AURA: ADAPTIVE UNCERTAINTY-AWARE REFINEMENT FOR LLM-AS-A-JUDGE AUDITING

Anonymous authors
Paper under double-blind review

ABSTRACT

Large language models (LLMs) are widely used as judges for open-ended generation because human evaluation is costly and hard to scale, yet their preferences remain imperfect proxies for human judgment. Human verification can reveal which judge decisions to trust, but it is typically affordable for only a small fraction of them. We propose AURA, an adaptive, uncertainty-aware framework that jointly refines estimates of judge–human agreement and guides human verification for pairwise LLM-as-a-judge decisions. The framework updates a probability of agreement for each comparison using a learned human-consistency scorer and local and anchor-based evidence. Sparse transport propagates reliable signals to uncertain comparisons, while uncertainty guides the allocation of additional human verification. Our synthetic and real-data evaluations show how performance varies with judge quality and verification budget, and illustrate when adaptive refinement can help allocate limited human review.

1 INTRODUCTION

Evaluating open-ended language-model answers increasingly depends on whether an evaluator can reliably distinguish among plausible answers that differ in factual accuracy, reasoning quality, completeness, or safety (Liu et al., 2023; Kim et al., 2024). Human evaluation remains the most direct way to capture these differences, but it is expensive, slow, and difficult to scale as models, prompts, and evaluation criteria evolve. As a result, *LLM-as-a-judge* evaluation, in which a strong model grades or compares the outputs of other models, has become a standard component of modern model development (Zheng et al., 2023; Dubois et al., 2023).

The practical value of LLM judges, however, rests on an assumption that often fails: that their preferences track human preferences. LLM judges are sensitive to response position, length, and formatting, and can favor their own outputs (Zheng et al., 2023; Dubois et al., 2024; Zeng et al., 2024; Thakur et al., 2025). These biases are especially damaging in pairwise evaluation: each judgment is binary, so a single error fully reverses which response is preferred, and such errors can propagate into the model rankings or preference data built on top of them (Wang et al., 2024a). The central challenge is therefore to audit judge decisions, determining which to trust, which to correct, and which to send for human verification, while keeping human effort to a minimum.

The direct approach is to train a classifier on human-verified comparisons to predict judge–human agreement, but the required labels can undermine the cost savings of LLM judges. In constrast, Positive-Unlabeled (PU) learning (Elkan & Noto, 2008; du Plessis et al., 2014; Kiryo et al., 2017; Bekker & Davis, 2020) offers a more label-efficient alternative: judge decisions confirmed to agree with humans provide trusted positives, while the remaining unverified decisions form an unlabeled mixture of correct and incorrect ones. Standard PU methods, however, assume that the positive set is known in advance and that the unlabeled pool stays fixed. This setup has two limitations in judge auditing. First, the initial positives often come from weak heuristics or a small, non-randomly selected set of human labels, so they may share the judge's biases, and fixing them lets those biases persist. Second, the unlabeled pool does not stay fixed: human review keeps moving comparisons out of it, revealing disagreements and agreements, and deciding which comparisons to review next becomes part of the problem itself.

To address both failures, we propose AURA, an adaptive and uncertainty-aware framework for auditing pairwise LLM-as-a-judge decisions under a limited human-verification budget. Its key idea is

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

to treat trust in each judge decision as a latent quantity that is refined as evidence accumulates, instead of fixed in advance. Starting from a small set of human-verified comparisons, AURA estimates the probability that each remaining judge decision agrees with human judgment, and iteratively refines these estimates by combining a learned human-consistency scorer with evidence propagated from verified comparisons to nearby uncertain ones. At each round, it sends the most uncertain comparisons for human review and feeds the returned labels, whether they confirm or overturn the judge, back into refinement. As a result, the trusted and uncertain sets evolve throughout auditing.

Our contributions are as follows:

- We formulate LLM-as-a-judge auditing as adaptive PU refinement under a limited human-verification budget, in which the trusted and uncertain sets are updated as verification proceeds rather than fixed in advance.
- The trust and transport updates are closed-form, requiring no iterative optimization beyond the one-time encoder fit, so the method adds negligible overhead beyond representation learning.
- We show that the trust and encoder updates are stable within each fixed-configuration refinement round, and derive an exact error-decomposition theorem that characterizes when transport reduces human-consistency estimation error.
- We evaluate AURA on synthetic data and on real judgments from seven judge models on MT-Bench and Chatbot Arena. It matches or exceeds fixed-budget supervised baselines while using an adaptively-realized verification budget that is substantially smaller than theirs (Section 6.2).

2 Background and Related Work

LLM-as-a-judge evaluation has become a common approach for assessing open-ended generations, especially when task-specific reference answers are unavailable. MT-Bench and Chatbot Arena popularized pairwise LLM and human preference evaluation as a scalable framework for comparing chat assistants (Zheng et al., 2023; Chiang et al., 2024). Subsequent work has proposed stronger evaluator models, prompting schemes, and benchmark protocols (Liu et al., 2023; Wang et al., 2024b; Kim et al., 2024; Zhu et al., 2025; Dubois et al., 2024). At the same time, LLM judges can exhibit position bias, verbosity bias, prompt sensitivity, score instability, and weaker alignment with humans on difficult examples (Wang et al., 2024a; Zeng et al., 2024; Thakur et al., 2025; Tan et al., 2025; Park et al., 2024). Rather than introducing another standalone judge, AURA asks when the output of an existing judge should be trusted.

Our work is also related to preference modeling and reward-model evaluation. Pairwise comparison has a long statistical history, including the Bradley–Terry model (Bradley & Terry, 1952), and modern alignment pipelines use preference data to train reward models or optimize policies through RLHF and DPO (Christiano et al., 2017; Ouyang et al., 2022; Bai et al., 2022; Rafailov et al., 2023). RewardBench evaluates reward models using prompt–chosen–rejected triples across chat, reasoning, and safety settings (Lambert et al., 2025). These works primarily study how to train or benchmark models using preference data. AURA instead studies a complementary auditing problem: given an LLM judge's pairwise preference, estimate whether it agrees with human judgment and decide whether to trust, correct, or verify it.

Methodologically, AURA connects to PU learning, weak supervision, graph-based semi-supervised learning, transport, and active verification. Standard PU learning assumes labeled positives and an unlabeled mixture (Elkan & Noto, 2008; Bekker & Davis, 2020), whereas our target label is LLM–human agreement rather than answer quality, and human verification can provide both positive and negative anchors. We therefore treat the problem as adaptive PU learning, where the positive and unlabeled groups are refined as evidence accumulates. The propagation step is related to graph-based label smoothing and transport methods (Zhu & Ghahramani, 2002; Zhou et al., 2003; Cuturi, 2013; Peyré & Cuturi, 2019; Chapel et al., 2020), but AURA uses a conservative transport-based operator for sparse, budgeted evidence propagation rather than full distribution alignment. The verification policy is related to active learning, selective prediction, and calibration (Settles, 2009; Geifman & El-Yaniv, 2017; Guo et al., 2017): it directs human review to uncertain or influential cases. Unlike methods that use feedback to optimize a judge's prompt or decoding, AURA audits the judge's existing decisions.

3 PROBLEM FORMULATION AND NOTATION

Let $\mathcal{D} = \{(Q_i, R_i^1, R_i^2, J_i^{\mathrm{H}}, J_i^{\mathrm{L}})\}_{i=1}^n$ be a collection of samples, where Q_i is the question or prompt, R_i^1 and R_i^2 are two candidate responses, J_i^{L} is the initial LLM judge, and J_i^{H} is the human judge. Figure 1 illustrates this pairwise data structure. The LLM judge assigns a strict preference $J_i^{\mathrm{L}} \in \{1, 2\}$, where $J_i^{\mathrm{L}} = k$ indicates that R_i^k is preferred. Human labels, when available, are denoted by $J_i^{\mathrm{H}} \in \{1, 2\}$. We exclude tie cases. For each comparison, the LLM judge induces an initial winner–loser assignment $R_i^w = R_i^{J_i^{\mathrm{L}}}$, $R_i^\ell = R_i^{3-J_i^{\mathrm{L}}}$. Here, R_i^w and R_i^ℓ are interpreted as the LLM-induced preliminary winner and loser, rather than ground-truth human preferences; they will be refined using the available human labels. For each example, we define the human-consistency label $y_i = \mathbf{1}\{J_i^{\mathrm{L}} = J_i^{\mathrm{H}}\}$, which indicates whether the LLM preference agrees with the human preference. This label is observed only for human-verified examples and remains latent for unverified examples. Thus, $y_i = 1$ indicates that the LLM preference agrees with the human preference, while $y_i = 0$ indicates that the LLM preference should be corrected. The corresponding agreement-positive and agreement-negative sets are $\mathcal{P}^\star = \{i : y_i = 1\}$, $\mathcal{N}^\star = \{i : y_i = 0\}$. Our goal is to estimate $\mathbb{P}(y_i = 1 \mid Q_i, R_i^1, R_i^2, J_i^{\mathrm{L}})$ for unverified examples, and use this estimate to refine the LLM-induced preference signal under a limited human-review budget.

Let $\mathcal{P}_s$ be the small human-verified positive seed set used to initialize anchors. The refinement pool $\mathcal{U}$ contains examples whose labels are updated; $\mathcal{V}^+$ and $\mathcal{V}^-$ collect verified human-consistent and inconsistent examples, while $\mathcal{W}^{(t)}$ tracks selected examples with pending labels. These sets evolve as verification proceeds. Notation and implementation constants are summarized in Appendix A.

In standard PU-learning terms, the latent label y_i plays the role of the binary class label, human-verified agreement examples ($\mathcal{V}^+$, and the seed set $\mathcal{P}_s$) form the observed positive set, the remaining unverified comparisons in $\mathcal{U}$ form the unlabeled mixture, and the responsibility $q_i^{(t)}$ (formalized in Section 4) estimates the posterior probability $\mathbb{P}(y_i = 1 \mid \cdot)$ that an unlabeled example belongs to the latent positive class. AURA extends this standard PU setup along three axes that classical PU methods treat as fixed: it iteratively updates the representation used to measure similarity, propagates evidence through transport-based refinement rather than a one-shot classifier fit, and actively selects which unverified examples to verify next rather than passively observing a fixed labeled sample.

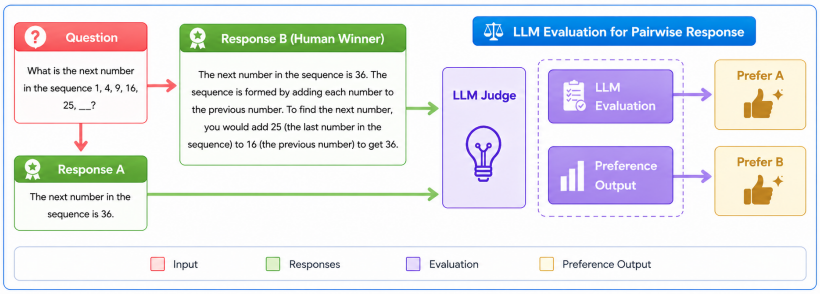

Figure 1: Illustration of the pairwise LLM-as-a-judge setting. Given a question and two candidate responses, the LLM judge produces an initial preference between the responses, which may differ from the human preference.

