
Shrinkage, Peer Evidence, and Calibration for Benchmark Cold-Start Prediction: The AITrailblazer Entry to PAIEC-2026

Konstantin Vassilev
AITrailblazer

Abstract

We describe a submitted program for the Predictive AI Evaluation Competition (PAIEC-2026, Codabench competition 17828) that estimates $P(\text{response} = 1)$ for a (subject, item) pair without running the subject, under nested label budgets 0/1/3/7/15/31 and benchmark-level cold start. The selected submission (v2, Codabench ID 1016554) is a fitted two-parameter logistic (2PL) item-response model with a metadata prior for unseen subjects, Platt calibration, a bounded MAP update from revealed labels, and a shrinkage blend toward the subject’s own revealed mean at weight $n/(n + 1)$. It finished the Codabench formative evaluation with a Brier ALC of 0.183081, the only finished Codabench score in this entry. Our main methodological contribution, the v3 method, keeps the v2 predictor and changes how revealed labels are consumed: the revealed mean enters at a fitted pseudo-count weight $n/(n + \kappa)$ with $\kappa = 2$, and a peer-evidence term shifts the prediction by the mean residual of other subjects’ labels on the same item, with an exact zero fallback when no peer label exists. On two independent internal benchmark-holdout splits, scored with a platform-faithful joint-pool harness, v3 improves the weighted Brier over v2 by 0.00595 (paired 95% CI [0.0038, 0.0081], S0 split) and by 0.00926 (CI [0.0071, 0.0114], S1 confirmation split). These gains are internal validation results only: every v3-family submission filed to Codabench failed on the organizers’ backend (five at the scoring stage, one at the prediction stage), for reasons we analyze in Section 6; no package-side cause was established, and the evidence best supports a backend failure independent of the package. The selected submission is therefore v2.

1 Introduction

PAIEC asks for a program, not a set of predictions. For each sampled (subject, benchmark) pair, the program receives items one at a time, may acquire at most 31 labels along one nested trajectory (budgets 0, 1, 3, 7, 15, 31), and must output a probability that the subject’s response to each evaluation item equals 1, without running the subject. Final evaluation uses benchmarks held out at the benchmark level, so the program must predict on benchmark families it has never seen, for subjects identified only by metadata and by whatever labels the acquisition policy has revealed. Scoring is the Brier score averaged over budgets with weights 0.1/0.2/0.2/0.2/0.2/0.1 (the Brier ALC), computed per pair and averaged over pairs.

Three reference behaviors frame the task: a constant 0.5 predictor, the organizers’ Empirical Mean baseline (the subject’s revealed mean on the benchmark, unsmoothed), and BLE, an LLM-based baseline that fits no parameters to response data. Our design follows the organizers’ canonical family for this task, amortized latent-trait models in the style of prediction-powered evaluation [1]:

a fitted item-response model whose parameters for unseen subjects and unseen benchmarks come from priors rather than from interaction data.

The entry was built under two self-imposed disciplines that shape this report. First, an anti-tuning rule: formative (development) feedback from Codabench was never used to iterate on the method; all method decisions were taken on internal holdout splits, with hyperparameters fitted on calibration halves and ship decisions read on validation halves. Second, exact fallback: every added evidence term is exactly zero when its evidence is absent, so each version degrades to the previous one rather than to a new failure mode.

Section 2 describes the task data and the fitted state. Section 3 gives the v2 predictor that is the selected submission, and then the v3 method. Section 4 reports internal validation, including the negative results that decided the acquisition policy and the calibration configuration. Section 5 is the Codabench record. Section 6 analyzes the v3-family backend failures factually, including what is not known. Sections 7 and 8 state limitations and the reproducibility and disclosure record.

2 Task and data

Interface. The submission is a ZIP with `model.py` exposing `predict(input, labeled) → a float in [0, 1]`, `labeling.py` exposing the acquisition function, an empty `models.txt`, and the fitted state under `any_other_files/`. The `labeled` argument is a single global list spanning the sampled pairs, not filtered to the queried subject; filtering and pooling are the program’s job. Candidates for acquisition arrive one at a time in fixed order and a skip is permanent. Formative feedback during the development phase is a sampled subset of the hidden set capped at 1,000 unique (subject, item) pairs, redrawn per submission; its sampling error on the ALC is on the order of 0.02, which is why it cannot steer method decisions.

