geq t > F, \quad a \in [\ell, u] \subset (F, -S - \delta]. \quad (16)$$

The public endpoints lie on the δ grid.

Theorem 2 (Negative-score recovery). *Under (4) and (16), the score pair and unknown fixed tie-breaking priority can be recovered in at most*

$$\lceil \log_2 K \rceil + 2 + \lceil \log_2 \lceil N/2 \rceil \rceil \leq 47 \quad (17)$$

binary recovery queries, where N is the remaining number of legal anchor grid points after gap recovery. The worst-case binary-query lower bound for the full-binade pair-and-priority problem is 46.

The gap search is unchanged. One half-step query in the finer region resolves the tie-breaking priority. A further query at a pivot above every legal anchor identifies the anchor’s parity. Restricting pivots to that parity turns the rounding response into a monotone comparison. Appendix C proves the construction and its payload bounds. Changing to another known IEEE rounding direction permits modified probes with the same uniform upper bound; a mode change alone is therefore not a general suppression argument.

Known score precision restricts both searches to the storage grid. Storage-aligned probes also remove the fine-grid residue step, giving at most $2s - 1$ recovery queries for s -bit stored scores when the bias and addition precision is at least $s + 2$ and the local range and representability conditions hold: 15 for BF16 and 21 for binary16 with binary32 addition (Appendix D). The required grid is that of the scores entering bias addition; weight or forward-pass precision alone does not establish it. Lower-precision storage retains the channel when promoted scores accept finer biases.

3.6. Consistency Checks and Abstention

Outside a supplied local interval, we use local recovery to generate candidates over a fixed collection of windows. Each candidate must reproduce the combined transcript under the declared operator; zero or multiple surviving candidates cause abstention. A unique candidate is accepted relative to the specified search domain.

Bounded biases can leave distinct score vectors observationally equivalent (Appendix B). Thus, transcript consistency establishes uniqueness within the searched domain; exact recovery additionally requires the local arithmetic and range assumptions.

3.7. Adaptive Anchor Selection

If no unique candidate survives the current pair’s checks, discard its anchor and retry. Every query fixes bias $-H$ on previously discarded anchors S . A query with zero bias elsewhere selects anchor A ; additionally suppressing A selects target T . A response from S , or failure to select a distinct target, stops the search.

Preservation of local recovery. The fixed mask defines a baseline vector with entries $\text{RN}_{32}(q_i - H)$ on S and q_i elsewhere. Its greedy winner is A , so nonnegative anchor biases preserve binary responses under the fixed tie-breaking order. For $A, T \notin S$, the service computes $\text{RN}_{32}(q_A + b_A)$ and $\text{RN}_{32}(q_T + b_T)$ directly from the original logits. The local theorems therefore apply whenever the selected pair satisfies their range assumptions.

The search returns a unique candidate passing all checks or stops after R pair attempts, biasing at most $R + 1$ tokens per query. Window searches, checks, and retries contribute to the complete recovery cost.

3.8. Output-Head Extraction

Building on the output-head factorization of Carlini et al. [3], we combine complete relative logits with recovered anchors to determine a bias-free linear head up to an invertible hidden-basis transformation. Bias bisection against the unbiased winner estimates each relative coordinate (Appendix G). Centering these estimates across the vocabulary gives a matrix that we factor as $\hat{B}\hat{X}$. The anchors determine the vector μ of mean logits over extraction inputs. Fitting $\mu = u\hat{X}$ gives the shared row u , and adding it to every vocabulary row completes the head as $\hat{W} = \hat{B} + \mathbf{1}u$.

If the centered head and hidden-state matrix have full hidden rank, these factors represent CWG and WG for an invertible G ; Appendix G derives the identity and handles rank loss under centering. Relative-coordinate and floating-point errors make the numerical head estimates approximate even when the anchors are exact. For a held-out input, queried relative coordinates determine its coordinates in the learned basis; u predicts the mean without using that input’s anchor.

4. Experimental Setup

4.1. Models and Interfaces

We evaluate 11 checkpoints from seven families, ranging from 14M to 4B parameters: Pythia [12], GPT-2 [13], GPT-Neo [14], OPT [15], BLOOM [16], SmolLM2 [17], and Qwen3 [18]. Generation experiments invoke unmodified `model.generate` for one greedy token with the `SequenceBiasLogitsProcessor` in Transformers 4.57.1 [7], [19]. Forward computation uses FP32 or BF16 as specified below; bias addition uses FP32.

Serving experiments use Qwen3-0.6B and SmolLM2-360M on research-controlled vLLM 0.10.2 [20], SGLang 0.5.3.post2 [21], and llama-server at commit 7f2dd88b0ac3 [22]. GPU servers use BF16 weights; llama.cpp uses CPU FP32 GGUF. All add binary32 biases. Output-head extraction uses pretrained TinyStories-260K [23], whose bias-free head has a complete 512-token vocabulary, through a research-controlled CPU FP32 HTTP service.

4.2. Observation and Validation Protocol

The observer receives token IDs; evaluator-only hooks record original logits as ground truth. End-to-end runs obtain every response through generation calls, while cached evaluations apply the decoder to saved full-vocabulary vectors behind the same token-only interface. Predictions are frozen before private scoring. Independent replay checks recorded responses, and selected serving traces are repeated without instrumentation. Calibration uses separate inputs; precision, budget, and framework comparisons reuse the specified input sets and count their model-input conditions separately. Each study below gives its input split and comparison protocol.

4.3. Evaluation Metrics

For true scores and tie-breaking priority (a, t, τ) and a returned estimate $(\hat{a}, \hat{t}, \hat{\tau})$, we define the maximum absolute score error and exact-recovery criterion as

$$E_{\max} = \max\{|\hat{a} - a|, |\hat{t} - t|\}, \quad (18)$$

$$\text{Exact recovery} \iff E_{\max} = 0 \wedge \hat{\tau} = \tau.$$

For the finite, nonzero scores in our recovery domains, score equality is bitwise equality of the binary32 representations. Even a one-ULP discrepancy is counted as incorrect. We report recovered, incorrect, and unresolved cases over all attempted inputs, as well as exact recovery conditional on the stated range assumptions.

Adaptive anchor recovery requires $\hat{a} = q_A$ for the returned token A ; we also evaluate the selected pair under (18), retaining all attempted inputs.

A *recovery query* is one call used after token-pair selection; the two selection calls are counted separately. A *generation call* recomputes the model and applies its decoding operations. A *cached decoder evaluation* applies those operations to a saved score vector. We report these computational costs separately.

5. Evaluation

We address four research questions:

- **RQ1:** Can token responses recover original logits across models and numerical precisions, and how does range coverage affect recovery? (Section 5.1)
- **RQ2:** Can the adaptive procedure recover logits through serving APIs, and what is its complete query cost? (Section 5.2)
- **RQ3:** Can recovered anchors extend token-only output-head extraction from *CWG* to *WG*? (Section 5.3)
- **RQ4:** How do execution changes and token-response errors affect recovery? (Section 5.4)

5.1. RQ1: Recovery across Models and Precisions

The FP32 evaluation uses the same 64 fixed test inputs per checkpoint. Eight prefixes, including Chinese, JSON, and always-A instructions, are balanced across arithmetic and word-continuation text to exercise the numerical interface. The single-window baseline decodes under one evaluator-calibrated interval; the adaptive controller searches windows, checks the combined transcript, and reselects anchors as defined in Sections 3.6–3.7.

Before testing, eight separate calibration inputs select one public positive or negative interval by maximum calibration coverage, breaking coverage ties by ascending exponent and then positive before negative. The candidate boundaries are $\pm 2^e$ for $e = -6, \dots, 10$, with the theorem’s endpoint margins and primary bias limit $H = 100$. Calibration uses evaluator score access to supply the assumed interval; it is not a token-only range-estimation procedure. No test score reaches the observer, and every fixed input is retained.

A post-hoc cached evaluation compares all 34 signed windows, first with the original pair and then with selection capped at eight pairs. It also evaluates larger bias budgets and generic versus known-precision recovery on the same vectors (Appendix E). End-to-end adaptive runs use 64 FP32 GPT-Neo inputs and 32 BF16 inputs each for OPT and Qwen3 at $H = 100$. Each query recomputes the model and returns only the selected token.

Across 32,984 generation calls on 704 FP32 cases, all 504 covered pairs and priorities recover exactly: 504/704 (71.6%) overall and 504/504 within the calibrated intervals (Table 1). GPT-2 and GPT-Neo exercise negative recovery on 54 and 13 covered inputs. Every response matches full-vocabulary binary32 replay, and original-logit hashes remain stable within each input, validating the tested generation path in (4).

Outside the calibrated interval, the single-window baseline returns 103 incorrect candidates and leaves 97 unresolved. Cached window search recovers 565/704 inputs with the original pair; adaptive selection increases this to 632/704 (89.8%) with zero incorrect accepted candidates. Nine checkpoints reach 64/64, using at most three pairs. All 72 remaining inputs exhaust eight attempts outside the tested local domains.