4 METHODOLOGY

We now present AURA, a progressive refinement framework for converting noisy LLM-judge preferences into preference signals that align with human judgments. Starting from preliminary preferences produced by an LLM judge and a small set of human-verified examples, AURA iteratively learns a task-specific representation, updates human-consistency estimates, propagates trust through a conservative transport step, and selectively queries additional human labels. Figure 2 illustrates the overall workflow, and Algorithm 1 in Appendix summarizes the full procedure.

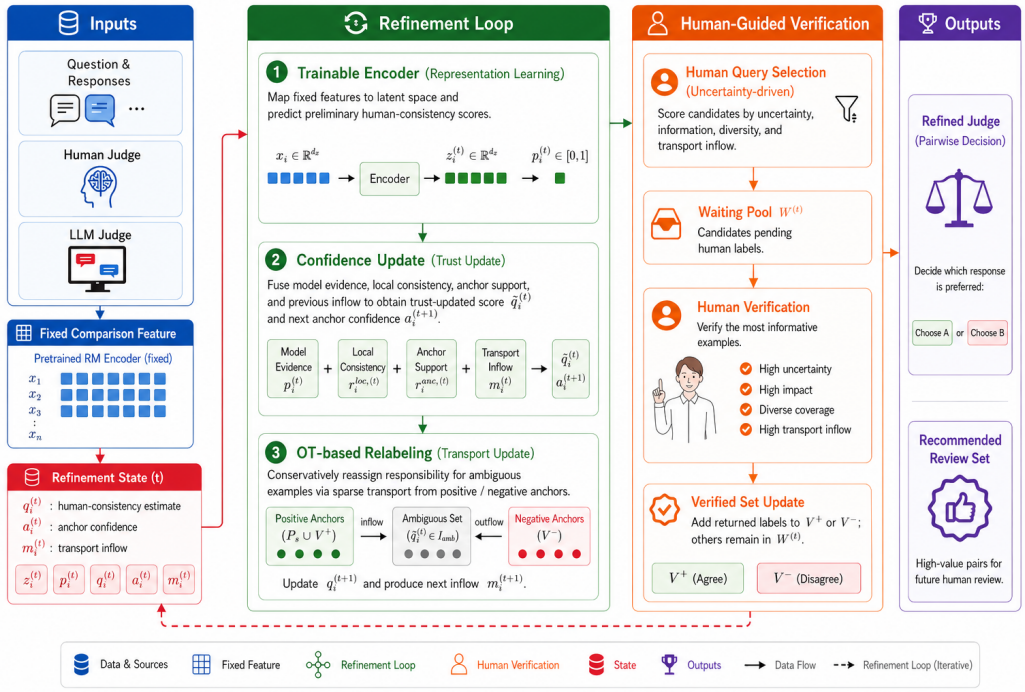

Figure 2: Overview of AURA framework. At iteration t, the current state $(q^{(t)}, a^{(t)}, m^{(t)})$ is used to train the encoder, which produces the latent representations $z^{(t)}$ and preliminary scores $p^{(t)}$. The trust-update module uses them to compute the consistency estimate $\tilde{q}^{(t)}$ and the next anchor confidence $a^{(t+1)}$. The transport step then refines $\tilde{q}^{(t)}$ into the next-round consistency estimate $q^{(t+1)}$ and produces the next inflow state $m^{(t+1)}$. Finally, the verification step selects examples for human review; returned labels are added to $\mathcal{V}^+$ or $\mathcal{V}^-$, while pending examples are kept in $\mathcal{W}^{(t+1)}$.

4.1 Progressive Representation and Trust Refinement

For each pairwise comparison, we first construct a fixed comparison feature $x_i = E_{\mathrm{RM}}(Q_i, R_i^w) - E_{\mathrm{RM}}(Q_i, R_i^\ell) \in \mathbb{R}^{d_x}$, where E_{RM} is a pretrained reward-model encoder used only for feature extraction and kept frozen throughout refinement. At iteration t, a trainable encoder maps x_i into a task-specific latent representation $z_i^{(t)} \in \mathbb{R}^{d_z}$, and a prediction head produces a preliminary human-consistency estimate $p_i^{(t)} = \sigma(\ell_i^{(t)})$. This score is not treated as the final label; instead, it provides model evidence for the subsequent trust and transport updates.

The encoder is trained from the current refinement state $(q^{(t)}, a^{(t)})$. The goal is to fit the available human-verified labels, use reliable unverified examples as additional soft supervision, and keep the learned latent space stable across refinement rounds. We use the objective $\mathcal{L}_{\mathrm{enc}}^{(t)} = \mathcal{L}_{\mathrm{ver}}^{(t)} + \lambda_{\mathrm{soft}} \mathcal{L}_{\mathrm{soft}}^{(t)} + \lambda_{\mathrm{geo}} \mathcal{R}_{\mathrm{geo}}^{(t)} + \lambda_{\mathrm{anchor}} \mathcal{R}_{\mathrm{anchor}}^{(t)}$. Here $\mathcal{L}_{\mathrm{ver}}^{(t)}$ is the supervised loss on human-verified examples, and $\mathcal{L}_{\mathrm{soft}}^{(t)}$ trains on high-confidence unverified examples using their current estimation $q_i^{(t)}$ as soft labels. The two regularization terms stabilize the representation: $\mathcal{R}_{\mathrm{geo}}^{(t)}$ enforces local smoothness, while $\mathcal{R}_{\mathrm{anchor}}^{(t)}$ discourages high-confidence examples from drifting across rounds. Full network parameterization and loss definitions are provided in Appendix B.1.

4.2 TRUST UPDATE, CONSERVATIVE TRANSPORT, AND VERIFICATION

After encoder training, AURA combines model, local, anchor, and previous-round transport evidence in a trust logit $u_i^{(t)} = \lambda_p \log \frac{p_i^{(t)}}{1-p_i^{(t)}} + \lambda_{\text{loc}}(2r_i^{\text{loc},(t)} - 1) + \lambda_{\text{anc}}(2r_i^{\text{anc},(t)} - 1) + \lambda_m(2m_i^{(t)} - 1) + \beta_0$. Applying the sigmoid gives the trust-updated estimate $\tilde{q}_i^{(t)}$, while the same evidence updates anchor confidence $a_i^{(t+1)}$. Detailed trust-update rules are given in Appendix B.2.

We then apply a conservative transport update to uncertain examples. Let $\mathcal{M}^{(t)} = \{j \in \mathcal{U} : \tilde{q}_j^{(t)} \in I_{\text{amb}}\}$ be the uncertain set. For each $j \in \mathcal{M}^{(t)}$, the transport module computes positive and negative inflow scores $m_j^{+,(t)}$ and $m_j^{-,(t)}$ in the learned latent space. Rather than solving a full optimal transport problem, we use a sparse budgeted propagation rule:

$$q_j^{+,(t)} = \tilde{q}_j^{(t)} + \eta_+ m_j^{+,(t)} \left(1 - \tilde{q}_j^{(t)}\right), \qquad q_j^{(t+1)} = q_j^{+,(t)} - \eta_- m_j^{-,(t)} q_j^{+,(t)}. \tag{1}$$

Examples outside $\mathcal{M}^{(t)}$ keep their trust-updated estimates, $q_j^{(t+1)} = \tilde{q}_j^{(t)}$. For the next round, we store $m_j^{(t+1)} = m_j^{+,(t)}$, while the negative inflow is used only in the current update. Details of the inflow computation are provided in Appendix B.3.

Finally, AURA selects unresolved examples for human verification. Returned labels are added to $\mathcal{V}^+$ or $\mathcal{V}^-$, while pending examples are kept in $\mathcal{W}^{(t)}$ to avoid repeated queries. The loop stops when the refined estimates, hard assignments, transport inflow, and verified set size become stable. Verification and stopping details are provided in Appendix B.4.

Overall, each refinement round follows

$$(q^{(t)}, a^{(t)}, m^{(t)}) \to (z^{(t)}, p^{(t)}) \to (\tilde{q}^{(t)}, a^{(t+1)}) \to (q^{(t+1)}, a^{(t+1)}, m^{(t+1)}),$$

with selective human verification updating $\mathcal{V}^+$, $\mathcal{V}^-$, and the waiting set $\mathcal{W}^{(t)}$.

5 THEORETICAL RESULTS

The adaptive outer loop of AURA alternates representation learning, trust refinement, transport-based propagation, and selective human verification. Because verification, thresholding, and top-K_{tr} selection may change the active sets across outer rounds, we establish stability of the fixed-configuration refinement operators and characterize when transport reduces human-consistency estimation error. Additional results on uncertainty-targeted transport, latent geometry, and finite-round error propagation are deferred to Appendix C.

5.1 STABILITY OF TRUST AND ENCODER REFINEMENT

Assumption 1 (Regularity of the fixed-round refinement)**.** *For each fixed outer round t, the encoder objective $\mathcal{L}_{\text{enc}}^{(t)}(\Theta)$ is continuously differentiable with L_Θ-Lipschitz gradient. The trust weights $\lambda_p, \lambda_{\text{loc}}, \lambda_{\text{anc}}, \lambda_m$ and offset β_0 are finite and fixed, $p_i^{(t)} \in [\epsilon_p, 1-\epsilon_p]$ for some $\epsilon_p \in (0, 1/2)$, and $r_i^{\text{loc},(t)}, r_i^{\text{anc},(t)} \in [0,1]$.*

For unconstrained indices, the trust update is $\tilde{q}_i = \sigma(u_i)$, where

$$u_i = \lambda_p \log \frac{p_i}{1-p_i} + \lambda_{\text{loc}}(2r_i^{\text{loc}} - 1) + \lambda_{\text{anc}}(2r_i^{\text{anc}} - 1) + \lambda_m(2m_i - 1) + \beta_0. \tag{2}$$

Theorem 1 (Stability of the trust update)**.** *Under Assumption 1, the trust update is Lipschitz on unconstrained indices. In particular, when $(p, r^{\text{loc}}, r^{\text{anc}})$ are fixed,*

$$\|\tilde{q}(m) - \tilde{q}(m')\|_1 \le \frac{|\lambda_m|}{2} \|m - m'\|_1. \tag{3}$$

Lemma 1 (Within-round encoder descent)**.** *Under Assumption 1, a gradient step with $0 < \eta < 2/L_\Theta$ satisfies*

$$\mathcal{L}_{\text{enc}}^{(t)}(\Theta^{(t,s+1)}) \le \mathcal{L}_{\text{enc}}^{(t)}(\Theta^{(t,s)}) - \eta \left(1 - \frac{L_\Theta \eta}{2}\right) \left\|\nabla_\Theta \mathcal{L}_{\text{enc}}^{(t)}(\Theta^{(t,s)})\right\|_2^2. \tag{4}$$

Theorem 1 and Lemma 1 are fixed-round statements. They do not imply global continuity or monotonicity across adaptive outer iterations, where verified sets, anchor pools, and transport neighborhoods may change.