Training data. Training was restricted to the competition benchmark inventory (the measurement-db dataset), per the rules. The verified pull on disk comprises 274 non-empty benchmark directories, 18,455,854,265 bytes, with a uniform four-table schema (benchmarks, items, subjects, responses). The inventory contains 21,603 subject rows keyed to 2,061 distinct subject keys (normalized model name, provider, access date, harness, harness version, reasoning effort), 8,222,569 items, and 91,080,126 response rows, of which 6,205,449 (6.8%) carry a null response. Aggregating responses to subject×item cells (cell mean outcome, trial-count weight) gives 43,432,768 cells.

Outcome policy. Outcomes are not uniformly binary: 167 benchmarks are binary, 45 are fractional on $[0, 1]$ (partial credit), 23 carry bounded integer rubric scores, 29 carry raw metric values on unbounded scales that are not correctness probabilities, and 10 have entirely null outcome columns. We fit binary and fractional benchmarks as-is under a fractional-Bernoulli likelihood, min-max normalize the bounded rubric benchmarks by their observed extremes, and exclude the 29 raw-scale and 10 empty benchmarks, each with its reason logged. Of 235 fitted candidates, the 12 validation holdout benchmarks were removed before fitting, so the shipped fitted state covers 223 benchmarks.

Fitted state. The state file (143,661,229 bytes, sha256 `a1a1396c...e03de4b0`) contains 1,421,086 item parameter entries, 20,630 subject entries, 2,009 metadata-prior keys, and 39,223 flagged items (fitted discrimination near zero or negative; downweighted in fitting and barred from acquisition candidacy). The global response rate in the fitted data is 0.370 and the default ability is -0.863 .

3 Method

3.1 The v2 predictor (selected submission)

Item-response core. For subject i and item j ,

$$P(y_{ij} = 1) = \sigma(a_j(\theta_i - b_j)), \quad (1)$$

with item discrimination a_j and difficulty b_j , subject ability θ_i , and a Rasch fallback ($a_j = 1$) where discrimination is unidentified. Parameters are fitted offline per benchmark by alternating MAP estimation under the fractional-Bernoulli likelihood over aggregated cells, in two passes; the second pass downweights flagged items ($a_j < 0.35$) to 0.25 weight.

Cold start. An unseen subject’s ability prior comes from a metadata table keyed by normalized model name, provider, and harness family (inferred model family, size class, and release date from public model information, shipped as fitted state). An unseen benchmark’s items take pooled parameters, with content- and length-based features as the prior’s carriers. Predictions are clipped to $[0.02, 0.98]$ at every stage; no output ever leaves that interval.

Online update and calibration. Revealed labels update θ_i by a bounded MAP step anchored at the fitted prior (prior variance 1.0). The raw model probability then passes a Platt calibration layer ($A = 0.3257$, $B = -0.3784$, fitted on held-out budget-0 predictions).

The revealed-mean blend (the v1→v2 repair). In v1, revealed labels reached the prediction only through the bounded MAP update, with two consequences measured in Section 4: at budget 31 the model’s Brier (0.1721) lost to the plain Empirical Mean (0.1460), and its calibration error grew with budget (ECE 0.1195 at budget 31 vs. 0.0201 for the replica). The v2 repair blends the calibrated probability p_{cal} with the subject’s own revealed fractional mean m on that benchmark:

$$p = (1 - w)p_{\text{cal}} + wm, \quad w = \frac{n}{n + 1}, \quad (2)$$

where n is the number of revealed labels for the subject on the benchmark. At $n = 0$ the prediction is bit-identical to v1. This single change is the whole v1→v2 method difference; the fitted state and the acquisition policy are unchanged.

Acquisition. The shipped policy is paced random: `labeling.py` accepts a candidate when a per-step seeded hash-uniform falls below `labels_remaining/items_remaining`, spending the budget exactly with forced takes at the tail. It consults no prediction and no state. Section 4 records why: a Fisher-information selection policy was measured on the real-data holdout and scored *below* random reveals, and a spanning variant scored below both; the policy that ships is the best of the configurations measured, per the build specification’s rule that acquisition ships on measured ALC, not on theory.

3.2 The v3 method (the contribution)

The v3 method keeps the v2 predictor, state, calibration, and acquisition, and changes only how the revealed labels in the global `labeled` list are consumed. It has two terms.

(i) Fitted pseudo-count blend. The blend weight becomes

$$w = \frac{n}{n + \kappa}, \quad \kappa = 2.0, \quad (3)$$

with κ fitted on calibration halves of the internal splits (Section 4). The model prior thus counts as two pseudo-observations rather than one.