The larger-budget ablation reaches 64/64 for Pythia-160M at $H = 256$ and BLOOM at $H = 160$, bringing cached adaptive coverage to 704/704. BLOOM’s eight unresolved $H = 100$ targets require $512 - t \approx 106.77\text{--}143.17$ bias, explaining their exclusion by that budget. Table 1 retains both budgets; the artifact contains the full budget sweep.

With all 34 windows and anchor selection, end-to-end GPT-Neo recovery reaches 64/64 at $H = 100$; 45 inputs require another anchor. The complete procedure averages 2,428 generation calls per input, including selection, window searches, checks, and retries. This cost measures the broad search beyond the local recovery bound.

BF16 logits. Four checkpoints additionally use BF16 forward computation: Pythia-160M, OPT-1.3B, BLOOM-

TABLE 1. FP32 RECOVERY WITH A CALIBRATED INTERVAL AND ADAPTIVE ANCHOR SELECTION.

Model	H	Generation			Cached multi-window search		
		Boundary	Exact	Calls	Boundaries	Fixed	Selected
Pythia-14M	100	16	64	3,284	16	64	64
Pythia-160M	100	–	0	1,600	–	0	0
Pythia-1.4B	100	16	64	3,269	16	64	64
GPT-2 Medium	100	–64	54	3,136	–64, –32	62	64
GPT-Neo-125M	100	–1	13	2,363	–2, –1, –0.5	19	64
OPT-1.3B	100	16	58	3,273	8, 16	59	64
BLOOM-560M	100	512	56	3,095	512	56	56
SmolLM2-360M	100	32	64	3,288	32	64	64
SmolLM2-1.7B	100	16	32	3,279	16, 32	64	64
Qwen3-0.6B	100	32	63	3,254	32	63	64
Qwen3-4B-Instruct	100	64	36	3,143	32, 64	50	64
Larger-budget ablation on the same inputs							
Pythia-160M	256	1024	64	3,156	1024	64	64
BLOOM-560M	160	512	64	3,198	512	64	64

Notes. Counts are out of 64 inputs. Generation uses one evaluator-calibrated interval; Calls exclude calibration. Its 704 primary cases include 103 incorrect and 97 unresolved outcomes. Cached Boundaries list accepted windows (–: none), from $\pm 2^e$, $e = -6, \dots, 10$. Fixed uses one pair; Selected tries up to eight. Accepted anchors meet exact pair-and-priority recovery; other cached outcomes are unresolved. Pythia-160M at $H = 100$ has zero calibration coverage (–); its 1,600 calls use the fallback $P = 2^{-6}$.

TABLE 2. BF16 RECOVERY WITH FP32 BIAS PROCESSING.

Model	H	Generation			Cached
		Exact	Generic	Known	Selected
Pythia-160M	100	0	–	–	0
Pythia-160M	256	32	48	13	32
OPT-1.3B	100	30	50	14–15	32
BLOOM-560M	100	28	46–48	11–13	28
BLOOM-560M	160	NR	NR	NR	32
Qwen3-0.6B	100	30	49–50	14–15	32

Notes. Counts are out of 32; both algorithms agree. Generic and Known columns give generic and BF16-aware recovery queries, excluding pair selection. Both use FP32 bias addition. Cached selection uses multiple windows and up to eight pairs, with zero incorrect returns. –: no covered case; NR: generation not run.

560M, and Qwen3-0.6B. For each, the same 32 inputs are evaluated with the binary32 recovery algorithm and the algorithm using the known logit precision; the generation processor still adds FP32 biases. We verify the logit lattice privately. A separately specified budget ablation repeats configurations with zero calibration coverage at $H = 100$ using $H = 256$. Selection depends on calibration, not test success. It uses the same test inputs and keeps the original-budget outcomes.

The channel persists with BF16 logits and FP32 bias processing in the tested configurations. At $H = 100$, both controllers recover all 88 covered inputs out of 128; the larger-budget Pythia-160M intervention adds 32 covered paired cases. Complete original logit vectors, token pairs, and rendered inputs match bit for bit between controllers. The original logits lie on a BF16 lattice, as required by Proposition 3.

Table 2 shows 11–15 local queries with known BF16

precision versus 46–50 with generic recovery. Cached anchor selection covers 128/128 inputs at the displayed larger budgets. In end-to-end adaptive runs at $H = 100$, OPT reaches 32/32 with both algorithms on bitwise-identical original logits. Mean complete costs are 439 calls with known precision and 1,265 with generic recovery. Qwen3 reaches 32/32 with generic recovery at 1,060 calls per input. Each model needs a second pair on two inputs.

5.2. RQ2: Recovery in Serving Systems

Each model–framework configuration uses 64 fixed inputs, $H = 100$, and one interval calibrated on eight separate inputs with evaluator-only scores. An independent client submits biases through the generation API under fixed execution settings. Read-only probes record evaluation scores; every eighth input’s complete trace is rerun on a fresh uninstrumented server. Appendix H specifies the interface fields.

Cached ablations on these 384 vectors compare $P = 16$, $P = 32$, and both, with up to eight pairs and six candidate checks. We freeze the two-window controller before evaluating it end to end on Qwen3 through each API, using the original 64 inputs and 64 disjoint held-out inputs per framework. All payloads, including repeats, are submitted to the server.

The single-window baseline at $P = 32$ recovers exactly 301/384 inputs: 37/36/36 Qwen3 cases across vLLM/SGLang/llama.cpp and all 192 SmolLM2 cases (Table 3). The other 83 return incorrect candidates outside the calibrated interval; none is unresolved (Section 8.1). Covered cases use 49–50 recovery queries plus two selection calls. All 19,686 responses match binary32 replay, with stable original logits; 2,447 uninstrumented-server queries reproduce their tokens.

TABLE 3. END-TO-END RECOVERY IN LLM SERVING SYSTEMS.

Framework	Single window		Adaptive Qwen3		
	Qwen3	SmolLM2	Dev.	Test	Calls
vLLM	37	64	64	64	119.6
SGLang	36	64	64	64	120.4
llama.cpp	36	64	64	64	115.4

Notes. Recovery counts are out of 64; $H = 100$. Models: Qwen3-0.6B and SmolLM2-360M. Single window uses $P = 32$. Adaptive uses $P = 16, 32$, checks, and up to eight pairs. Dev./Test denote the development and disjoint held-out inputs. Calls are mean complete API calls across both adaptive sets, including selection, searches, checks, and retries; calibration and independent replay are excluded.

Single-window checks still accept seven incorrect Qwen3 candidates on llama.cpp: four at $P = 16$ and three at $P = 32$. All lie outside the assumed local domain and contradict the other window’s transcript. Combining both windows recovers 58/57/56 of the 64 development inputs across the three frameworks with the original pair; anchor selection raises each to 64/64.

The frozen two-window controller recovers 64/64 development and 64/64 held-out inputs per framework: 384/384 exact pairs and priorities, with no incorrect or unresolved returns. Another anchor is needed in 39 cases. Complete cost averages 118.5 API calls per input condition; per-framework medians are 109/109/108 and maxima are 310/312/215. All 45,497 responses match binary32 replay, and 6,271 uninstrumented-server queries reproduce their tokens. Appendix E gives the search-cost breakdown.

5.3. RQ3: Output-Head Extraction

We evaluate the completion in Section 3.8 on three independently generated input sets for TinyStories-260K. Sets 1 and 3 contain 96 extraction inputs and 16 held-out inputs; set 2 contains 192 and 32. Random token prefixes have lengths 4, 8, and 16. Each HTTP request recomputes 64 copies of the input, applies independent biases, and returns token IDs; padding fixes the batch shape. The observer uses $H = 100$, boundaries $P \in \{1, 2, 4, 8, 16, 32\}$, and at most four pair attempts.

The observer receives no hidden width. Rank selection uses the largest adjacent singular-value ratio, requiring a ratio of at least 10, a preceding singular value above 10^{-7} of the largest, and two input columns beyond the selected rank.

Separate programs collect observations, extract factors, and validate frozen results. The centered and completed factors share one basis alignment fitted on 256 vocabulary rows, centered using those rows; head errors are evaluated on the remaining 256. Held-out predictions use queried relative coordinates and u ; separately recovered test anchors are used only for evaluation. A post-hoc constant baseline predicts the average of the extraction inputs’ recovered means. An input-budget analysis refactors nested extraction prefixes with the held-out inputs fixed.

On TinyStories-260K, all three random-prefix input sets yield complete relative-coordinate estimates and exact anchors, pairs, and priorities for every extraction and held-out input. Each full extraction identifies rank 64, matching the hidden width. Table 4 reports head reconstruction and prediction of held-out mean logits from their queried relative coordinates.

Recovery of the shared component. Under the shared basis alignment, the completed head has relative Frobenius error 1.19×10^{-6} – 3.96×10^{-6} . The relative branch recovers its own target CW with RMSE 3.50×10^{-7} – 1.10×10^{-6} , but differs from the original W by RMSE 0.189 because it omits the shared row. The recovered shared row has RMSE 8.25×10^{-8} – 4.32×10^{-7} . Figure 4(a) shows the entrywise absolute errors before and after this completion. The extracted representation is WG , retaining the invertible hidden-basis ambiguity.