5.2 Transport error decomposition

Let $y_i = \mathbf{1}\{J_i^{\mathrm{L}} = J_i^{\mathrm{H}}\}$ denote the latent human-consistency label, and let $\tilde{q}^{(t)}$ denote the trust-refined score before transport. For $\nu \in \{+,-\}$, write the transport matrix as $T_{ij}^{\nu,(t)} = B_t^{\nu}\pi_i^{\nu,(t)}\Gamma_{ij}^{\nu,(t)}$, with nonnegative weights satisfying $\sum_i \pi_i^{\nu,(t)} \le 1$ and $\sum_j \Gamma_{ij}^{\nu,(t)} \le 1$. Define $m_j^{\nu,(t)} = \sum_i T_{ij}^{\nu,(t)}$, and assume $0 \le \eta_\nu m_j^{\nu,(t)} \le 1$. The update rule can refer to (1).

Define the effective transport strength

$$\theta_j^{(t)} = 1 - (1 - \eta_+ m_j^{+,(t)})(1 - \eta_- m_j^{-,(t)}) \tag{5}$$

and the effective incoming transport pointing toward the wrong human-consistency label,

$$h_j^{(t)} = (1 - y_j)(1 - \eta_- m_j^{-,(t)})\eta_+ m_j^{+,(t)} + y_j \eta_- m_j^{-,(t)}. \tag{6}$$

Theorem 2 (Exact one-step transport error decomposition)**.** *Under the transport conditions above, for each target j,*

$$|q_j^{(t+1)} - y_j| = (1 - \theta_j^{(t)})|\tilde{q}_j^{(t)} - y_j| + h_j^{(t)}. \tag{7}$$

Hence transport strictly reduces both absolute and squared error for sample j if and only if

$$h_j^{(t)} < \theta_j^{(t)}|\tilde{q}_j^{(t)} - y_j|. \tag{8}$$

Moreover, if $G^{(t)} = \sum_j \theta_j^{(t)}(2 - \theta_j^{(t)})(\tilde{q}_j^{(t)} - y_j)^2$, then

$$\|q^{(t+1)} - y\|_2^2 - \|\tilde{q}^{(t)} - y\|_2^2 \le -G^{(t)} + 2\sum_j h_j^{(t)}. \tag{9}$$

Theorem 2 isolates the central trade-off of AURA transport: $\theta_j^{(t)}|\tilde{q}_j^{(t)} - y_j|$ is the amount of current error that can be corrected, while $h_j^{(t)}$ is the amount of incoming evidence pointing in the wrong direction.

For $\theta_j^{(t)} > 0$, Equation (7) writes the new absolute error as a weighted average of the current absolute error and $h_j^{(t)}/\theta_j^{(t)}$. This ratio is the fraction of effective incoming weight that supports the wrong label. Transport reduces the error exactly when this fraction is smaller than the current error. Positive support raises an uncertain comparison's score toward one. This helps when the judge agrees with the human label and harms when it does not. Negative support lowers the score and has the opposite effect. The result separates the quality of the incoming evidence from the strength of the update.

The results explain three design choices in AURA. First, transport is restricted to uncertain comparisons, whose scores lie away from zero and one. The uncertainty-band bound relates the change in their total squared error to the update strengths and the amount of incoming weight that supports wrong labels. Second, confidence weights control how much each positive reference can influence other comparisons. The source-error bound measures the effect of incorrect references by the transport mass they contribute, rather than simply by their number. Third, the learned representation and target selection determine where this influence is sent. Under local label regularity on the retained links, the geometry bound controls transport between different labels through distances weighted by transported mass. This connects useful similarity to shared human-consistency labels, rather than merely a shared topic. Appendix C provides (i) the uncertainty-band bound, (ii) the source-confidence and latent-geometry bounds, (iii) the bound on changes in final binary decisions, and (iv) the error bound across refinement rounds.

6 Experiments

We evaluate AURA in three parts. First, we use simulations to test whether the method can recover latent human-consistency structure under noisy judge signals and selective verification. Second,

Table 1: Simulation results averaged over five independent runs. Each run contains 640 examples: 160 source positive examples and 480 target/unlabeled examples, among which 168 are hidden positives and 312 are negatives. Adjusted accuracy, accuracy difference, and flip rate are reported as mean (max–min range) across runs. Bold values indicate positive improvements. The full data-generating process is given in Appendix E.2; the same corruption rate is used across settings, so the small differences in original accuracy across rows reflect only the different CD/DD corruption mechanisms.

Simulation Results for CD/DD Noise and SCAR/SAR Verification

Noise	Verification	Orig. Acc.	Adj. Acc.	Diff.	Flip Rate
CD	SCAR	76.47%	81.12% (4.53%)	**+4.66%** (2.97%)	30.16% (6.41%)
CD	SAR	76.47%	81.78% (3.75%)	**+5.31%** (2.34%)	30.12% (5.00%)
DD	SCAR	76.53%	80.78% (4.84%)	**+4.25%** (5.31%)	22.69% (5.47%)
DD	SAR	76.53%	81.28% (3.12%)	**+4.75%** (4.37%)	23.62% (5.00%)

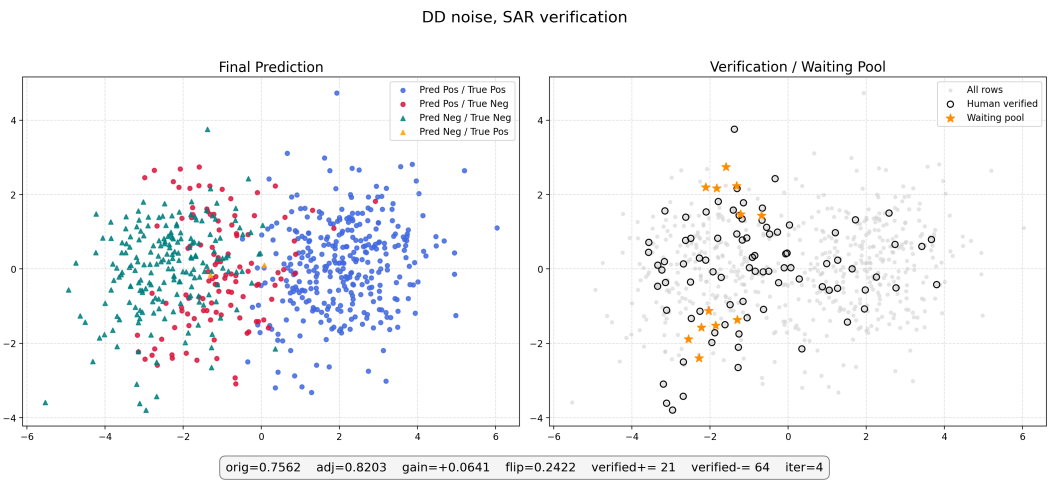

Figure 3: DD+SAR simulation visualization. **Left**: verified and waiting-pool examples under selective verification. **Right**: final hard prediction after applying AURA.

we evaluate real LLM-as-a-judge data across multiple judge models and question types. Third, we provide ablation studies to examine the contribution of each component. The evaluation metrics are provided in Appendix E.1.

6.1 Simulation

We simulate two latent groups indicating whether the initial judge decision agrees with human preference. AURA observes only noisy comparison features and corrupted agreement labels. We cross class-dependent or feature-dependent corruption (CD/DD) with SCAR or feature-dependent SAR verification; the design is detailed in Appendix E.2.

Table 1 shows gains of 4.25–5.31 points across settings, with adjusted accuracy reaching 80.78%–81.78%. Five seeds do not establish a difficulty ordering among CD/DD and SCAR/SAR. Figure 3 shows DD+SAR; other plots and geometry sensitivity are in Appendices E.6 and E.3.

Table 2: Coding results for GPT-5.4 and Qwen with 20% verified baseline data. Accuracy and gain are reported as mean (max–min range). Human Verif. Req. denotes the number of human-verified examples required by each method. Baselines use 20% verified data, while AURA uses a substantially smaller adaptive verification budget.

Coding Results with 20% Verified Baseline Data					
Metric	**Logistic Reg.**	**MLP**	**Random Forest**	**Label Prop.**	**AURA**
Model: GPT-5.4 (n = 2949, Orig. Acc = 63.92%)					
Adj. Acc.	59.54% (3.56)	60.83% (2.80)	63.23% (1.87)	60.50% (9.92)	**67.20% (2.00)**
Gain	-4.38% (3.56)	-3.09% (2.80)	-0.69% (1.87)	-3.42% (9.92)	**+3.28% (2.00)**
Human Verif. Req.	590	590	590	590	**100.6**
Model: Qwen (n = 8884, Orig. Acc = 53.73%)					
Adj. Acc.	58.47% (1.59)	57.44% (1.83)	59.20% (0.96)	53.77% (2.11)	**77.74% (9.78)**
Gain	+4.75% (1.59)	+3.72% (1.83)	+5.47% (0.96)	+0.04% (2.11)	**+24.02% (9.78)**
Human Verif. Req.	1777	1777	1777	1777	**278.8**

6.2 Real-Data Evaluation

We next evaluate AURA on real LLM-as-a-judge data constructed from MT-Bench (Zheng et al., 2023) and Chatbot Arena (Chiang et al., 2024). We use three question types, coding, math reasoning, and structured factual questions, and include one mixed question type MT-Bench setting. We evaluate five judge models: GPT-5.4, GPT-5.4-mini (OpenAI, 2026), Gemini-2.5-Flash (DeepMind, 2024), Qwen2.5-7B-Instruct (Team, 2024), and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023). The full setup and additional results are in Appendices E.5 and E.7; hyperparameter sensitivity is in Appendix E.4.

Table 2 shows that AURA consistently improves the original LLM-judge signal across datasets, judge models, and question types. The gains are consistent across judge families and evaluation settings. The main advantage of AURA is label efficiency. We compare against baselines trained with 3%, 20%, and 80% verified data, while AURA uses a final verification budget close to the 3% setting. Despite this much smaller human-label budget, AURA is competitive with or better than the baselines in most configurations. Table 2 gives a representative example. On GPT-5.4 and Qwen Coding, AURA achieves better adjusted accuracy while using only about 101 and 279 verified examples, compared with 590 and 1,777 examples used by baselines. AURA also has modest additional computational cost. Its dominant overhead is the transport-based inflow update, which requires $O(n_u n_a d_z)$ time and $O(n_u n_a)$ memory with a direct similarity-matrix implementation.