(ii) Peer-label evidence. Because `labeled` is global, another sampled subject may have a revealed label on the very item being predicted. For each peer subject q with a label $y_{q,\text{item}}$ on the query item and $n_q \geq 2$ labels on the query benchmark, let r_q^{LOO} be the peer’s leave-one-out rate on that benchmark (its benchmark mean with the query item’s label removed). The prediction is shifted by

$$\Delta = \lambda \cdot \frac{m}{m + 1} \cdot \text{mean}_q \left(y_{q,\text{item}} - r_q^{\text{LOO}} \right), \quad \lambda = 0.75, \quad (4)$$

where m is the number of peer labels on the query item, and the shifted probability is clamped to $[0.02, 0.98]$. If no peer has a label on the item, Δ is exactly zero and v3 reduces to the κ -blend. The term prices an item’s idiosyncratic difficulty relative to each peer’s own level: a peer who is strong on the benchmark but wrong on this item is evidence the item is harder than its parameters say, and symmetrically for the reverse.

What v3 is not. Per-budget temperature scaling was fitted on the calibration halves (S0 temperatures 0.8841, 0.928, 1.0645, 1.0495, 1.0511, 0.9407 for budgets 0–31) and *rejected*: the temperature variant’s overall weighted Brier was statistically indistinguishable from the no-temperature variant

Table 1: Brier per budget on the SPEC §6 benchmark-cold-start split (12 held-out benchmarks, 363 pairs, own-acquisition arm). Lower is better. The constant-0.5 reference has weighted Brier 0.23546.

Predictor	B0	B1	B3	B7	B15	B31	Weighted
Empirical Mean (replica)	0.2355	0.2671	0.1823	0.1602	0.1509	0.1460	0.19024
v1 (shipped configuration)	0.2102	0.2028	0.1939	0.1848	0.1776	0.1721	0.19006
v2 (repair, = v1 + blend)	0.2102	0.1998	0.1733	0.1575	0.1502	0.1458	0.17177

while its budget-0 error degraded out of sample (Section 4). The shipped v3 configuration is therefore $\kappa = 2.0$, $\lambda = 0.75$, temperatures off, calibration $A = 0.3257$, $B = -0.3784$, MAP update on. The submitted program is pure arithmetic at predict time: Python standard library only, no network, no LLM calls.

4 Internal validation

Protocol. Two generations of internal evaluation were used, and their numbers are not interchangeable. The v1/v2 generation (SPEC §6 split) holds out 12 benchmarks, scores up to 40 pairs per benchmark (363 pairs scored) with a seeded 50/50 acquisition/evaluation split per pair and nested budgets, each pair’s labeled list containing only its own reveals. The v3 generation uses a platform-faithful joint-pool harness: a pair’s labeled list at budget b is its own first b reveals plus every other sampled pair’s first b reveals on the same benchmark, which is the pool the platform actually presents and the only way peer evidence can be measured. Hyperparameters (κ , λ , temperatures) were fitted only on calibration halves; all v3 numbers below are validation halves that the fitting never saw. Paired bootstrap confidence intervals are over pairs. Because absolute Brier levels differ sharply between benchmark populations (the same v2 scores 0.167 on S0 and 0.222 on S1), only within-split paired differences are evidence; no cross-split level comparison is made anywhere in this report.

v1 and v2 on the SPEC §6 split. Table 1 gives Brier per budget. v1 was at parity with the Empirical Mean replica (weighted Brier 0.19006 vs. 0.19024), winning budget 0 (0.2102 vs. 0.2355) and losing budget 31. The single pre-declared repair (the $n/(n+1)$ blend) moved the shipped configuration to weighted Brier 0.17177 (ALC skill 0.07823 against the replica’s 0.05976), with budget-31 Brier 0.1458 against the replica’s 0.1460 (a margin of 0.0002, inside this split’s noise band, and reported as such) and budget-31 ECE improved from 0.1195 to 0.0096, better than the replica’s 0.0201. The win is carried by the middle budgets and by calibration, not by the budget-31 leg.