For the 64 held-out inputs, the learned map predicts the mean from relative coordinates with RMSE 3.11×10^{-7} – 7.21×10^{-7} . Predicting the average extraction-input mean instead gives RMSE 0.895–1.162, showing the variation captured by the map. Combining the predicted means with relative coordinates gives complete original-logit RMSE 2.15×10^{-6} – 3.60×10^{-6} . Appendix G reports numerical diagnostics.

Input budget and query cost. At 32–64 inputs, the rank rule selects rank 1 or remains unresolved: its full-rank candidate needs more than 64 input columns. All three sets select rank 64 at 72 inputs. For set 2, increasing the budget from 96 to 192 reduces relative head error from 2.75×10^{-6} to 1.19×10^{-6} (Figure 4(b)). Across the three sets, mean cost per extraction input is about 14,300 token decisions: 255–267 for the anchor and 14,019–14,029 for relative coordinates. Anchors account for about 1.8% of decisions; their sequential queries and the batched relative-coordinate probes have different latency costs. Table 4 separates extraction cost from held-out evaluation.

5.4. RQ4: Execution Stability and Response Noise

Stability of the numerical target. Greedy decoding can produce execution-dependent floating-point scores [24]. We test Pythia-14M and Pythia-160M with eight new inputs each, CPU FP32 inference, and batch sizes 1, 2, 4, 8. Each input–batch condition repeats six probes ten times: no bias, early and final gap search, priority identification, residue identification, and final anchor search. We also repeat complete recovery at fixed batch sizes 1 and 4.

Across 3,840 generation calls, unmodified greedy tokens remain unchanged, but the selected Pythia-14M and Pythia-160M scores move by up to 5 and 2 ULPs across batch conditions. Scores remain bitwise stable within each fixed input–batch condition. Complete recovery succeeds on 16/16 inputs at each of batch sizes 1 and 4, although 15/16 recovered pairs differ between conditions. Fixing execution conditions defines a stable recovery target in this study.

Errors in responses about a fixed target. To isolate response noise from score changes, we inject in-

TABLE 4. TOKEN-ONLY EXTRACTION OF TINYSTORIES-260K’S COMPLETE 512×64 OUTPUT HEAD AND HELD-OUT MEAN PREDICTION.

Input set	Extraction/test	Exact anchors	Relative head error	Mean RMSE	Token decisions	HTTP batches
1	96/16	96/96	3.96×10^{-6}	3.11×10^{-7}	1,370,786	45,873
2	192/32	192/192	1.19×10^{-6}	4.59×10^{-7}	2,740,717	91,584
3	96/16	96/96	3.35×10^{-6}	7.21×10^{-7}	1,372,446	46,928

Notes. Independent input sets use the same checkpoint. Head error is relative Frobenius error on 256 vocabulary rows excluded from basis alignment. Mean-logit RMSE evaluates prediction from held-out relative coordinates without their anchors. Anchor counts and costs cover extraction inputs only; all 64 separately evaluated test anchors are also exact. HTTP requests batch independently biased decisions. Probability queries: zero.

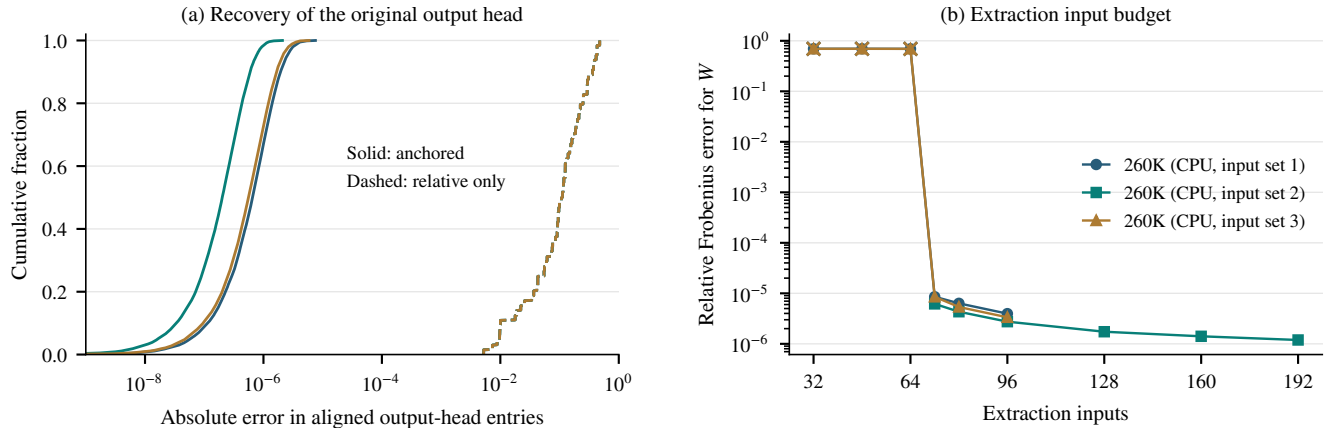


Figure 4. Token-only output-head extraction from TinyStories-260K across three independently generated input sets. (a) Empirical distributions of absolute entry errors against the original W , after the same basis alignment. Solid curves include the recovered shared component; dashed curves omit it. The relative branch’s own target is CW . (b) Output-head relative error as the extraction input count increases, with each set’s test inputs fixed. Crosses mark inferred ranks below 64; set 2 is unresolved at 32–64 inputs and has no corresponding error points. Lines connect measured counts.

dependent token-response flips with probability $p_{\text{flip}} \in \{0, 0.01, 0.05, 0.1, 0.2, 0.3, 0.4\}$ into the 504 previously covered FP32 cases. Five noise seeds per case give 2,520 simulated recoveries per method and noise level. Token pairs and public intervals are fixed before injection; the 200 out-of-range cases and pair discovery are outside this conditional study.

We compare four binary-response rules. *Single* uses one response; *Majority-9* votes over nine. The sequential likelihood-ratio rule (*LR*) uses error bound $\bar{p}_{\text{flip}} = \max(0.1, p_{\text{flip}})$ [25]. The confidence-interval rule (*CI*) samples until an interval excludes $1/2$, without a noise-rate bound, allocating error across predetermined sample sizes [26]. Both use recovery-wide error budget $\delta = 0.01$ for at most $M = 50$ logical comparisons. Each comparison is capped at 4,096 responses; an undecided comparison stops recovery without an estimate. Appendix F specifies the assumptions and tests. We report exact recovery and mean token responses after pair discovery.

Figure 5 and Table 5 evaluate independent flips of responses from fixed in-range pairs. At 10% flips, LR achieves higher recovery and lower mean cost than nine-response voting. LR uses a noise-rate upper bound; CI recovers all simulated cases at higher cost without that bound. The 2,520 attempts comprise 504 numerical cases and five noise seeds.

A separate condition selects uniformly between tokens

TABLE 5. RECOVERY AT 10% INDEPENDENT RESPONSE FLIPS.

Rule	Exact recovery	Mean responses
Single	17/2,520	47.8
Majority-9	2,411/2,520	437.4
LR	2,502/2,520	242.7
CI	2,520/2,520	2,944.2

Notes. Rules are defined in Section 5.4. Exact recovery requires both scores and priority; all 2,520 attempts per rule are counted. Responses include repetitions after pair discovery. No attempt is unresolved at this noise level.

with equal maximum rounded scores; strict comparisons remain exact. We compare CI with a rule that emulates a chosen fixed priority: return token 1 only when all 13 repeated responses are token 1, and token 0 otherwise. Its guarantee and score-pair recovery objective are given in Appendix F.

Under uniform random token tie breaking, CI leaves all 2,520 cases unresolved. Emulating fixed priority instead recovers 2,516/2,520 score pairs at mean cost 632.1. These rules estimate comparisons for fixed scores; they do not undo the score changes measured in the batch experiment.

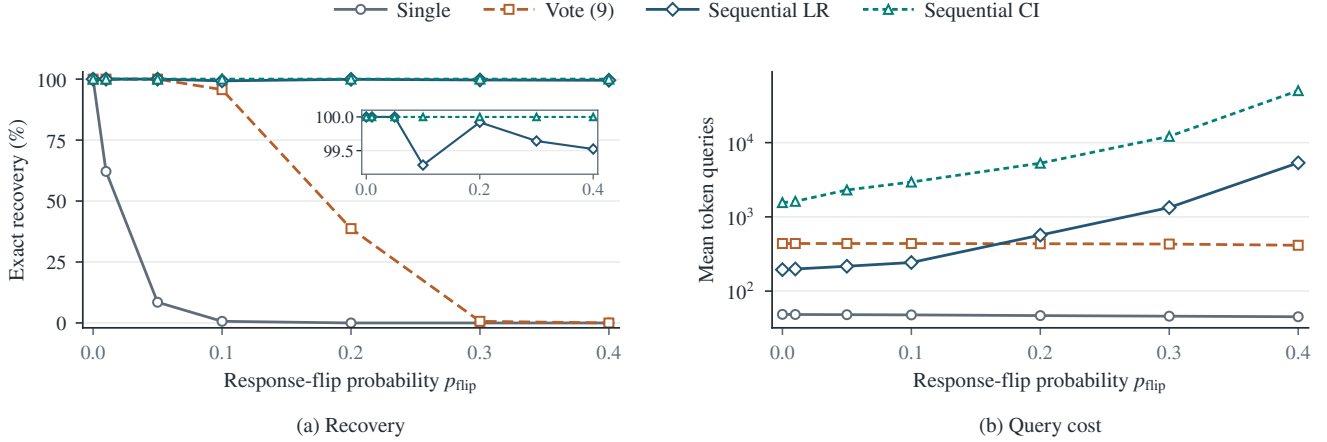


Figure 5. Exact recovery and cost under independent response flips. Single uses one response per comparison; Majority-9 uses nine-response voting; LR and CI use sequential likelihood-ratio and confidence-interval tests, respectively. Each point aggregates 504 fixed in-range pairs and five noise seeds. Cost counts responses after pair discovery. The inset enlarges LR and CI near 100%; the cost axis is logarithmic. Lines connect measured conditions.