Measured runtime and peak memory: AURA's dominant overhead is the transport-based inflow update. If n_u uncertain examples are compared with n_a anchor examples in a d_z-dimensional latent space, a direct similarity-matrix implementation costs $O(n_u n_a d_z)$ time and $O(n_u n_a)$ memory. Table 16 in Appendix E.8 reports measured runtime and peak memory on the real-data experiments.

6.3 Ablation Study

We ablate the main components of AURA on the Coding setting with Gemini-2.5-Flash. We vary four components: whether the encoder is trained, whether the trust update uses the full confidence state or only the model probability p_i, whether the transport update is enabled, and whether human queries are selected by the proposed query score or uniformly at random. Implementation details for these four components are provided in Appendices B.1, B.2, B.3, and B.4, respectively.

Table 3 reports a full factorial ablation of the four main components. The complete model achieves the best adjusted accuracy, improving the original judge accuracy from 63.55% to 66.88%. Replacing human querying with random selection reduces the gain from 3.33 to 2.80 percentage points, and using only p_i in the trust update also lowers performance, showing that both adaptive verification and the full trust state are useful. The transport update has the largest effect: disabling it causes all variants to flip more than two thousand assignments and sharply reduces accuracy, indicating

Table 3: Full factorial ablation on the Coding setting with GEMINI-2.5-FLASH. The original judge accuracy is 63.55%. We vary four components: encoder training, full trust-state update, transport update, and human query selection.

Full Trust	Transport	Human Query	Trained Encoder			Untrained Encoder		
			Adj. Acc.	Gain	Flips	Adj. Acc.	Gain	Flips
✓	✓	✓	**66.88%**	**+3.33%**	164	64.72%	+1.17%	183
✓	✓	✗	66.35%	+2.80%	153	66.13%	+2.59%	213
✗	✓	✓	66.81%	+3.26%	168	64.50%	+0.96%	179
✗	✓	✗	66.67%	+3.12%	168	66.03%	+2.48%	220
✓	✗	✓	53.40%	-10.14%	2250	40.78%	-22.77%	2576
✓	✗	✗	53.16%	-10.39%	2257	40.71%	-22.84%	2574
✗	✗	✓	39.72%	-23.83%	2720	41.67%	-21.88%	2547
✗	✗	✗	39.72%	-23.83%	2720	41.03%	-22.52%	2581

that the transport component plays an important role in stabilizing the refinement state and propagating anchor evidence. Trained encoders also generally outperform untrained ones when transport is enabled. Overall, the ablation supports the coupled design of AURA, combining representation learning, trust-state refinement, transport-based stabilization, and adaptive verification.

7 DISCUSSION AND FUTURE WORK

Across simulation and real-data experiments, AURA matches or improves supervised baselines in adjusted accuracy while using fewer human-verified comparisons. These results support refining judge reliability as evidence accumulates, with selective verification guiding corrections and directing scarce human review to cases where added evidence matters most.

The target is agreement with the human-preference protocol that produced J_i^{H}, not an absolute measure of response quality. Because open-ended preferences can be ambiguous, uncertainty in y_i reflects both limited verification and annotator disagreement. We therefore interpret higher accuracy as closer agreement with the protocol, while accounting for ambiguity and disagreement in the human labels used to define it.

This distinction also motivates a key direction for future work: improving how refinement represents ambiguous comparisons. Task-adaptive representations could better capture the distinctions underlying human preferences, while explicitly modeling annotator disagreement could help separate preference ambiguity from uncertainty that additional verification can resolve. Beyond representation, the current formulation is limited to pairwise preferences and the binary agreement label defined in Section 3. Extending auditing to ordinal or numerical ratings would require a corresponding generalization of the responsibility state $q_i^{(t)}$, while retaining the principle of jointly refining reliability estimates and allocating human verification.

AI USE STATEMENT

We used generative AI tools (LLM-based coding assistants) to help draft and revise the text of this paper, to generate and reformat LaTeX tables from experimental result files, and to write and debug data-processing and analysis scripts (e.g., significance testing, metric computation, and job-submission scripts for the compute cluster). We did not use generative AI tools to generate the underlying research idea, the theoretical results, or the experimental design; these were developed by the authors. All AI-assisted writing, code, and generated tables were reviewed, checked against the underlying experimental data, and edited by the authors before inclusion, and all reported numerical results were independently verified against the raw experiment outputs. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Ethics statement

This work does not involve new data collection from human subjects. The real-data experiments use pairwise LLM-response comparisons and existing human preference judgments drawn from the publicly available MT-Bench and Chatbot Arena benchmarks; we do not collect, store, or process any personally identifiable information beyond what is already present in these public releases. AURA is intended as an auditing tool that flags LLM-judge decisions for human review rather than a mechanism for autonomous, unreviewed decision-making, and its purpose is to reduce reliance on potentially biased automated judges by allocating human oversight where it is most needed. We are not aware of any conflicts of interest, sponsorship-related concerns, or other ethical issues raised by this submission.

Reproducibility statement

Appendix A summarizes notation. Appendices B.1–B.4 give implementation details, Appendix C contains supplementary theory, and Appendix D gives complete proofs. Appendix E.2 specifies the simulation; its metrics and geometry sensitivity are in Appendices E.1 and E.3. Real-data setup, hyperparameter sensitivity, full results, and runtime measurements are in Appendices E.5, E.4, E.7, and E.8.

References

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*, 2022. URL https://arxiv.org/abs/2204.05862.

Jessa Bekker and Jesse Davis. Learning from positive and unlabeled data: A survey. *Machine Learning*, 109(4):719–760, 2020. doi: 10.1007/s10994-020-05877-5. URL https://link.springer.com/article/10.1007/s10994-020-05877-5.

Antonin Berthon, Bo Han, Gang Niu, Tongliang Liu, and Masashi Sugiyama. Confidence scores make instance-dependent label-noise learning possible. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pp. 825–836. PMLR, 2021.

Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952. doi: 10.1093/biomet/39.3-4.324. URL https://doi.org/10.1093/biomet/39.3-4.324.

Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001.

Laetitia Chapel, Mokhtar Z. Alaya, and Gilles Gasso. Partial optimal transport with applications on positive-unlabeled learning. In *Advances in Neural Information Processing Systems*, volume 33, pp. 2903–2913, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/1e6e25d952a0d639b676ee20d0519ee2-Abstract.html.

De Cheng, Tongliang Liu, Yixiong Ning, Nannan Wang, Bo Han, Gang Niu, Xinbo Gao, and Masashi Sugiyama. Instance-dependent label-noise learning with manifold-regularized transition matrix estimation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16630–16639, 2022.

Wei-Lin Chiang, Lianmin Zheng, Ying Sheng, Anastasios Nikolas Angelopoulos, Tianle Li, Dacheng Li, Banghua Zhu, Hao Zhang, Michael Jordan, Joseph E. Gonzalez, and Ion Stoica. Chatbot arena: An open platform for evaluating LLMs by human preference. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 8359–8388. PMLR, 2024. URL https://proceedings.mlr.press/v235/chiang24b.html.

Paul F. Christiano, Jan Leike, Tom B. Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://papers.nips.cc/paper_files/paper/2017/hash/d5e2c0adad503c91f91df240d0cd4e49-Abstract.html.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in Neural Information Processing Systems*, volume 26, pp. 2292–2300, 2013. URL https://papers.nips.cc/paper_files/paper/2013/hash/af21d0c97db2e27e13572cbf59eb343d-Abstract.html.

Google DeepMind. Gemini 2.5 flash. https://deepmind.google/technologies/gemini/, 2024.

Marthinus C. du Plessis, Gang Niu, and Masashi Sugiyama. Analysis of learning from positive and unlabeled data. In *Advances in Neural Information Processing Systems*, volume 27, pp. 703–711, 2014. URL https://papers.nips.cc/paper/5509-analysis-of-learning-from-positive-and-unlabeled-data.

Yann Dubois, Xuechen Li, Rohan Taori, Tianyi Zhang, Ishaan Gulrajani, Jimmy Ba, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. AlpacaFarm: A simulation framework for methods that learn from human feedback. In *Advances in Neural Information Processing Systems*, volume 36, pp. 30039–30069, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/5fc47800ee5b30b8777fdd30abcaaf3b-Abstract-Conference.html.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B. Hashimoto. Length-controlled AlpacaEval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024. URL https://arxiv.org/abs/2404.04475.

Charles Elkan and Keith Noto. Learning classifiers from only positive and unlabeled data. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 213–220. ACM, 2008. doi: 10.1145/1401890.1401920. URL https://dl.acm.org/doi/10.1145/1401890.1401920.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/4a8423d5e91fda00bb7e46540e2b0cf1-Abstract.html.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q. Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pp. 1321–1330. PMLR, 2017. URL https://proceedings.mlr.press/v70/guo17a.html.

Albert Q. Jiang, Alexandre Sablayrolles, and Antoine et al. Roux. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=8euJaTveKw.

Ryuichi Kiryo, Gang Niu, Marthinus C. du Plessis, and Masashi Sugiyama. Positive-unlabeled learning with non-negative risk estimator. In *Advances in Neural Information Processing Systems*, volume 30, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/7cce53cf90577442771720a370c3c723-Abstract.html.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. RewardBench: Evaluating reward models for language modeling. In *Findings of the Association for Computational Linguistics: NAACL 2025*, pp. 1755–1797, Albuquerque, New Mexico, 2025. Association for Computational Linguistics. doi: 10.18653/v1/2025.findings-naacl.96. URL https://aclanthology.org/2025.findings-naacl.96/.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-Eval: NLG evaluation using GPT-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 2511–2522, Singapore, 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.emnlp-main.153. URL https://aclanthology.org/2023.emnlp-main.153/.

OpenAI. Introducing gpt-5.4. https://openai.com/index/introducing-gpt-5-4/, 2026.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pp. 27730–27744, 2022. URL https://proceedings.neurips.cc/paper/2022/hash/b1efde53be364a73914f58805a001731-Abstract-Conference.html.

Junsoo Park, Seungyeon Jwa, Meiying Ren, Daeyoung Kim, and Sanghyuk Choi. OffsetBias: Leveraging debiased data for tuning evaluators. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 1043–1067, Miami, Florida, USA, 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.57. URL https://aclanthology.org/2024.findings-emnlp.57/.

Gabriel Peyré and Marco Cuturi. Computational optimal transport. *Foundations and Trends in Machine Learning*, 11(5–6):355–607, 2019. doi: 10.1561/2200000073. URL https://arxiv.org/abs/1803.00567.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D. Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://papers.nips.cc/paper_files/paper/2023/hash/a85b405ed65c6477a4fe8302b5e06ce7-Abstract-Conference.html.