Acquisition measurement. On the same split, Fisher selection scored ALC skill 0.05975 (weighted Brier 0.19025), a hair below random reveals (0.05978 / 0.19022); a spanning-plus-random variant scored 0.05945 / 0.19055, below both. Paced random, measured end-to-end, was the best configuration (0.05994 / 0.19006 in the v1 generation). Fisher selection was retired on this evidence.

v3 on the joint-pool harness. Two splits were run. S0 uses the original 12 holdout benchmarks and the fitted state trained without them; S1 is a confirmation split with its own independently fitted state and calibration and a disjoint set of 12 holdout benchmarks (sib200, ultrafeedback, math-vista_mini, hallucinationbench, haiid, livebench, rewardbench, truthfulqa_mc, mmlu, realtimeqa, babylong, ai2d_test). On both splits, the calibration-half grid chose $\lambda = 0.75$ at every κ ; S0’s fit chose $\kappa = 2.0$ and S1’s independent fit chose $\kappa = 3.0$ with the same λ , corroborating the peer term’s strength while leaving κ ’s exact value split-dependent. The ship configuration ($\kappa = 2.0$, $\lambda = 0.75$, temperatures off) gives, on validation halves:

Per-budget, the v3 gain concentrates where peer evidence exists and labels are still few relative to the pool: on S0, budget-31 Brier improves from 0.13552 (v2) to 0.12255 (v3) and budget-1 from 0.19338 to 0.18749, with budget 0 identical by construction (0.21197); on S1 the same pattern holds (0.19687 $\rightarrow$ 0.18972 at budget 31). Calibration also improves at low budgets: budget-1 ECE falls from 0.0805 (v2) to 0.0161 (v3) on S0 and from 0.1465 to 0.0248 on S1. The fitted-temperature variant scored 0.16104 (S0) against the no-temperature 0.16100 overall, but its budget-0 Brier degraded from 0.21197 to 0.21240 and its budget-0 ECE from 0.0017 to 0.0209; since budget

Table 2: Joint-pool harness, validation halves. Weighted Brier (Brier ALC) per line; the improvement is the paired difference v3 minus v2 with a pairs-bootstrap 95% CI. S0: 183 pairs in the bootstrap; S1: 240 pairs.

Split	Line	Weighted Brier	Improvement	95% CI
S0	v2	0.16695	—	—
S0	v3 (ship config)	0.16100	0.00595	[0.0038, 0.0081]
S1	v2	0.22164	—	—
S1	v3 (ship config)	0.21238	0.00926	[0.0071, 0.0114]

0 is the one budget at which no evidence term can help, we rejected temperatures and shipped the no-temperature configuration.

5 Codabench record and the selected submission

Eight submissions were filed to competition 17828. Their platform record, read from the submissions board on 2026-10-29, is:

Table 3: Codabench submissions, team AITrailblazer, competition 17828.

ID	Package	Filed	Platform outcome
1016508	v1	Oct 28	Running (still Running at the Oct 29 read, 30+ h)
1016554	v2	Oct 28	Finished; formative Brier ALC 0.183081
1016971	v3	Oct 28	Failed, scoring stage
1017436	v3 (byte-identical)	Oct 28	Failed, scoring stage
1018409	v3.1	Oct 29	Failed, prediction stage
1019171	v3.2	Oct 29	Failed, scoring stage
1019486	v3.3	Oct 29	Failed, scoring stage
1019611	v3.3 (byte-identical retry)	Oct 29	Failed, scoring stage

Selected submission: v2, Codabench ID 1016554 (formative Brier ALC 0.183081). It is the only submission that finished, and under the anti-tuning rule the selection was not made by comparing formative draws.

The v1 submission’s state is reported as observed: it was filed on October 28 and the board still showed it as Running at the last read on October 29, more than 30 hours later. We offer no explanation for that state; it is recorded because it bears on Section 6.

6 Failure analysis: the v3-family backend failures

Six of the eight submissions, all carrying the v3 method, failed on the organizers’ backend. The facts, from the platform’s status fields and sanitized logs, are: submissions 1016971, 1017436, 1019171, 1019486, and 1019611 died at the *scoring* stage with the status “Scoring failed on the gcp backend”; their logs show ingestion completing (“Prediction completed”) and complete formative tables in the scoring output, then stop, with no error line and no persisted score. For submission 1019486 the complete scoring output was retrieved; our computation from that log gives a pair-weighted aggregate of 0.206207 over 484 subject-item pairs and 8 subjects, with the printed per-subject values reproducing exactly from the per-budget tables. That figure is our computation from the platform’s log; the platform persisted no score for the submission, and it is not a score of record. Submission 1018409 (v3.1) died earlier, at the *prediction* stage. We report what our local analysis established, what it identified as a hypothesis, and what it could not establish.