6. Defense

6.1. Subtracting the Maximum before Bias Addition

Subtract the maximum score before adding any bias (Figure 1(d)): $A_i(q) = \text{RN}_{32}(q_i - \max_j q_j)$, computed coordinatewise.

Proposition 1 (Max subtraction before bias addition). *For every allowed input x , let the finite stored binary32 vectors satisfy the exact real relation $q'(x) = q(x) + c(x)\mathbf{1}$. Both services apply correctly rounded max subtraction A with consistent signed-zero handling. Suppose subsequent observable responses depend on the original scores only through $A(q(x))$, under the same conditional response kernel given the public history, input, and controls, including tie breaking and internal randomness. Then every adaptive observer obtains identical transcript distributions for any finite query budget or almost-surely finite stopping rule.*

Proof. For each input, $q'_i - \max_j q'_j = q_i - \max_j q_j$. Correct rounding therefore gives $A(q') = A(q)$ bit for bit. Couple the observer’s randomness and the services’ responses. Equal histories give the same next query and response distribution, so induction gives identical histories at every finite round. A common stopping rule yields identical stopped transcripts. $\square$

The exact-shift condition applies to the stored vectors. If a common shift is rounded earlier, it can already change their relative scores. For example, binary32 $q = (1, 1 - 2^{-24})$ centers to $(0, -2^{-24})$, whereas componentwise rounding of $q + 2^{24}\mathbf{1}$ gives $(2^{24}, 2^{24})$, which centers to $(0, 0)$. The prior rounding has changed the relative scores, so the proposition’s exact-shift premise no longer holds.

When the maximum is subtracted *after* bias addition, rounding has already mapped the candidates to potentially

different vectors. A later operation that preserves the winning token retains the distinguishing response. Stable softmax implementations already subtract a maximum to avoid overflow [10], [27]; the relevant question is its position relative to the exposed control.

6.2. Effect on Distinguishing Queries

The suppression experiment compares the original processor with subtraction of the maximum score either before or after bias addition. We take eight cached positions from each of two FP32 cohorts, Pythia-14M and Qwen3-0.6B, round their complete score vectors to grids 2^{-8} and 2^{-6} , and apply shifts $-0.25, -0.125, 0, 0.125, 0.25$. This gives 160 conditions in 32 groups, with bitwise-identical complete max-centered and FP32 log-softmax vectors within each group. We freeze recovery predictions and replay the union of observed bias payloads across shifts.

Figure 6 compares cross-shift winner disagreements for the original pipeline (Original), max subtraction before bias addition (Pre), and max subtraction afterward (Post). Applying 9,742 union payloads to the five shifts in each group gives 48,710 condition–payload evaluations and 38,968 cross-shift comparisons. Pre removes all 2,578 original disagreements; Post retains them.

Preprocessing changes 10,038 of the 48,710 original *biased* outputs. This is expected for payloads selected to expose fine rounding behavior: suppressing the channel changes those responses. In contrast, all 160 unperturbed argmax, softmax, and log-softmax outputs remain bitwise identical.

6.3. Bias-Control Preservation

To measure bias-control preservation, we use 16 evenly spaced inputs from each of the six serving configurations: 96 original cached vectors. Each receives 96 random sparse

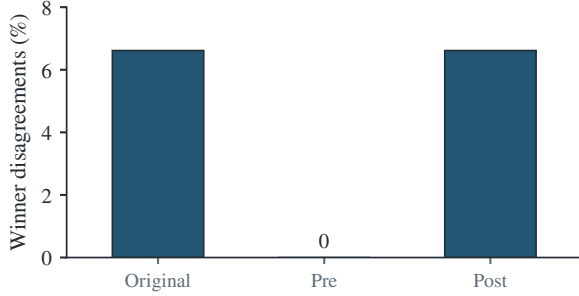


Figure 6. Cross-shift winner disagreements under three processing orders. Pre and Post apply max subtraction before and after bias addition, respectively. Rates use 38,968 comparisons: 2,578 disagreements for Original and Post, and 0 for Pre. The experiment uses fixed-grid decoder groups; unperturbed probabilities and argmax decisions remain bitwise unchanged in all 160 conditions.

biases, with eight tokens from the full vocabulary or top 64 and 16 draws per support and amplitude 0.5, 2, 10. It also receives 33 target-only biases $RN_{32}(d + kU/8)$, $k = -16, \dots, 16$, where d is the top-two gap and U is the binary32 spacing at their maximum absolute score. We compare processing orders, complete FP32 probabilities, and two fixed offline tie-breaking rules: smallest or largest token ID. All payloads satisfy $H = 100$. The inputs, seed, and bias-generation rules are fixed before execution; Appendix H gives the details.

Pre-bias max subtraction preserves all 96 unperturbed greedy decisions and complete probability vectors bit for bit. It changes no greedy decision across 9,216 random biases; the mean complete-probability L_1 difference is 6.08×10^{-8} and its maximum is 9.41×10^{-7} . The biases themselves change the unperturbed winner in 1,776 cases, including 1,737 of 4,608 top-64 controls.

Among 3,168 biases near the comparison threshold, preprocessing changes 738 maximizing sets and 369 greedy decisions under each fixed tie-breaking rule. Every change separates an original rounded tie; no strict-winner reversal occurs. Post-bias subtraction preserves all decisions and complete probabilities. Independent NumPy replay reproduces every FP32 processing vector. Thus, fine control near a decision boundary can change token selection even when the probability difference is small.

Processing order also matters for other controls. Repetition penalties can depend on the common additive component, so moving max subtraction across such a penalty can change its effect.

7. Related Work

7.1. Prediction-Interface and Language Model Extraction

Tramèr et al. established extraction from prediction interfaces [2]; subsequent work studies high-fidelity extraction and its relationship to recovered parameters [28], [29].

For language models, Morris et al. use biases and token responses to recover probability information [5]. Carlini et al. recover output-projection information and develop efficient logprob-free score-difference attacks [3]. Finlayson et al. analyze the low-dimensional linear structure exposed by API logits [4]. The softmax bottleneck is established [30]. Our contribution concerns the common additive component left unresolved by ideal normalized or biased-argmax observations.

Geometric output signatures use constraints shared across model inputs. Finlayson et al. exploit an ellipse associated with the output projection for source verification [31]. We study a per-input observation that recovers original logit values.

7.2. Finite Precision and Numerical Side Channels

Numerical analysis studies rounding and evaluation order [9], [10], [11], including stable softmax [27]. Andryscio et al. demonstrate operand-dependent floating-point timing leakage, including subnormal arithmetic [32]. We instead observe token selection under chosen biases near a normal exponent boundary.

Machine-learning implementations also expose execution-dependent signatures. iNNformant uses numerical differences in inference for device identification [33], and HSPI studies hardware signatures in model outputs [34]. Our setting fixes the implementation and recovers a score component through chosen decoder controls.

Quantamination exploits shared dynamic-quantization parameters to leak data across a batch [35], while FloatDoor modifies a model to turn platform-dependent numerical differences into a backdoor trigger [36]. Our mechanism requires neither a co-batched victim input nor modified model weights; it recovers the numerical values entering the exposed bias operation.

Kadav et al. study softmax-preserving output-head reparameterizations to improve output-head quantization [8]. Their work establishes that ideal-equivalent representatives can behave differently after quantization. Our direction is observational: determine which representative-dependent information can be identified externally from token responses, and provide a local recovery construction.

7.3. Decoder Controls and Interface Semantics

Decoding controls operate on model logits. Repetition penalties originate in work such as CTRL [37]; Baichuan 2 discusses their dependence on logit values beyond relative differences [38]. This dependence on the common additive component is therefore established. Our recovery construction instead concerns the rounding of an additive bias, whose ideal real-arithmetic counterpart preserves the common-additive-shift ambiguity.

Transformers and vLLM expose implementations of logit processing and token selection [19], [39]. We analyze the rounded addition used by such processors and evaluate

recovery end to end in LLM serving systems, linking the numerical operator to observable token responses.

8. Discussion

8.1. Range Identification and Future Work

Opaque endpoints require checks of bias resolution, operation order, score stability, and reachable ranges. Controlled passive-equality experiments isolate the mechanism; its prevalence in deployments requires separate measurement.