Burr Settles. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison, 2009. URL https://burrsettles.com/pub/settles.activelearning.pdf.

Skywork AI. Skywork-reward-llama-3.1-8b-v0.2. https://huggingface.co/Skywork/Skywork-Reward-Llama-3.1-8B-v0.2, 2024.

Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Yuan Tang, Alejandro Cuadron, Chenguang Wang, Raluca Ada Popa, and Ion Stoica. JudgeBench: A benchmark for evaluating LLM-based judges. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=G0dksFayVq.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Aman Singh Thakur, Kartik Choudhary, Venkat Srinik Ramayapally, Sankaran Vaidyanathan, and Dieuwke Hupkes. Judging the judges: Evaluating alignment and vulnerabilities in LLMs-as-judges. In *Proceedings of the Fourth Workshop on Generation, Evaluation and Metrics (GEM2)*, pp. 404–430, Vienna, Austria and virtual meeting, 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.gem-1.33/.

Peiyi Wang, Lei Li, Liang Chen, Zefan Cai, Dawei Zhu, Binghuai Lin, Yunbo Cao, Lingpeng Kong, Qi Liu, Tianyu Liu, and Zhifang Sui. Large language models are not fair evaluators. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 9440–9450, Bangkok, Thailand, 2024a. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.511. URL https://aclanthology.org/2024.acl-long.511/.

Yidong Wang, Zhuohao Yu, Wenjin Yao, Zhengran Zeng, Linyi Yang, Cunxiang Wang, Hao Chen, Chaoya Jiang, Rui Xie, Jindong Wang, Xing Xie, Wei Ye, Shikun Zhang, and Yue Zhang. PandaLM: An automatic evaluation benchmark for LLM instruction tuning optimization. In *International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=5Nn2BLV7SB.

Xiaobo Xia, Tongliang Liu, Bo Han, Mingming Gong, Jun Yu, Gang Niu, and Masashi Sugiyama. Part-dependent label noise: Towards instance-dependent label noise. In *Advances in Neural Information Processing Systems*, volume 33, pp. 7597–7610, 2020.

Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating large language models at evaluating instruction following. In *International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=tr0KidwPLc.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging LLM-as-a-judge with MT-Bench and chatbot arena. In *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://arxiv.org/abs/2306.05685.

Dengyong Zhou, Olivier Bousquet, Thomas N. Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in Neural Information Processing Systems*, volume 16, pp. 321–328, 2003. URL https://proceedings.neurips.cc/paper_files/paper/2003/hash/87682805257e619d49b8e0dfdc14affa-Abstract.html.

Lianghui Zhu, Xinggang Wang, and Xinlong Wang. JudgeLM: Fine-tuned large language models are scalable judges. In *International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=xsELpEPn4A.

Xiaojin Zhu and Zoubin Ghahramani. Learning from labeled and unlabeled data with label propagation. Technical Report CMU-CALD-02-107, Carnegie Mellon University, 2002. URL https://pages.cs.wisc.edu/~jerryzhu/pub/CMU-CALD-02-107.pdf.

APPENDIX

A NOTATION AND FIGURES

Table 4: Summary of main notation. Detailed quantities used only locally are defined near their corresponding equations.

Symbol	Meaning
$\mathcal{D}$	pairwise comparison dataset
Q_i	prompt or question
R_i^1, R_i^2	two candidate responses
J_i^{L}	preference given by the LLM judge
J_i^{H}	human preference, when observed
R_i^w, R_i^ℓ	winner and loser induced by J_i^{L}
y_i	judge–human consistency label, $y_i = \mathbf{1}\{J_i^{\mathrm{L}} = J_i^{\mathrm{H}}\}$
$\mathcal{P}^\star, \mathcal{N}^\star$	ideal human-consistent and human-inconsistent sets
$\mathcal{U}$	refinement pool
$\mathcal{P}_s$	trusted positive seed/reference set available before refinement
$\mathcal{V}^+, \mathcal{V}^-$	verified human-consistent and human-inconsistent examples
$\mathcal{W}^{(t)}$	waiting pool of examples selected for review but not yet incorporated
E_{RM}	frozen reward-model or text encoder
x_i	fixed directional comparison feature
$z_i^{(t)}$	learned latent representation at iteration t
$\tilde{z}_i^{(t)}$	normalized latent representation
$p_i^{(t)}$	preliminary human-consistency estimate from the encoder
$q_i^{(t)}$	refined human-consistency estimate at iteration t
$\tilde{q}_i^{(t)}$	trust-updated consistency estimate before transport refinement
$a_i^{(t)}$	anchor confidence at iteration t
$m_i^{(t)}$	carried positive transport inflow from the previous round
τ_{soft}	quantile level controlling soft-supervision selection
τ_{geo}	inverse-temperature controlling concentration of geometric-neighborhood weights
τ_{tr}	inverse-temperature controlling concentration of transport weights
$r_i^{\mathrm{loc},(t)}$	local-consistency evidence score
$r_i^{\mathrm{anc},(t)}$	anchor-support evidence score
$\mathcal{M}^{(t)}$	uncertain examples receiving transport refinement
$\mathcal{B}_+^{(t)}, \mathcal{B}_-^{(t)}$	positive and negative anchor pools for transport
$m_j^{+,(t)}, m_j^{-,(t)}$	positive and negative transport inflows
$\mathcal{C}^{(t)}$	candidate set for human verification
$H_i^{(t)}$	final query score for human verification

Algorithm 1 Progressive PU Refinement Workflow

Require: Pairwise dataset $\mathcal{D}$, reward encoder E_{RM}, human-verification oracle

$X \leftarrow \mathrm{BuildDiffFeatures}(\mathcal{D}, E_{\mathrm{RM}})$
$(\mathcal{P}_s, \mathcal{V}^+, \mathcal{V}^-, \mathcal{W}, q, a, m) \leftarrow \mathrm{InitializeState}(\mathcal{D})$
for $t = 0, \ldots, T-1$ **do**
  $(z^{(t)}, p^{(t)}) \leftarrow \mathrm{TrainEncoder}(X, q, a, m, \mathcal{V}^+, \mathcal{V}^-)$
  $(\tilde{q}^{(t)}, a) \leftarrow \mathrm{TrustUpdate}(z^{(t)}, p^{(t)}, q, a, m, \mathcal{P}_s, \mathcal{V}^+, \mathcal{V}^-)$
  $(q, m) \leftarrow \mathrm{TransportUpdate}(\tilde{q}^{(t)}, a, z^{(t)}, \mathcal{P}_s, \mathcal{V}^+, \mathcal{V}^-)$
  $(\mathcal{V}^+, \mathcal{V}^-, \mathcal{W}) \leftarrow \mathrm{SelectForVerification}(q, m, z^{(t)}, p^{(t)}, \mathcal{W})$
  if $\mathrm{StopCriterion}(q, m, \mathcal{V}^+, \mathcal{V}^-)$ **then**
    break
  end if
end for
return refined responsibilities q, refined positive set $\widehat{\mathcal{P}} = \{i : q_i \geq 1/2\}$, and verified sets $(\mathcal{V}^+, \mathcal{V}^-)$

B Details of methodology

B.1 Trainable encoder and representation learning

The refinement algorithm takes $\{x_i\}_{i=1}^n$ as fixed inputs and trains a task-specific encoder on top of these comparison features. At outer iteration t, the encoder maps x_i into a latent representation

$$z_i^{(t)} = W_2^{(t)} \,\mathrm{GELU}\left(W_1^{(t)} x_i + b_1^{(t)}\right) + b_2^{(t)} \in \mathbb{R}^{d_z}.$$

Here, GELU denotes the Gaussian Error Linear Unit activation function, defined as

$$\mathrm{GELU}(x) = x\Phi(x),$$

where $\Phi(\cdot)$ is the cumulative distribution function of the standard normal distribution. A linear prediction head then produces a logit

$$\ell_i^{(t)} = (w^{(t)})^\top z_i^{(t)} + b^{(t)}, \quad p_i^{(t)} = \sigma(\ell_i^{(t)}) = \frac{1}{1 + \exp(-\ell_i^{(t)})}.$$

We use $p_i^{(t)}$ as a preliminary estimate of whether the LLM-induced preference is human-consistent. For similarity-based operations, we use

$$\bar{z}_i^{(t)} = \mathrm{Norm}_\epsilon(z_i^{(t)}) = \frac{z_i^{(t)}}{\|z_i^{(t)}\| + \epsilon},$$

where $\epsilon > 0$ is fixed for numerical stability.

The encoder and prediction head are trained with verified supervision, confidence-aware soft supervision, local geometric regularization, and anchor-preserving regularization:

$$\mathcal{L}_{\mathrm{enc}}^{(t)} = \mathcal{L}_{\mathrm{ver}}^{(t)} + \lambda_{\mathrm{soft}} \mathcal{L}_{\mathrm{soft}}^{(t)} + \lambda_{\mathrm{geo}} \mathcal{R}_{\mathrm{geo}}^{(t)} + \lambda_{\mathrm{anchor}} \mathcal{R}_{\mathrm{anchor}}^{(t)}.$$

Verified supervision. For verified examples $i \in \mathcal{V}^+ \cup \mathcal{V}^-$, define the target $\widetilde{y}_i = \mathbf{1}\{i \in \mathcal{V}^+\}$. We use

$$\mathcal{L}_{\mathrm{ver}}^{(t)} = \frac{\sum_{i \in \mathcal{V}^+ \cup \mathcal{V}^-} \hat{\omega}_{\mathrm{ver}} \,\mathrm{BCE}(\widetilde{y}_i, \ell_i^{(t)})}{\max(1, |\mathcal{V}^+ \cup \mathcal{V}^-|)},$$

where BCE denotes binary cross-entropy between a binary target $y \in \{0, 1\}$ and a logit ℓ:

$$\mathrm{BCE}(y, \ell) = -y \log \sigma(\ell) - (1 - y) \log\big(1 - \sigma(\ell)\big), \qquad \hat{\omega}_{\mathrm{ver}} = 1 + \omega_{\mathrm{ver}} a_v.$$

Since verified examples have fixed anchor confidence a_v, this verified-example weight is constant across verified examples and iterations.