The prediction-stage failure (v3.1) is explained. Between v3 and v3.1, the only change was integrity hardening of the labeled-pool index cache: the cache key gained a per-call content checksum of the labeled list. That made every prediction call pay a full pass over the pool. In local timing on a pooled labeled list, the v3.1 build’s steady-state cost was 12.75 ms per call against the shipped v3 build’s 0.05 ms per call on the same list, and the cost grows with the pool. A per-call cost of that

order, at platform pool sizes and call counts, is sufficient to explain a prediction-stage timeout. The checksum was removed in v3.2.

No correctness defect was found in the scoring-failed packages. A local differential battery (about 1,600 calls per model across budget, matching, mutation, malformed-input, concurrency, and shuffling cases, plus a 300-call randomized fuzz per model) compared v3.2 against the v2 package that finished. Findings: v3.2 never raised; every output was a Python `float`, finite, and inside $[0.02, 0.98]$; outputs were exactly order-stable under labeled list shuffling ($\max |\Delta| = 0.0$); and its state file is deep-equal to v2's except for the added method block $\{\kappa : 2.0, \lambda : 0.75, \text{temperatures} : \{\}, \text{use_map} : \text{true}\}$. Row-count, NaN/inf, output-type, and ordering explanations for a scoring failure are therefore ruled out locally. Two semantics divergences from v2 were found and are recorded for completeness (subject matching by key fields rather than exact dict equality; exact-benchmark pooling rather than benchmark-plus-empty pooling); both are correctness-neutral on the platform's data shapes as far as local tests can show.

A candidate mechanism for the scoring-stage failures is identified, and labeled a hypothesis. The v3-family packages index the labeled pool once and cache the index across calls, keyed by the list's identity and length. That is $O(1)$ per call (about 0.07 ms at any pool size) *if* the runner reuses one list object. The organizers' published streaming client deep-copies the labeled list on every call in its acquire path (`streaming_ingestion.py`, line 169); wherever that convention holds, the cache misses on every call and the index rebuilds fully, at about 30 μs per entry per call against v2's about 1 μs per entry per call. Measured at a 16,000-entry pool with a fresh list object per call: 479.5 ms per call (v3.2) against 16.5 ms per call (v2). The cost peaks exactly where a scoring-side re-run or the final checkpoint would operate, i.e. after earlier formative tables already exist, which matches the observed failure point. This was, at that point, the only tested property present in all three scoring-failed packages, absent from v2, and scaling so as to kill a late stage rather than an early one. The v3.3 outcome reported below removes it as a candidate for the later failures. It is not established as the cause: the scoring program is not public, ingestion evidently tolerated the cost, and a backend failure independent of the package cannot be excluded, not least because v1's run has sat in Running for over 30 hours on the same backend.

The v3.3 transplant. Following the diagnosis, a v3.3 package was built by transplanting the v3 method into the v2 code tree verbatim (v2's per-call scan idiom, no cache, no cross-call state), with the state file byte-identical to v3.2's. Local verification: outputs match v3.2 exactly on 4,000 side-by-side predictions; the harness anchors reproduce (S0 weighted Brier 0.161002473942, S1 0.212376663150); fresh-copy timing at a 16,000-entry pool is $1.44\times$ v2's per-call cost, against v3.2's $25\times$ -plus; and the mutation probe (an in-place, same-length edit of the labeled list) is tracked correctly, where the cached builds served a stale index. v3.3 was filed twice: submission 1019486 on October 29, 2026, and a byte-identical retry, submission 1019611, the same day. Both died at the scoring stage with the same status text. v3.3 carries no cache and no cross-call state, so the cache property identified above is absent from it, yet the failure recurred at the same step. That narrows the differential: no tested computational property now separates the five scoring-failed submissions from the v2 package that finished, and a backend failure independent of the package is the explanation consistent with all eight outcomes, including v1's run still shown as Running after more than 30 hours. We state that as the best-supported reading, not as proof; the scoring program is not public, and only the organizers can confirm it.

7 Limitations

- **The v3 method has no finished Codabench score.** Its evidence is two internal splits and, for v3.3, the platform's own scoring output retrieved from the log of submission 1019486 (Section 6), which the platform did not persist as a score. Internal absolute levels and the formative level are not comparable across populations, and the formative sample is too small (standard error on the order of 0.02) to confirm or refute a 0.006–0.009 effect.
- **Benchmark cold start is the hard direction.** Item-side parameters transfer weakly to unseen benchmark families; our own two splits put the same predictor at 0.167 and 0.222, and the improvement intervals, while excluding zero on both, are paired within-split statements only.