Range selection affects both coverage and cost. Outside the calibrated intervals, erroneous candidates often preserve the score gap while shifting both logits toward a window boundary. The seven single-window false acceptances in Section 5.2 illustrate this ambiguity; the second window rejects those candidates by providing a contradictory response. Globally justified range tests and recovery across binades remain open problems under bounded biases.

Extending complete output-head extraction to larger vocabularies requires measuring the relative-coordinate cost, which dominates the present study. Variable execution conditions require a target definition that accounts for changing scores; independent response-noise guarantees apply to a fixed target.

8.2. Research Ethics and Reproducibility

We use public checkpoints, synthetic inputs, research-controlled CPU/GPU resources, and one output token per bias setting; the output-head experiment batches independent settings. Controlled variants modify locally held vectors. All fixed cases remain in the records, including unmet preconditions. The study collects no other users' data and involves no access-control bypass.

Our experimental records include input generators, pinned model revisions, query transcripts, frozen predictions, and CPU replay checks (Appendix H).

We are preparing disclosure materials for the vLLM, SGLang, and llama.cpp maintainers, including minimal reproductions, tested configurations, and the pre-bias centering analysis. We plan to coordinate validation and mitigation with the maintainers.

Use of generative AI. We used ChatGPT and Codex to assist literature review, theoretical development, experimental design, implementation, data analysis, and manuscript drafting, revision, and translation. The authors are responsible for reviewing and validating AI-assisted material and for the accuracy and originality of the final work.

9. Conclusion

Floating-point bias addition can disclose the common additive component of logits that ideal probabilities and ideal bias comparisons leave unidentified. We characterize this difference, provide a local binary32 recovery method

with explicit query bounds, and verify exact recovery through complete generation calls. A token-only output-head experiment verifies the recovery of the shared row component under an invertible change of hidden basis. Preprocessing-order experiments identify a condition that suppresses disclosure. The results show that the numerical implementation of a token-only interface is part of its information-disclosure behavior.

References

- [1] OpenRouter, "API parameters," 2026, accessed 2026-10-01. [Online]. Available: https://openrouter.ai/docs/api_reference/parameters
- [2] F. Tramèr, F. Zhang, A. Juels, M. K. Reiter, and T. Ristenpart, "Stealing machine learning models via prediction APIs," in *25th USENIX Security Symposium*, 2016, pp. 601–618.
- [3] N. Carlini, D. Paleka, K. D. Dvijotham, T. Steinke, J. Hayase, A. F. Cooper, K. Lee, M. Jagielski, M. Nasr, A. Conmy, E. Wallace, D. Rolnick, and F. Tramèr, "Stealing part of a production language model," in *Proceedings of the 41st International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 235, 2024, pp. 5680–5705.
- [4] M. Finlayson, X. Ren, and S. Swayamdipta, "Logits of API-protected LLMs leak proprietary information," in *Conference on Language Modeling*, 2024.
- [5] J. X. Morris, W. Zhao, J. T. Chiu, V. Shmatikov, and A. M. Rush, "Language model inversion," in *International Conference on Learning Representations*, 2024.
- [6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems*, vol. 30, 2017.
- [7] Hugging Face, "Utilities for generation," Transformers 4.57.1 documentation, 2026, accessed 2026-10-02. [Online]. Available: https://huggingface.co/docs/transformers/v4.57.1/en/internal/generation_utils
- [8] A. Kadav, C. Flores, C. Arora, V. Kotte, H. Zheng, L. Yan, P. Shanmugasundaram, and T. H. King, "Softmax reparameterization for output-head quantization," arXiv:2609.31291, 2026, version 2.
- [9] IEEE, "IEEE standard for floating-point arithmetic," IEEE Std 754-2019, 2019. [Online]. Available: <https://standards.ieee.org/ieee/754/6210/>
- [10] D. Goldberg, "What every computer scientist should know about floating-point arithmetic," *ACM Computing Surveys*, vol. 23, no. 1, pp. 5–48, 1991.
- [11] N. J. Higham, *Accuracy and Stability of Numerical Algorithms*, 2nd ed. Society for Industrial and Applied Mathematics, 2002.
- [12] S. Biderman, H. Schoelkopf, Q. G. Anthony, H. Bradley, K. O'Brien, E. Hallahan, M. A. Khan, S. Purohit, U. S. Prashanth, E. Raff, A. Skowron, L. Sutawika, and O. Van Der Wal, "Pythia: A suite for analyzing large language models across training and scaling," in *Proceedings of the 40th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 202, 2023, pp. 2397–2430.
- [13] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," OpenAI technical report, 2019. [Online]. Available: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf
- [14] S. Black, L. Gao, P. Wang, C. Leahy, and S. Biderman, "GPT-Neo: Large scale autoregressive language modeling with mesh-tensorflow," Zenodo, 2021. [Online]. Available: <https://doi.org/10.5281/zenodo.5297715>

- [15] S. Zhang, S. Roller, N. Goyal *et al.*, “OPT: Open pre-trained transformer language models,” arXiv:2205.01068, 2022.
- [16] BigScience Workshop, “BLOOM: A 176b-parameter open-access multilingual language model,” arXiv:2211.05100, 2022.
- [17] L. Ben Allal, A. Lozhkov, E. Bakouch *et al.*, “SmolLM2: When smol goes big—data-centric training of a fully open small language model,” in *Conference on Language Modeling*, 2025. [Online]. Available: <https://openreview.net/forum?id=3JiC12A14H>
- [18] A. Yang, A. Li, B. Yang *et al.*, “Qwen3 technical report,” arXiv:2505.09388, 2025.
- [19] T. Wolf, L. Debut, V. Sanh *et al.*, “Transformers: State-of-the-art natural language processing,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, 2020, pp. 38–45.
- [20] W. Kwon, Z. Li, S. Zhuang *et al.*, “Efficient memory management for large language model serving with PagedAttention,” in *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023, pp. 611–626.
- [21] L. Zheng, L. Yin, Z. Xie *et al.*, “SGLang: Efficient execution of structured language model programs,” in *Advances in Neural Information Processing Systems*, vol. 37, 2024.
- [22] ggml-org Contributors, “llama.cpp server,” 2026, commit 7f2dd88b0ac393357ae6a9e1992185a48c20b13b. [Online]. Available: <https://github.com/ggml-org/llama.cpp/blob/7f2dd88b0ac393357ae6a9e1992185a48c20b13b/tools/server/README.md>
- [23] A. Karpathy, “TinyLlamas: TinyStories-260K Checkpoint,” Hugging Face model repository, revision 0bd21da7698eaf29a0d7de3992de8a46ef624add, stories260K; accessed October 9, 2026. [Online]. Available: <https://huggingface.co/karpathy/tinyllamas>
- [24] H. He, “Defeating nondeterminism in LLM inference,” Thinking Machines Lab, September 2025. [Online]. Available: <https://thinkingmachines.ai/blog/defeating-nondeterminism-in-llm-inference/>
- [25] A. Wald, “Sequential tests of statistical hypotheses,” *The Annals of Mathematical Statistics*, vol. 16, no. 2, pp. 117–186, 1945.
- [26] S. R. Howard, A. Ramdas, J. McAuliffe, and J. Sekhon, “Time-uniform, nonparametric, nonasymptotic confidence sequences,” *The Annals of Statistics*, vol. 49, no. 2, pp. 1055–1080, 2021.
- [27] P. Blanchard, D. J. Higham, and N. J. Higham, “Accurately computing the log-sum-exp and softmax functions,” *IMA Journal of Numerical Analysis*, vol. 41, no. 4, pp. 2311–2330, 2021.
- [28] M. Jagielski, N. Carlini, D. Berthelot, A. Kurakin, and N. Papernot, “High accuracy and high fidelity extraction of neural networks,” in *29th USENIX Security Symposium*. USENIX Association, 2020, pp. 1345–1362.
- [29] N. Carlini, M. Jagielski, and I. Mironov, “Cryptanalytic extraction of neural network models,” in *Advances in Cryptology—CRYPTO 2020*, ser. Lecture Notes in Computer Science, vol. 12172. Springer, 2020, pp. 189–218.
- [30] Z. Yang, Z. Dai, R. Salakhutdinov, and W. W. Cohen, “Breaking the softmax bottleneck: A high-rank RNN language model,” in *International Conference on Learning Representations*, 2018.
- [31] M. Finlayson, X. Ren, and S. Swayamdipta, “Every language model has a forgery-resistant signature,” in *International Conference on Learning Representations*, 2026.
- [32] M. Andryscio, D. Kohlbrenner, K. Mowery, R. Jhala, S. Lerner, and H. Shacham, “On subnormal floating point and abnormal timing,” in *2015 IEEE Symposium on Security and Privacy*, 2015, pp. 623–639.
- [33] A. Schlögl, T. Kupek, and R. Böhme, “iNNformant: Boundary samples as telltale watermarks,” in *Proceedings of the 2021 ACM Workshop on Information Hiding and Multimedia Security*. ACM, 2021, pp. 81–86. [Online]. Available: <https://doi.org/10.1145/3437880.3460411>
- [34] C. Zhang, H. Foerster, R. D. Mullins, Y. Zhao, and I. Shumailov, “Hardware and software platform inference,” in *Proceedings of the 42nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 267, 2025, pp. 74 882–74 895.
- [35] H. Foerster, I. Shumailov, C. Zhang, Y. Zhao, J. Hayes, and R. Mullins, “Quantamination: Dynamic quantization leaks your data across the batch,” arXiv:2604.26505, 2026.
- [36] N. Loose, J. Sander, F. Mächtle, and T. Eisenbarth, “FloatDoor: Platform-triggered backdoors in LLMs,” arXiv:2606.19535, 2026.
- [37] N. S. Keskar, B. McCann, L. R. Varshney, C. Xiong, and R. Socher, “CTRL: A conditional transformer language model for controllable generation,” arXiv:1909.05858, 2019.
- [38] A. Yang, B. Xiao, B. Wang *et al.*, “Baichuan 2: Open large-scale language models,” arXiv:2309.10305, 2023.
- [39] vLLM Contributors, “vLLM V1 GPU sampling implementation,” 2026, commit cb4f016c5d02c9d483e6e3f1702d51da239b1a9b. [Online]. Available: <https://github.com/vllm-project/vllm/blob/cb4f016c5d02c9d483e6e3f1702d51da239b1a9b/vllm/v1/worker/gpu/sample/sampler.py>
- [40] D. Kalamkar, D. Mudigere, N. Mellempudi, D. Das, K. Banerjee, S. Avancha, D. T. Vooturi, N. Jammalamadaka, J. Huang, H. Yuen, J. Yang, J. Park, A. Heinecke, E. Georganas, S. Srinivasan, A. Kundu, M. Smelyanskiy, B. Kaul, and P. Dubey, “A study of BFLOAT16 for deep learning training,” arXiv:1905.12322, 2019.