Confidence-aware soft supervision. We also train on high-confidence unverified examples using their current responsibilities as soft labels. Define the confidence of the current responsibility as $\kappa_i^{(t)} = \max\{q_i^{(t)}, 1 - q_i^{(t)}\}$. Among examples that are not verified and not source positives, denoted by $\mathcal{U}_{\text{soft}}^{(t)}$, we select the high-confidence set

$$\mathcal{H}_{\text{soft}}^{(t)} = \{i \in \mathcal{U}_{\text{soft}}^{(t)} : \kappa_i^{(t)} \geq \delta_{\text{soft}}^{(t)}\}, \quad \delta_{\text{soft}}^{(t)} = \text{Quantile}_{\tau_{\text{soft}}^{(t)}}(\{\kappa_j^{(t)} : j \in \mathcal{U}_{\text{soft}}^{(t)}\}).$$

The loss is

$$\mathcal{L}_{\text{soft}}^{(t)} = \frac{\sum_{i \in \mathcal{H}_{\text{soft}}^{(t)}} \omega_i^{\text{soft},(t)} \, \text{BCE}(q_i^{(t)}, \ell_i^{(t)})}{\sum_{i \in \mathcal{H}_{\text{soft}}^{(t)}} \omega_i^{\text{soft},(t)} + \epsilon}, \quad \omega_i^{\text{soft},(t)} = \frac{1}{2} + \frac{1}{2} a_i^{(t)}.$$

Local geometric regularization. To encourage locally similar examples to have consistent predictions and representations, we use a precomputed neighborhood system. Specifically, we construct a k-nearest-neighbor graph in the fixed comparison-feature space $\{x_i\}_{i=1}^n$ using cosine similarity. Let $\mathcal{N}(i)$ denote the k nearest neighbors of example i, and define

$$A_{ij} = \frac{\exp(\tau_{\text{geo}} \langle \bar{x}_i, \bar{x}_j \rangle)}{\sum_{r \in \mathcal{N}(i)} \exp(\tau_{\text{geo}} \langle \bar{x}_i, \bar{x}_r \rangle) + \epsilon}, \qquad j \in \mathcal{N}(i),$$

where $\bar{x}_i = x_i / (\|x_i\| + \epsilon)$. We define

$$\mathcal{R}_{\text{geo}}^{(t)} = \frac{1}{n} \sum_{i=1}^{n} \sum_{j \in \mathcal{N}(i)} A_{ij} \left[(p_i^{(t)} - p_j^{(t)})^2 + \eta_z \|z_i^{(t)} - z_j^{(t)}\|^2 \right].$$

The first term encourages nearby examples to have similar human-consistency scores, while the second term encourages smoothness of the learned latent representation. The coefficient $\eta_z \geq 0$ controls the strength of the latent-space smoothness penalty.

Anchor-preserving regularization. Since the encoder is re-trained across outer iterations, we penalize large movement of high-confidence examples in latent space:

$$\mathcal{R}_{\text{anchor}}^{(t)} = \frac{1}{n} \sum_{i=1}^{n} a_i^{(t)} \|z_i^{(t)} - z_i^{(t-1)}\|^2.$$

This term is omitted for $t = 0$. It stabilizes the learned representation around reliable anchors. Hence, the encoder parameters $\Theta^{(t)}$ are updated by minimizing $\mathcal{L}_{\text{enc}}^{(t)}(\Theta; q^{(t)}, a^{(t)}, z^{(t-1)})$.

B.2 Trust update

After encoder training, we use the learned representation and model score to update the refinement state. The trust update combines four signals: the encoder score $p_i^{(t)}$, local consistency $r_i^{\text{loc},(t)}$, anchor support $r_i^{\text{anc},(t)}$, and the previous transport inflow $m_i^{(t)}$.

Local consistency evidence. Using the neighborhood weights A_{ij}, we define

$$r_i^{\text{loc},(t)} = \Pi_{[0,1]} \left(1 - \sum_{j \in \mathcal{N}(i)} A_{ij} \left| p_i^{(t)} - p_j^{(t)} \right| \right),$$

where $\Pi_{[0,1]}(x) = \min\{1, \max\{0, x\}\}$ projects a scalar onto the interval $[0, 1]$. This score is large when example i's encoder score is consistent with those of its local neighbors.

Anchor-support evidence. We compare each example with positive and negative anchors in the learned latent space. Let

$$\mathcal{A}_+^{(t)} = \mathcal{P}_s \cup \mathcal{V}^+, \qquad \mathcal{A}_-^{(t)} = \mathcal{V}^-.$$

Define the normalized positive and negative prototypes as

$$\mu_+^{(t)} = \mathrm{Norm}_\epsilon\left(\frac{1}{|\mathcal{A}_+^{(t)}|}\sum_{j\in\mathcal{A}_+^{(t)}}\bar{z}_j^{(t)}\right), \qquad \mu_-^{(t)} = \mathrm{Norm}_\epsilon\left(\frac{1}{|\mathcal{A}_-^{(t)}|}\sum_{j\in\mathcal{A}_-^{(t)}}\bar{z}_j^{(t)}\right),$$

The anchor-support evidence is

$$r_i^{\mathrm{anc},(t)} = \mathrm{Scale}\left(\left\langle \bar{z}_i^{(t)}, \mu_+^{(t)}\right\rangle - \left\langle \bar{z}_i^{(t)}, \mu_-^{(t)}\right\rangle\right).$$

Here $\mathrm{Scale}(\cdot)$ denotes min–max scaling across all examples in the current round, i.e., $\mathrm{Scale}(s_i) = (s_i - \min_k s_k)/(\max_k s_k - \min_k s_k + \epsilon)$ for scores $\{s_k\}_{k=1}^n$. A larger $r_i^{\mathrm{anc},(t)}$ means that example i is closer to the positive-anchor direction than to the negative-anchor direction.

Trust-state update. The trust module combines encoder evidence, local evidence, anchor evidence, and transport evidence into a single trust logit:

$$u_i^{(t)} = \lambda_p \log\frac{p_i^{(t)}}{1-p_i^{(t)}} + \lambda_{\mathrm{loc}}\left(2r_i^{\mathrm{loc},(t)} - 1\right) + \lambda_{\mathrm{anc}}\left(2r_i^{\mathrm{anc},(t)} - 1\right) + \lambda_m\left(2m_i^{(t)} - 1\right) + \beta_0,$$

where $\lambda_p, \lambda_{\mathrm{loc}}, \lambda_{\mathrm{anc}}, \lambda_m$ weight the four evidence sources, and β_0 is an intercept term controlling the overall conservativeness of the update.

We then apply source and verified constraints:

$$\tilde{q}_i^{(t)} = \begin{cases} 1, & i \in \mathcal{P}_s \cup \mathcal{V}^+, \\ 0, & i \in \mathcal{V}^-, \\ \sigma(u_i^{(t)}), & \text{otherwise}, \end{cases} \qquad a_i^{(t+1)} = \begin{cases} a_s, & i \in \mathcal{P}_s, \\ a_v, & i \in \mathcal{V}^+ \cup \mathcal{V}^-, \\ \tilde{a}_i^{(t+1)}, & \text{otherwise}. \end{cases}$$

where $\tilde{a}_i^{(t+1)} = \Pi_{[0,1]}\left(\gamma_q \tilde{q}_i^{(t)} + \gamma_{\mathrm{loc}} r_i^{\mathrm{loc},(t)}\right)$. Here γ_q and γ_{loc} control the contributions of each parts. The resulting pair $(\tilde{q}^{(t)}, a^{(t+1)})$ is passed to the transport reassignment module, which produces the next responsibility $q^{(t+1)}$.

B.3 Conservative transport-based inflow update

After the trust update, we compute transport-based inflow scores for uncertain unlabeled examples. In this subsection, anchors are indexed by i, while uncertain examples in $\mathcal{M}^{(t)}$ are indexed by j. Here, $\mathcal{M}^{(t)} = \{j \in \mathcal{U} : \tilde{q}_j^{(t)} \in I_{\mathrm{amb}}\}$ denotes the uncertain set, where $I_{\mathrm{amb}} \subset (0,1)$ selects examples with non-decisive responsibilities. Motivated by partial optimal transport, which allows only a budgeted portion of mass to be matched, we use a sparse, budgeted propagation rule rather than solving a full optimal transport problem (Chapel et al., 2020; Peyré & Cuturi, 2019). This gives a positive inflow $m_j^{+,(t)}$ from reliable positive anchors and a negative inflow $m_j^{-,(t)}$ from verified negative anchors. Only $m_j^{+,(t)}$ is carried forward as $m_j^{(t+1)}$; $m_j^{-,(t)}$ is used only in the current reassignment step. At iteration t, the positive and negative anchor pools are

$$\mathcal{B}_+^{(t)} = \mathcal{P}_s \cup \mathcal{V}^+ \cup \{i \in \mathcal{U} : \tilde{q}_i^{(t)} \geq 1/2,\ a_i^{(t+1)} \geq \kappa_t\}, \quad \mathcal{B}_-^{(t)} = \mathcal{V}^-$$

where κ_t is a confidence threshold.

For each uncertain example $j \in \mathcal{M}^{(t)}$, we first compute a local relation-consistency score

$$c_j^{(t)} = \Pi_{[0,1]}\left(1 - \left|\tilde{q}_j^{(t)} - \frac{1}{|\mathcal{N}(j)|}\sum_{r\in\mathcal{N}(j)}\tilde{q}_r^{(t)}\right|\right).$$

This score is large when the trust-updated responsibility of j agrees with its neighbourhood, and it is used to downweight transport into locally inconsistent examples.

We compute the inflows by propagating a bounded amount of mass from positive anchors to uncertain examples. For a positive anchor $i \in \mathcal{B}_+^{(t)}$ and an uncertain example $j \in \mathcal{M}^{(t)}$, we first compute

their latent cosine similarity $S_{ij}^{+,(t)} = \left\langle \bar{z}_i^{(t)}, \bar{z}_j^{(t)} \right\rangle$. For each anchor i, we keep only the most similar uncertain examples and assign them a transport affinity

$$\Gamma_{ij}^{+,(t)} = \frac{\exp(\tau_{\mathrm{tr}} S_{ij}^{+,(t)}) \left(1 - \lambda_{\mathrm{rel}} + \lambda_{\mathrm{rel}} c_j^{(t)}\right)}{\sum_{j'} \exp(\tau_{\mathrm{tr}} S_{ij'}^{+,(t)}) \left(1 - \lambda_{\mathrm{rel}} + \lambda_{\mathrm{rel}} c_{j'}^{(t)}\right) + \epsilon},$$

where the sum is over the retained uncertain examples for anchor i. Thus, $\Gamma_{ij}^{+,(t)}$ is large when j is close to i in latent space and is locally consistent with its neighbors. The parameter τ_{tr} controls how strongly transport favors the most similar targets, while λ_{rel} controls the contribution of local relation consistency. We assign source masses separately for positive and negative anchors:

$$\pi_i^{+,(t)} = \frac{a_i^{(t+1)}}{\sum_{r \in \mathcal{B}_+^{(t)}} a_r^{(t+1)} + \epsilon}, \quad i \in \mathcal{B}_+^{(t)}, \qquad \pi_i^{-,(t)} = \frac{1}{|\mathcal{B}_-^{(t)}| + \epsilon}, \quad i \in \mathcal{B}_-^{(t)}.$$

Positive anchors are weighted by anchor confidence, while negative anchors are weighted uniformly. The negative affinity $\Gamma_{ij}^{-,(t)}$ is defined analogously by replacing $\mathcal{B}_+^{(t)}$ with $\mathcal{B}_-^{(t)}$. Given budgets B_t^+ and B_t^-, we aggregate anchor-to-target transported mass:

$$m_j^{+,(t)} = \sum_{i \in \mathcal{B}_+^{(t)}} B_t^+ \, \pi_i^{+,(t)} \Gamma_{ij}^{+,(t)}, \quad m_j^{-,(t)} = \sum_{i \in \mathcal{B}_-^{(t)}} B_t^- \, \pi_i^{-,(t)} \Gamma_{ij}^{-,(t)}.$$

The positive inflow $m_j^{+,(t)}$ supports increasing the responsibility of j, whereas the negative inflow $m_j^{-,(t)}$ supports decreasing it. Thus, $m_j^{+,(t)}$ pushes j toward the positive group, while $m_j^{-,(t)}$ pushes it away from the positive group.