- **The Empirical Mean is a strong high-budget opponent.** v2’s budget-31 margin over it is 0.0002 on the SPEC split, inside noise; the entry’s edge is concentrated at budget 0 and the middle budgets, which is also where 0.6 of the ALC weight sits (budgets 1/3/7 at 0.2 each) — a favorable accident of the weighting, not a guarantee on the hidden set.
- **Peer evidence depends on the platform’s sampling.** The peer term fires only when a co-sampled subject has revealed a label on the query item; in sparse formative draws only part of the sample shares a benchmark, so realized gains may run below the internal estimates, which used dense joint pools.
- **Excluded benchmark types.** The fitted state excludes the 29 raw-scale and 10 all-null benchmarks; for hidden benchmarks of those types the program relies on priors and revealed labels alone.
- **The scoring-stage failure is unexplained.** Section 6’s mechanism is a hypothesis with local evidence, not a finding about the organizers’ backend; the failure could be independent of our packages.
- **Stale-index trade in the filed v3 packages.** The cached index in v3/v3.2 serves stale results after an in-place, same-length mutation of the labeled list. This was a documented, accepted trade when filed; v3.3 removes it by construction.

8 Reproducibility and disclosures

Rerunning the selected submission. Unzip the v2 package (sha256 below), confirm it with the organizers’ `check_submission_zip.py` in full mode (it passes), and run it under the organizers’ streaming ingestion client; the program is deterministic (fixed seeds; identical inputs give identical outputs), loads its state in seconds, and requires only the Python standard library. The internal harnesses used for Section 4 (`validate_real.py` for the SPEC split, `validate_v3.py` for the joint-pool splits) reproduce every internal number in this report from the fitted states and holdout lists named there.

Package record. All packages share the four-entry layout (`model.py`, `labeling.py`, `models.txt`, `any_other_files/state.json`); `labeling.py` and `models.txt` are byte-identical across v2 and the v3 family, and the v3-family state differs from v2’s only by the method block.

Table 4: Submission packages (ZIP size in bytes and sha256).

Package	Bytes	sha256 (first 16 hex)
v1	48,215,623	46af2db4ab35697f
v2 (selected)	48,216,399	fc3488e38e1f982d
v3	48,216,699	688ac8c85dde2076
v3.1	48,217,448	079028ce51e766dc
v3.2	48,290,868	4c4fe855d145a880e
v3.3 (not filed at writing)	48,292,173	0ecb5e212988dad8

Training-data disclosure. The only training source was the competition benchmark inventory (measurement-db), restricted per the competition rules; no other dataset was used for fitting, development, or curation in any version. The subject metadata prior is built from public model information (model family, size class, release date) and is shipped inside the fitted state. Item parameters for benchmarks in the inventory were fitted from the inventory’s own response tables.

LLM coding-assistant disclosure. This system was designed and implemented with extensive use of LLM coding assistants (Muse / Muse Spark class models) under human direction, including code generation, validation harnesses, and diagnostic tooling. The submitted program itself makes no LLM calls, no network calls, and no runtime downloads; at predict time it is arithmetic over its fitted state and the revealed labels.

Feedback-use disclosure. Codabench formative feedback was used only to confirm that submissions ran and to record their states. It was not used to select hyperparameters, choose between method variants, or time submissions; all such decisions predate or ignore the formative draws, per the anti-tuning rule stated in Section 1.

References

- [1] V. Truong, F. Tu, P. Liang, B. Li, S. Koyejo. Reliable and Efficient Amortized Model-based Evaluation. *Proc. ICML*, 2025. (The organizers’ canonical amortized latent-trait method for this task family.)
- [2] J. Platt. Probabilistic Outputs for Support Vector Machines and Comparisons to Regularized Likelihood Methods. In *Advances in Large Margin Classifiers*, 1999.
- [3] C. Guo, G. Pleiss, Y. Sun, K. Q. Weinberg. On Calibration of Modern Neural Networks. *Proc. ICML*, 2017. (Temperature scaling; the fitted temperatures of Section 3.2 were evaluated against this alternative and rejected on out-of-sample budget-0 error.)
- [4] A. Niculescu-Mizil, R. Caruana. Predicting Good Probabilities with Supervised Learning. *Proc. ICML*, 2005. (Platt scaling preferred over isotonic regression at small calibration-set sizes.)