Appendix A. Boundary Cases in Local Recovery

The recovery proof uses independently formed payloads. If a processor instead evaluates $RN_{32}(RN_{32}(q_i + B) + d)$, it can have a different rounding history from $RN_{32}(q_i + RN_{32}(B + d))$. The artifact checks the latter operation order and rejects payloads that are not exactly representable or exceed the allowed budget. This distinction matters most near the exponent boundary used by the probes.

For the tie-breaking priority test, $p = \max\{\ell, F + d_0 + \Delta\}$ satisfies $p \leq a$ under both hypotheses because $t > F$ and the scores lie on the Δ grid. The last pivot is at most $u + 2\Delta \leq P$. Its anchor bias is therefore nonnegative, and the first target bias is at most H_0 ; later target biases are smaller. When the true scores are equal, the original-anchor definition enforces anchor tie-breaking priority and the same test applies.

For residue identification, the two responses at $p_0, p_0 + \Delta$ are 00, 10, 11, 01 for residues 0, 1, 2, 3, apart from $k = 0$, which also produces 01. The query at $p_0 + 3\Delta$ produces one for $k = 0$ and zero for nonnegative residue-three candidates. If $p_0 < u$, then $p_0 \leq u - \Delta \leq P - 3\Delta$, so that extra pivot is valid. If $p_0 = u$, the algorithm returns before making it.

The lower-bound state count includes the unknown pairwise tie-breaking priority. For each anchor index i , the $i - 1$ strictly lower targets admit two priorities, while the equal target admits only the original-anchor priority. Summing $2(i - 1) + 1$ over $i = 1, \dots, K - 2$ gives $(K - 2)^2$. If only the score pair must be returned and tie-breaking priority is a nuisance variable, the number of distinct pairs is $(K - 2)(K - 1)/2$, giving a 45-query counting lower bound for binary32. The 46-query comparison in the main text therefore explicitly includes tie-breaking priority. Allowing nonbinary responses also changes the argument.

Appendix B.

A Bounded-Bias Nonidentifiability Region

Proposition 2 (An explicit BF16 nonidentifiability region).

Let q be a finite BF16 vector; let $M = \max_i q_i \in \{360, 362, \dots, 410\}$, and define $q'_i = \text{RN}_{\text{BF16}}(q_i - 2)$. Under (4), allowing arbitrary simultaneously biased tokens with every payload in $[-100, 100]$, the complete sets of maximizing indices are equal for q and q' for every query. They therefore have identical adaptive winner transcripts under the same fixed tie-breaking order.

Proof. Let $S = \{i : q_i \geq M - 200\}$. For $i \in S$, $160 \leq q_i \leq 410$ and subtracting two is exact in BF16, including across 256. Hence $q'_i = q_i - 2$. Every other score is at most $M - 201$, and its shifted-and-rounded value is at most $M - 203$. With biases bounded by 100, such a token remains strictly below a maximum-score token in both worlds and cannot win.

For $i \in S$, put $\beta_i = \text{RN}_{32}(b_i)$ and $x_i = q_i + \beta_i$, where the latter denotes the exact real sum. Its maximum x_* lies in $[M - 100, M + 100] \subset [260, 510]$. The translated maximum lies in $[258, 508]$. Both maxima and their rounding cells are strictly inside the binary32 binade $[256, 512]$, with spacing $\delta = 2^{-15}$. Translation by $-2 = -65536\delta$ shifts each relevant rounding cell by an even number of grid steps, preserving ties-to-even endpoint membership. Therefore

$$\begin{aligned} \text{RN}_{32}(x_i) &= \text{RN}_{32}(x_*) \\ \iff \text{RN}_{32}(x_i - 2) &= \text{RN}_{32}(x_* - 2). \end{aligned}$$

This equivalence also holds for x_i outside the maximum's binade because membership is in the translated maximum rounding cell. Thus the complete maximizing sets agree. The same tie-breaking order gives equal responses to every query; induction on the resulting histories gives identical adaptive transcripts. Rounding the most negative finite BF16 value after subtracting two remains finite, so q' is a valid score vector. $\square$

Appendix C.

Negative-Binade Construction

The sign of the scores affects which side of an exponent boundary has finer spacing. This distinction permits a separate construction for negative model scores; no common score shift is inserted into the service.

Let the addition format have binary precision $p \geq 3$, and let $S = 2^e$ be public. Set $\delta = S/2^{p-1}$, $K = \lfloor \min(H, S)/\delta \rfloor$, $H_0 = K\delta$, and $F = -S - H_0$. Assume $K \geq 2$, that the original greedy winner has score a , and that the selected target has score t , with

$$a \geq t > F, \quad a \in [\ell, u] \subset (F, -S - \delta].$$

The endpoints are on the δ lattice. Both scores therefore lie in the negative binade $(-2S, -S)$. The interface uses separately cast biases, a single correctly rounded addition, and an unknown fixed global tie-breaking order. All constructed biases are nonnegative and at most H_0 . The format

must represent $\delta/2$; we exclude overflow and flushing of the required subnormal operands.

The following proves Theorem 2 and its precision-dependent extension.

Proof. Write $d = a - t$. Adding $j\delta$ to the target gives a monotone threshold because all score sums near the comparison remain exactly on the original lattice. Let j be the first winning integer. Binary search costs at most $\lceil \log_2 K \rceil$ queries and leaves two hypotheses: $d = d_0 = (j - 1)\delta$ with anchor tie-breaking priority, or $d = d_0 + \delta$ with target tie-breaking priority. The search endpoints follow from the original winner and $t + H_0 > -S > a$.

Set $r = F + d_0 + \delta$ and submit biases

$$b_a = -S - r, \quad b_t = -S - r + d_0 + \delta/2.$$

These are legal: $b_t = H_0 - \delta/2$ and $0 \leq b_a \leq H_0$. Under anchor priority, $a - r = t - F - \delta \geq 0$, so the two exact sums are x and $x + \delta/2$, with $x = -S + (a - r) \geq -S$. Both are representable in the finer region, and the target wins. Under target priority, $a - r = t - F \geq \delta$, and the exact sums are x and $x - \delta/2$, both strictly in the finer region; the anchor wins. Thus one query determines the true gap and tie-breaking priority.

For a pivot r in the remaining legal range, first align the two exact sums to $x = -S + (a - r)$ using biases $-S - r$ and $-S - r + d$. Increase the bias of the lower-priority token by $\delta/2$. With $k = (a - r)/\delta$, its winning indicator is

$$g(k) = \begin{cases} 1, & k \geq 0, \\ 1, & k < 0 \text{ and } k \text{ odd}, \\ 0, & k < 0 \text{ and } k \text{ even}. \end{cases}$$

For $k \geq 0$, the half-step is exactly representable. For $k < 0$, it is a midpoint in the coarser binade; ties-to-even rounds upward precisely for odd k . Choose the initial residue-test pivot $r = u + \delta$. It satisfies $r > a$ and $r \leq -S$, so the response identifies the parity of $(a - r)/\delta$ without a boundary ambiguity. Restrict subsequent candidate pivots to the inferred parity class. Then k is even, and the response is exactly $1[a \geq r]$. Binary search on at most $\lceil N/2 \rceil$ candidates completes recovery.

Every parity-test or search pivot is at least $F + d + \delta$. Thus its target payload, even after the half-step increment, is at most $H_0 - \delta/2$; the other payload is nonnegative and no larger. All payloads are exact multiples of $\delta/2$ within $[0, S]$, hence representable. The anchor is never decreased, so no unmodified token can displace the original winner under the fixed global tie order. Each response is consequently binary.