Finally, the responsibility is updated by applying the two inflows in opposite directions:

$$q_j^{+,(t)} = \tilde{q}_j^{(t)} + \eta_+ m_j^{+,(t)} \left(1 - \tilde{q}_j^{(t)}\right), \qquad q_j^{(t+1)} = q_j^{+,(t)} - \eta_- m_j^{-,(t)} q_j^{+,(t)}.$$

The first update increases responsibility using positive inflow, while the second decreases it using negative inflow. For examples not in $\mathcal{M}^{(t)}$, we keep $q_j^{(t+1)} = \tilde{q}_j^{(t)}$.

Notably, only the positive inflow is carried forward to the next iteration, i.e., $m_j^{(t+1)} = m_j^{+,(t)}$, while the negative inflow $m_j^{-,(t)}$ is used only within the current transport update.

B.4 Selective verification and stopping details

After transport update, the candidate set is

$$\mathcal{C}^{(t)} = \mathcal{U} \setminus \left(\mathcal{V}^+ \cup \mathcal{V}^- \cup \mathcal{W}^{(t)}\right),$$

where $\mathcal{W}^{(t)}$ contains examples already selected for human review but whose labels have not yet been incorporated.

For each candidate $i \in \mathcal{C}^{(t)}$, we compute

$$U_i^{(t)} = 1 - \left|2q_i^{(t+1)} - 1\right|, \quad I_i^{(t)} = p_i^{(t)}(1 - p_i^{(t)})(\|z_i^{(t)}\|^2 + 1), \quad D_i^{(t)} = 1 - \max_{r \in \mathcal{V}^{(t)}} \langle \bar{z}_i^{(t)}, \bar{z}_r^{(t)} \rangle.$$

Here $\mathcal{V}^{(t)} = \mathcal{V}^+ \cup \mathcal{V}^-$ denotes the currently verified set, and we set $D_i^{(t)} = 1$ when $\mathcal{V}^{(t)} = \varnothing$. The final query score is

$$H_i^{(t)} = \omega_U \, \mathrm{Scale}\left(U_i^{(t)}\right) + \omega_I \, \mathrm{Scale}\left(I_i^{(t)}\right) + \omega_D \, \mathrm{Scale}\left(D_i^{(t)}\right) + \omega_M \, \mathrm{Scale}\left(m_i^{(t+1)}\right),$$

where $\mathrm{Scale}(\cdot)$ is computed over $\mathcal{C}^{(t)}$. We select the top b candidates with the largest $H_i^{(t)}$ for human verification. If the returned human label agrees with the LLM judge, the example is added to $\mathcal{V}^+$; otherwise, it is added to $\mathcal{V}^-$. If the label is not yet available, the example is kept in $\mathcal{W}^{(t+1)}$.

For stopping, we monitor

$$\Delta_q^{(t)} = \frac{1}{n}\sum_{i=1}^{n} |q_i^{(t+1)} - q_i^{(t)}|,$$

$$\Delta_{\text{flip}}^{(t)} = \frac{1}{n}\sum_{i=1}^{n} \mathbf{1}\left\{\mathbf{1}(q_i^{(t+1)} \geq 1/2) \neq \mathbf{1}(q_i^{(t)} \geq 1/2)\right\},$$

$$\Delta_m^{(t)} = \frac{1}{n}\sum_{i=1}^{n} |m_i^{(t+1)} - m_i^{(t)}|, \qquad \Delta_{\text{ver}}^{(t)} = |\mathcal{V}^{(t+1)}| - |\mathcal{V}^{(t)}|.$$

The rule-based stopping criterion checks

$$\Delta_q^{(t)} \leq \epsilon_q, \qquad \Delta_{\text{flip}}^{(t)} \leq \epsilon_{\text{flip}}, \qquad \Delta_m^{(t)} \leq \epsilon_m, \qquad \Delta_{\text{ver}}^{(t)} \leq \epsilon_{\text{ver}}.$$

The algorithm stops when at least three of the four conditions hold for the required number of consecutive rounds, after a minimum number of iterations. The loop also terminates when the maximum iteration count or annotation budget is reached.

C Additional theoretical results

C.1 Uncertainty-targeted transport

Assumption 2 (Ambiguous transport targets)**.** *For some $\ell \in (0, 1/2]$, $I_{\text{amb}} \subseteq [\ell, 1-\ell]$.*

Corollary 1 (Allowed error in incoming evidence)**.** *Under Assumption 2, if $\sum_j \theta_j^{(t)} > 0$ and*

$$\bar{\theta}^{(t)} = \frac{\sum_j (\theta_j^{(t)})^2}{\sum_j \theta_j^{(t)}},$$

then

$$\|q^{(t+1)} - y\|_2^2 - \|\tilde{q}^{(t)} - y\|_2^2 \leq -\ell^2(2 - \bar{\theta}^{(t)}) \sum_j \theta_j^{(t)} + 2(1-\ell)\sum_j h_j^{(t)}. \tag{10}$$

In particular, the total squared error decreases strictly if

$$\frac{\sum_j h_j^{(t)}}{\sum_j \theta_j^{(t)}} < \frac{\ell^2(2 - \bar{\theta}^{(t)})}{2(1-\ell)}. \tag{11}$$

C.2 Source confidence and latent transport geometry

Let

$$A_t^+ = \sum_{i,j} T_{ij}^{+,(t)}(1 - y_i), \qquad A_t^- = \sum_{i,j} T_{ij}^{-,(t)} y_i,$$

and

$$X_t^\nu = \sum_{i,j} T_{ij}^{\nu,(t)} \mathbf{1}\{y_i \neq y_j\}, \qquad \nu \in \{+, -\}.$$

Proposition 1 (Reference and neighborhood contributions)**.** *The incorrect incoming transport satisfies*

$$\sum_j h_j^{(t)} \leq \eta_+(A_t^+ + X_t^+) + \eta_-(A_t^- + X_t^-). \tag{12}$$

Assumption 3 (Local label regularity on retained edges)**.** *There exists $L_{\text{loc}} > 0$ such that*

$$|y_i - y_j| \leq L_{\text{loc}} \|\bar{z}_i^{(t)} - \bar{z}_j^{(t)}\|_2$$

whenever $T_{ij}^{+,(t)} + T_{ij}^{-,(t)} > 0$.

Proposition 2 (Geometry controls cross-label transport)**.** *Define*

$$D_t^\nu = \sum_{i,j} T_{ij}^{\nu,(t)} \|\bar{z}_i^{(t)} - \bar{z}_j^{(t)}\|_2.$$

Under Assumption 3,

$$X_t^\nu \le L_{\mathrm{loc}} D_t^\nu. \tag{13}$$

Moreover,

$$A_t^+ \le B_t^+ \sum_{i \in \mathcal{B}_+^{(t)} \cap \mathcal{N}^\star} \pi_i^{+,(t)}, \qquad A_t^- \le B_t^- \sum_{i \in \mathcal{B}_-^{(t)} \cap \mathcal{P}^\star} \pi_i^{-,(t)}.$$

C.3 Transport perturbations and decision stability

Let $\overline{T}^{\nu,(t)}$ be another nonnegative transport matrix on the same comparison pool. The original update satisfies the transport conditions used in Theorem 2. Both updates use the same $\tilde{q}^{(t)}$ and the same step sizes η_+, η_-. Define $\overline{m}_j^{\nu,(t)} = \sum_i \overline{T}_{ij}^{\nu,(t)}$, and let $\overline{q}^{(t+1)}$ be obtained from (1) using these alternative inflows. Define

$$\Delta_t = \eta_+ \sum_{i,j} |T_{ij}^{+,(t)} - \overline{T}_{ij}^{+,(t)}| + \eta_- \sum_{i,j} |T_{ij}^{-,(t)} - \overline{T}_{ij}^{-,(t)}|.$$

Proposition 3 (Decision stability)**.** *Suppose both $T^{\nu,(t)}$ and $\overline{T}^{\nu,(t)}$ satisfy the same pointwise update-size restrictions,*

$$0 \le \eta_+ m_j^{+,(t)},\ \eta_- m_j^{-,(t)},\ \eta_+ \overline{m}_j^{+,(t)},\ \eta_- \overline{m}_j^{-,(t)} \le 1.$$

Then

$$\|q^{(t+1)} - \overline{q}^{(t+1)}\|_1 \le \Delta_t.$$

If $\widehat{\mathcal{P}}(q) = \{j : q_j \ge 1/2\}$, then for every $\delta > 0$,

$$|\widehat{\mathcal{P}}(q^{(t+1)}) \triangle \widehat{\mathcal{P}}(\overline{q}^{(t+1)})| \le |\{j : |\overline{q}_j^{(t+1)} - 1/2| \le \delta\}| + \frac{\Delta_t}{\delta}. \tag{14}$$

C.4 Finite round error propagation

Define $d_j^{(t)} = |\tilde{q}_j^{(t)} - q_j^{(t)}|$.

Theorem 3 (Finite round error propagation)**.** *Suppose the one-step transport conditions of Theorem 2 hold at each round $t = 0, \dots, T-1$. Then, for every $T \ge 1$,*

$$\begin{aligned} |q_j^{(T)} - y_j| \le |q_j^{(0)} - y_j| \prod_{t=0}^{T-1} (1 - \theta_j^{(t)}) \\ + \sum_{t=0}^{T-1} \Big[(1 - \theta_j^{(t)}) d_j^{(t)} + h_j^{(t)} \Big] \prod_{s=t+1}^{T-1} (1 - \theta_j^{(s)}). \end{aligned} \tag{15}$$

Assumption 4 (Bounds on incoming errors and score changes)**.** *Consider refinement rounds $t = 0, 1, \dots$ with fixed labels $y \in \{0,1\}^n$ and $q^{(0)} \in [0,1]^n$. At every round, $\tilde{q}^{(t)} \in [0,1]^n$ and the update* (1) *is applied, with $0 \le \eta_\nu m_i^{\nu,(t)} \le 1$ for every sample i and both signs $\nu \in \{+,-\}$.*

For the sample j under study, there are finite constants $\varepsilon_j, \delta_j \ge 0$, independent of t, such that

$$h_j^{(t)} \le \theta_j^{(t)} \varepsilon_j, \qquad (1 - \theta_j^{(t)}) d_j^{(t)} \le \theta_j^{(t)} \delta_j$$

at every round.