Since $K \leq 2^{p-1}$ and $N \leq K - 1$, the upper bound is $2p - 1$. For the lower bound, take $H_0 = S$ and scores $a = F + i\delta$, $t = F + j\delta$ with $1 \leq j \leq i \leq K - 1$. There are $(K - 1)^2$ allowed score/priority triples. A binary decision tree requires at least $\lceil \log_2(K - 1)^2 \rceil = 2p - 2$ queries to identify all of them. This count concerns triple recovery, not score-only recovery or an oracle with more than two possible responses. $\square$

Other known rounding directions. The finer-region gap-disambiguation query is exact and does not depend on the rounding direction. The final comparison is adapted to the specified rounding direction. Under rounding toward negative infinity, or nearest with ties away from zero, the lower-priority half-step increment wins exactly when $a \geq r$. Under rounding toward positive infinity, or toward zero (all constructed sums are negative), instead decrease the higher-priority token's aligned bias by $\delta/2$. The lower-priority token then wins exactly when $a > r$. The latter probe is legal for search pivots $r \leq u \leq -S - \delta$. Ordinary binary search gives the same $2p - 1$ uniform bound. These are algorithms for a *known* rounding direction, not an identification theorem for an unknown runtime mode. Nearest-away is analyzed as an IEEE-defined direction; its support is not assumed for binary hardware.

Appendix D. Storage Precision and Control Resolution

Stored-score precision can also reduce the candidate set. Let s be the significand precision of a known score-storage format and let $p \geq s + 2$ be the bias and addition precision. The scores lie in $g\mathbb{Z}$, where $g = P2^{-s}$ below a positive boundary P and $g = S2^{1-s}$ below a negative boundary $-S$. The appropriate local range and representability conditions still apply. Storage precision and arithmetic precision are distinct interface properties [40].

Proposition 3 (Recovery with known score precision). *Let $B = P$ or $B = -S$ be the chosen boundary and g the corresponding storage spacing defined above. Set $H_0 = Kg \leq H$ and $F = B - H_0$, where K is an integer and $2 \leq K \leq 2^{s-1}$. Suppose $a, t \in g\mathbb{Z}$, $a \geq t > F$, and $a \in [\ell, u] \subset (F, B - g]$, with endpoints on the storage grid. Assume the required bias increments are representable and the arithmetic neither overflows nor flushes required subnormals. With the original anchor, unknown fixed tie-breaking priority, and round-to-nearest, ties-to-even addition, exact pair-and-priority recovery requires at most*

$$\lceil \log_2 K \rceil + 1 + \lceil \log_2 N \rceil \leq 2s - 1 \quad (19)$$

binary recovery queries, where N is the number of remaining anchor grid points after recovering the gap. BF16-to-binary32 and binary16-to-binary32 interfaces admit bounds of 15 and 21, respectively.

Proof. Search the target-bias threshold on the coarse storage grid. If the first winning bias is yg , the remaining hypotheses are $d = (j - 1)g$ with anchor tie-breaking priority and $d = yg$ with target tie-breaking priority. A single bias $(j - 1/2)g$ on the target gives score $a + g/2$ under the former and $a - g/2$ under the latter. Both are exactly representable in the arithmetic format, so this query identifies the gap and tie-breaking priority.

For a positive boundary P , let $\Delta = P/2^p$ be the arithmetic spacing below P . The storage spacing satisfies $g = 2^{p-s}\Delta$. Choose pivots on the storage grid, align the

scores at $P + (a - r)$, and increase the lower-priority token's bias by Δ . Since $(a - r)/\Delta$ is a multiple of four, the nearest-even response rule reduces directly to $1[a < r]$; no residue-identification queries are needed. For a negative boundary $-S$, let $\delta = S/2^{p-1}$. Align at $-S + (a - r)$ and use an increment $\delta/2$ instead. Here $(a - r)/\delta$ is even, so the winning indicator is $1[a \geq r]$. Each case yields ordinary binary search on the storage candidates.

After finding the gap, intersect the public interval with $[F + d + g, \text{boundary} - g]$. All search biases are nonnegative and bounded by H_0 , including the small increment, because g exceeds that increment. They are exactly representable in the finer arithmetic format. Monotonicity and the fixed global tie-breaking order exclude untouched tokens, as in the binary32 construction. The gap and anchor searches each cost at most $s - 1$ queries. $\square$

The lattice condition applies to scores entering bias addition; advertised weight or forward precision alone does not establish it. The proposition also requires biases to retain the finer arithmetic precision.

A finite-alphabet control. For $q = (30, 28)$ and $q' = q - \frac{1}{8}\mathbf{1}$, fine binary32 biases produce a winner difference. When biases are instead restricted to all finite BF16 values in $[-100, 100]$, and addition remains binary32, the two pair oracles are identical for both tie-breaking priorities. The same holds for the binary16 bias alphabet. We verified this finite statement by comparing the first-winning target-bias index for every anchor bias: 34,193 BF16 values and 44,161 binary16 values. Monotonicity extends the threshold comparison to every bias pair in each specified Cartesian alphabet for these two vectors. If both the bias and addition precision are lowered, format-scaled distinguishing probes exist again.

A sufficient condition for exact addition. Let g be a representable power of two and p the addition precision. Suppose every hidden score and accepted bias is a multiple of g , and $q' = q + c\mathbf{1}$ with $c \in g\mathbb{Z}$. If every possible biased sum in both vectors has magnitude at most $2^p g$, with no overflow, each sum is an exactly representable ng with $|n| \leq 2^p$. All additions are therefore exact. The biased vectors differ by $c\mathbf{1}$, so their maximizer sets and winners coincide under the same fixed tie-breaking order. Identical responses at each round give identical adaptive transcripts.

This representability condition identifies when control quantization removes the nonlinearity used by the channel. Coarse biases alone are insufficient: with binary32 scores $q = (2^{24}, 2^{24} - 2)$, $q' = q - 2\mathbf{1}$, and integer biases $(0, 3)$, the first vector yields equal rounded scores; the second has a strict target winner. Here the exact-lattice magnitude condition is violated.

Appendix E. Adaptive Selection Protocol

For Table 1, enumerate $B = \pm 2^e$, $e = -6, \dots, 10$, in ascending exponent order, positive before negative. Use each full local interval and require candidates to reproduce the

pair’s combined transcript. Set $\eta = |B|/2^{24}$ for $B > 0$ and $\eta = |B|/2^{23}$ otherwise. Probe each candidate with

$$b_0 = \text{RN}_{32}(B - \hat{a} + s\eta), \quad b_1 = \text{RN}_{32}(B - \hat{t} + (s + u)\eta).$$

Enumerate $s = 0, \pm 1, \pm 2, \pm 3, \pm 4$ and $u = 0, \pm 1/4, \pm 1/2, \pm 3/4, \pm 1$, negative before positive within each pair. Retain only legal payloads with $b_0 \geq 0$ and reject at the first contradiction. At most 81 checks per candidate precede the unique-candidate rule and eight-pair limit in Section 3.7.

The two-window controller uses $P \in \{16, 32\}$, $\eta = P/2^{24}$, $s \in \{0, -1, 1\}$, and $u \in \{-3/4, 3/4\}$. The cached ablations freeze predictions before scoring; the two BF16 controllers return identical selected pairs, scores, and priorities. Their full replay counts and input hashes are retained in the artifact.

Complete query costs. The 34-window generation controller uses 155,390 calls on 64 GPT-Neo inputs; its median/maximum is 2,245/3,927. On 32 BF16 OPT inputs, generic and BF16-aware recovery use 40,486 and 14,048 calls, with medians 1,184.5 and 413. Generic Qwen3 recovery uses 33,915 calls on 32 inputs, with median 994.5. Native Qwen3 totals are 15,312/15,417/14,768 for vLLM/SGLang/llama.cpp across 128 inputs each. Mean costs decompose into 2.2/2.2/2.2 selection calls, 111.4/112.2/107.0 window-search calls, and 6.0/6.0/6.2 checks. Repeated payloads and retries are included; calibration and independent replay are excluded.

Single-window ablation. With eight-pair selection, $H = 100$, and unchanged checks, $P = 16$ yields 64/64/60 Qwen3 recoveries across vLLM/SGLang/llama.cpp; $P = 32$ yields 37/36/36. SmolLM2 yields 1/64 and 64/64 per framework, respectively. Qwen3 on llama.cpp returns four incorrect candidates at $P = 16$ and three at $P = 32$; other failures are unresolved. All seven pass single-window checks outside the local domain but contradict the two-window transcript for the same pair and mask. All 163,242 unique queries match full-vocabulary replay.

Appendix F.

Recovery with Noisy Binary Responses

These results condition on a correctly isolated token pair, a fixed score vector in the public interval, and the original arithmetic assumptions. Let $v \in \{0, 1\}$ be the ideal answer to a logical query. The controller uses at most M such queries on its correct path. Repetitions count as separate token responses. An undecided test stops recovery without a score estimate; the error bounds below concern returning an incorrect estimate, not the probability of abstention.

F.1. Known Response-Error Bound

Assume each response, conditional on the complete preceding history, disagrees with v with probability at most $0 < \bar{p}_{\text{flip}} < 1/2$. Set

$$c = \frac{1 - \bar{p}_{\text{flip}}}{\bar{p}_{\text{flip}}}, \quad h = \left\lceil \frac{\log(M/\delta)}{\log c} \right\rceil. \quad (20)$$

For repeated responses $Y_j \in \{0, 1\}$, accumulate $S_n = \sum_{j=1}^n (2Y_j - 1)$. Return 1 at $S_n \geq h$ and 0 at $S_n \leq -h$. If neither boundary is reached by the response cap, abstain. This is a sequential likelihood-ratio rule [25] with a threshold chosen for a nonasymptotic, recovery-wide guarantee.

Proposition 4 (Error control for adaptive recovery). *Under these assumptions, substituting the sequential rule for every logical comparison gives probability at most δ of returning an incorrect score pair or tie-breaking priority.*

Proof. For $v = 0$, the process c^{S_n} is a nonnegative supermartingale starting at 1, since

$$\mathbb{E}[c^{2Y_j-1} \mid \mathcal{F}_{j-1}] \leq \bar{p}_{\text{flip}}c + (1 - \bar{p}_{\text{flip}})c^{-1} = 1.$$

The maximal inequality bounds the probability of reaching the wrong boundary by $c^{-h} \leq \delta/M$. For $v = 1$, apply the same argument to c^{-S_n} . Condition on the correct prefix of the adaptive controller and sum over the at most M possible first erroneous comparisons. If none occurs, the deterministic recovery proof applies; stopping at the cap produces no incorrect estimate. $\square$

For independent flips with actual probability $0 < p_{\text{flip}} < 1/2$, the uncapped test has error probability and expected response count

$$e_h(p_{\text{flip}}) = \frac{1}{1 + ((1 - p_{\text{flip}})/p_{\text{flip}})^h}, \quad (21)$$

$$\mathbb{E}[N] = \frac{h}{1 - 2p_{\text{flip}}} (1 - 2e_h(p_{\text{flip}})).$$

The absorbing random walk on $\{-h, \dots, h\}$ gives these expressions. At $M = 50$, $\delta = 0.01$, and $\bar{p}_{\text{flip}} = p_{\text{flip}} = 0.1$, $h = 4$ gives about five responses per comparison. Odd- r independent majority voting has error at most $M \exp[-2r(1/2 - \bar{p}_{\text{flip}})^2]$; $r = 27$ suffices, costing at most 1,350 responses.

F.2. Unknown Response-Error Rate

Suppose repetitions of a query are independent Bernoulli observations with fixed mean p . At sample sizes $n_j = 2^j$, use

$$\alpha_j = \frac{\delta}{Mj(j+1)}, \quad r_j = \sqrt{\frac{\log(2/\alpha_j)}{2n_j}}, \quad (22)$$

$$I_j = [\hat{p}_j - r_j, \hat{p}_j + r_j] \cap [0, 1].$$

Hoeffding’s inequality and $\sum_{j \geq 1} \alpha_j = \delta/M$ give simultaneous coverage at all scheduled looks. Return 1 only when $\inf I_j > 1/2$, or 0 only when $\sup I_j < 1/2$. This implements the time-uniform coverage principle [26] through an elementary allocation of error probabilities. If the majority direction equals v for each logical query, the same first-error argument bounds recovery error by δ . Our implementation uses $j = 1, \dots, 12$ and otherwise abstains. At a uniformly randomized token tie, $p = 1/2$ provides no majority direction; increasing the cap cannot justify a fixed-priority conclusion.

F.3. Random Token Tie Breaking and Changing Scores

Assume fixed scores, independent uniform selection between tied maximizers, exact strict comparisons, and no third-token winner or tie. Repeat each query r times, returning 1 only if all responses are 1, and returning 0 otherwise. This emulates token-0 priority, with exact strict comparisons and tie error probability 2^{-r} . Thus $r = \lceil \log_2(M/\delta) \rceil$ bounds recovery error by δ . For $M = 50$ and $\delta = 0.01$, $r = 13$ gives at most 650 responses. The recovered priority is the chosen emulation convention, not an intrinsic priority of the randomized service.

These guarantees do not follow from a marginal disagreement rate alone. An error shared by all repetitions can make every response support the same wrong direction. Nor is an unspecified changing score vector equivalent to response flips: if observations use $q + \xi$, the representation $(q + c, \xi - c)$ gives identical observations for any fixed vector c . Identifying a particular unperturbed score therefore requires a restriction on the perturbation or a stable execution condition. The controlled batch experiment in Section 5.4 tests the latter condition.

Appendix G.

Logit Anchors and Output-Head Extraction

Following the factorization argument of Carlini et al. [3], let $W \in \mathbb{R}^{V \times d}$ be a bias-free linear output head and $X \in \mathbb{R}^{d \times n}$ its hidden states. Complete relative logits give $D = CWX$, with C from (2). If CW and X have rank d , a rank factorization $D = \hat{B}\hat{X}$ satisfies $\hat{B} = CWG$ and $\hat{X} = G^{-1}X$ for an invertible G .

Write $\mu = \mathbf{1}^\top WX/V \in \mathbb{R}^{1 \times n}$ for the row of per-input mean logits. Each entry is determined by an original-logit anchor and its relative coordinates using (2). Then

$$\begin{aligned} u &= \mu \hat{X}^\top (\hat{X} \hat{X}^\top)^{-1}, \\ \hat{W} &= \hat{B} + \mathbf{1}u = WG. \end{aligned} \quad (23)$$

Since $\mu = (\mathbf{1}^\top WG/V)\hat{X}$ and $\hat{X}$ has full row rank, $u = \mathbf{1}^\top WG/V$. The identity $C + \mathbf{1}\mathbf{1}^\top/V = I$ then gives $\hat{W} = WG$. Anchors may vary by input: each determines its corresponding mean. If centering reduces rank, factor the completed matrix $WX = D + \mathbf{1}\mu$ instead.

These identities assume exact linear outputs; floating-point residuals, relative-coordinate errors, and conditioning affect their numerical accuracy. For held-out inputs, centered coordinates determine hidden coordinates and u predicts the mean without their anchors.

Token-only numerical reconstruction. For each non-winning token, bias bisection retains losing and winning endpoints, stopping at width 2^{-21} or adjacent binary32 values. The negative midpoint estimates its score relative to the unbiased winner. Rounding contributes additional error. If a token cannot be reached within the bias limit, we mark the input incomplete. The artifact includes per-coordinate rounding bounds and independent checks.

Numerical diagnostics and evaluation cost. Across input sets 1–3, the rank-64 spectral gaps are 3.09×10^4 , 1.07×10^5 , and 4.05×10^4 ; relative singular-value thresholds from 10^{-4} to 10^{-7} all retain rank 64. Effective condition numbers are 1,340, 441, and 1,048. Recomputing private Wh in FP64 gives RMSE 1.40×10^{-6} – 1.45×10^{-6} against the recorded FP32 logits, a numerical reference for the prediction errors. The held-out mean standard deviations are 0.831, 1.038, and 1.132; the constant-mean baseline uses only recovered extraction means. Held-out evaluation, including its independently scored anchors, adds 228,475/456,905/228,494 decisions and 7,564/15,858/7,500 HTTP requests. Those queries are excluded from the extraction costs in Table 4.

Appendix H.

Reproducibility Details

Paired bias-control experiment. The behavior-preservation study uses input indices 0, 4, 8, 12, 16, 21, 25, 29, 33, 37, 42, 46, 50, 54, 58, 63 from each serving configuration and PCG64 seed 2026101201. Random controls choose eight distinct tokens from the complete vocabulary or top 64, then draw independent biases uniformly from $[-A, A]$, for $A = 0.5, 2, 10$. Top tokens are ordered by decreasing original score, then increasing ID. For the two leading tokens A, T , set $d = q_A - q_T$ and $U = \text{nextafter}_{32}(m, +\infty) - m$, where $m = \max(|q_A|, |q_T|)$. The targeted controls set only $b_T = \text{RN}_{32}(d + kU/8)$, $k = -16, \dots, 16$. All 12,384 payloads are legal under $H = 100$ and distinct within each input. The two tie-breaking rules are fixed offline comparison conventions. Complete probabilities use CPU FP32 softmax, with L_1 differences accumulated in FP64.

Among threshold-adjacent controls, the original rounded top-two margin is zero in 834 cases, at most one ULP in 1,404 positive-margin cases, and between one and four ULPs in 930 cases. All 369 greedy changes occur in the zero-margin group. Random controls have 299 zero-margin cases and 8,917 cases above 16 ULPs; neither group changes.

Serving API configuration. The client sends HTTP requests to each server’s generation endpoint. vLLM: `/v1/completions`, `logit_bias`, `choices[0].token_ids`; SGLang: `/generate`, `sampling_params.logit_bias`, `output_ids`; llama.cpp: `/completion`, `logit_bias`, `tokens`. Each triple gives the endpoint, bias field, and returned IDs. Decimal token IDs map to exact binary32 bias values within $[-100, 100]$; probabilities are disabled. The artifact specifies payloads, launch flags, source hashes, and intervals.

Artifact availability. The artifact is not publicly available at the time of this preprint. We plan to release these materials in a future revision.