Corollary 2 (Long-term error bound)**.** *Under Assumption 4, if $\sum_{t=0}^{\infty} \theta_j^{(t)} = \infty$, then $\limsup_{T \to \infty} |q_j^{(T)} - y_j| \le \varepsilon_j + \delta_j$.*

D PROOFS OF THE THEORETICAL RESULTS

Proof of Theorem 1. For unconstrained coordinates, $\tilde{q}_i = \sigma(u_i)$. Since $\sup_u \sigma'(u) = 1/4$ and, on $[\epsilon_p, 1-\epsilon_p]$,

$$\left|\frac{d}{dp}\log\frac{p}{1-p}\right| = \frac{1}{p(1-p)} \le \frac{1}{\epsilon_p(1-\epsilon_p)},$$

the trust logit is Lipschitz in all its arguments. In particular, when $(p, r^{\mathrm{loc}}, r^{\mathrm{anc}})$ are fixed,

$$|u_i(m) - u_i(m')| = 2|\lambda_m||m_i - m_i'|,$$

and therefore

$$|\tilde{q}_i(m) - \tilde{q}_i(m')| \le \frac{|\lambda_m|}{2}|m_i - m_i'|.$$

Summing over the free coordinates proves (3); hard-constrained coordinates contribute zero. □

Proof of Lemma 1. Let $L(\Theta) = \mathcal{L}_{\mathrm{enc}}^{(t)}(\Theta)$ for a fixed outer round. By L_Θ-smoothness,

$$L(\Theta - \eta\nabla L(\Theta)) \le L(\Theta) - \eta\left(1 - \frac{L_\Theta\eta}{2}\right)\|\nabla L(\Theta)\|_2^2.$$

The coefficient is positive for $0 < \eta < 2/L_\Theta$, which gives (4). □

Proof of Theorem 2. Write $a_j = \eta_+ m_j^{+,(t)}$ and $b_j = \eta_- m_j^{-,(t)}$. Then $a_j, b_j \in [0,1]$ and

$$q_j^{(t+1)} = (1-b_j)\big[(1-a_j)\tilde{q}_j^{(t)} + a_j\big] = (1-\theta_j^{(t)})\tilde{q}_j^{(t)} + (1-b_j)a_j,$$

where $\theta_j^{(t)} = a_j + b_j - a_j b_j$.

If $y_j = 0$, then $h_j^{(t)} = (1-b_j)a_j$, hence

$$|q_j^{(t+1)} - y_j| = (1-\theta_j^{(t)})|\tilde{q}_j^{(t)} - y_j| + h_j^{(t)}.$$

If $y_j = 1$, then $h_j^{(t)} = b_j$ and

$$1 - q_j^{(t+1)} = (1-\theta_j^{(t)})(1-\tilde{q}_j^{(t)}) + b_j,$$

so the same identity holds. Moreover $0 \le h_j^{(t)} \le \theta_j^{(t)}$.

Subtracting the pre-transport absolute error yields

$$|q_j^{(t+1)} - y_j| - |\tilde{q}_j^{(t)} - y_j| = h_j^{(t)} - \theta_j^{(t)}|\tilde{q}_j^{(t)} - y_j|,$$

which proves (8). Squaring (7) gives

$$\begin{aligned}&(q_j^{(t+1)} - y_j)^2 - (\tilde{q}_j^{(t)} - y_j)^2 \\ &\quad = -\theta_j^{(t)}(2-\theta_j^{(t)})(\tilde{q}_j^{(t)} - y_j)^2 + 2(1-\theta_j^{(t)})|\tilde{q}_j^{(t)} - y_j|h_j^{(t)} + (h_j^{(t)})^2.\end{aligned}$$

Summing over j and using $|\tilde{q}_j^{(t)} - y_j| \le 1$ and $h_j^{(t)} \le \theta_j^{(t)}$ yields

$$\|q^{(t+1)} - y\|_2^2 - \|\tilde{q}^{(t)} - y\|_2^2 \le -G^{(t)} + 2\sum_j h_j^{(t)},$$

which is (9). □

Proof of Proposition 1. Define

$$W_{ij}^{+,(t)} = \eta_+(1-\eta_- m_j^{-,(t)})T_{ij}^{+,(t)}, \qquad W_{ij}^{-,(t)} = \eta_- T_{ij}^{-,(t)}.$$

Then

$$\sum_j h_j^{(t)} = \sum_{i,j} W_{ij}^{+,(t)}(1-y_j) + \sum_{i,j} W_{ij}^{-,(t)} y_j.$$

For binary labels,

$$1 - y_j \le (1 - y_i) + \mathbf{1}\{y_i \neq y_j\}, \qquad y_j \le y_i + \mathbf{1}\{y_i \neq y_j\}.$$

Since $W_{ij}^{+,(t)} \le \eta_+ T_{ij}^{+,(t)}$ and $W_{ij}^{-,(t)} = \eta_- T_{ij}^{-,(t)}$, summation gives

$$\sum_j h_j^{(t)} \le \eta_+(A_t^+ + X_t^+) + \eta_-(A_t^- + X_t^-),$$

which proves (12). □

Proof of Corollary 1. If $\theta_j^{(t)} > 0$, then j receives nonzero transport and hence belongs to the uncertain target set. Assumption 2 gives

$$\ell \le |\tilde{q}_j^{(t)} - y_j| \le 1 - \ell.$$

Therefore

$$G^{(t)} \ge \ell^2 \sum_j \theta_j^{(t)}(2 - \theta_j^{(t)}) = \ell^2(2 - \bar{\theta}^{(t)}) \sum_j \theta_j^{(t)}.$$

Also, because $h_j^{(t)} \le \theta_j^{(t)}$,

$$2(1 - \theta_j^{(t)})|\tilde{q}_j^{(t)} - y_j| h_j^{(t)} + (h_j^{(t)})^2 \le 2(1 - \ell) h_j^{(t)}.$$

Substitution into the exact squared-error expansion in Theorem 2 proves (10). Since $\sum_j \theta_j^{(t)} > 0$ and $1 - \ell > 0$, (11) makes the right side strictly negative. □

Proof of Proposition 2. Using $T_{ij}^{\nu,(t)} = B_t^\nu \pi_i^{\nu,(t)} \Gamma_{ij}^{\nu,(t)}$ and $\sum_j \Gamma_{ij}^{\nu,(t)} \le 1$,

$$A_t^+ \le B_t^+ \sum_{i \in \mathcal{B}_+^{(t)} \cap \mathcal{N}^\star} \pi_i^{+,(t)}, \qquad A_t^- \le B_t^- \sum_{i \in \mathcal{B}_-^{(t)} \cap \mathcal{P}^\star} \pi_i^{-,(t)}.$$

Since $y_i, y_j \in \{0, 1\}$, $\mathbf{1}\{y_i \neq y_j\} = |y_i - y_j|$. Thus Assumption 3 implies

$$X_t^\nu = \sum_{i,j} T_{ij}^{\nu,(t)} |y_i - y_j| \le L_{\mathrm{loc}} \sum_{i,j} T_{ij}^{\nu,(t)} \|\bar{z}_i^{(t)} - \bar{z}_j^{(t)}\|_2 = L_{\mathrm{loc}} D_t^\nu.$$

□

Proof of Proposition 3. For fixed $x \in [0, 1]$, set

$$f(x, \alpha, \beta) = (1 - \beta)\big[x + (1 - x)\alpha\big].$$

For $\alpha, \beta, \bar{\alpha}, \bar{\beta} \in [0, 1]$,

$$f(x, \alpha, \beta) - f(x, \bar{\alpha}, \bar{\beta}) = (1 - \beta)(1 - x)(\alpha - \bar{\alpha}) + (\bar{\beta} - \beta)\big[x + (1 - x)\bar{\alpha}\big],$$

hence

$$|f(x, \alpha, \beta) - f(x, \bar{\alpha}, \bar{\beta})| \le |\alpha - \bar{\alpha}| + |\beta - \bar{\beta}|.$$

Applying this coordinatewise with $\alpha = \eta_+ m_j^{+,(t)}$ and $\beta = \eta_- m_j^{-,(t)}$ gives

$$\|q^{(t+1)} - \overline{q}^{(t+1)}\|_1 \le \Delta_t.$$

If the two thresholded decisions differ at index j while $|\overline{q}_j^{(t+1)} - 1/2| > \delta$, then necessarily $|q_j^{(t+1)} - \overline{q}_j^{(t+1)}| > \delta$. Therefore at most Δ_t/δ such indices can occur, and adding those already within δ of the threshold proves (14). □

Proof of Theorem 3. Theorem 2 and the triangle inequality give

$$|q_j^{(t+1)} - y_j| \le (1-\theta_j^{(t)})|q_j^{(t)} - y_j| + (1-\theta_j^{(t)})d_j^{(t)} + h_j^{(t)}.$$

Iterating this recursion yields

$$|q_j^{(T)} - y_j| \le |q_j^{(0)} - y_j| \prod_{t=0}^{T-1}(1-\theta_j^{(t)}) + \sum_{t=0}^{T-1}\left[(1-\theta_j^{(t)})d_j^{(t)} + h_j^{(t)}\right]\prod_{s=t+1}^{T-1}(1-\theta_j^{(s)}),$$

which is (15). □

Proof of Corollary 2. The one-step error identity and the triangle inequality give

$$\begin{aligned}|q_j^{(t+1)} - y_j| &= (1-\theta_j^{(t)})|\tilde{q}_j^{(t)} - y_j| + h_j^{(t)} \\ &\le (1-\theta_j^{(t)})|q_j^{(t)} - y_j| + (1-\theta_j^{(t)})d_j^{(t)} + h_j^{(t)}.\end{aligned}$$

By Assumption 4, we have

$$|q_j^{(t+1)} - y_j| \le (1-\theta_j^{(t)})|q_j^{(t)} - y_j| + \theta_j^{(t)}(\varepsilon_j + \delta_j).$$

Subtracting $\varepsilon_j + \delta_j$ from both sides yields the inequality

$$|q_j^{(t+1)} - y_j| - (\varepsilon_j + \delta_j) \le (1-\theta_j^{(t)})\left[|q_j^{(t)} - y_j| - (\varepsilon_j + \delta_j)\right].$$

Since $0 \le 1 - \theta_j^{(t)} \le 1$, repeated application of this inequality gives, for every integer $T \ge 1$,

$$|q_j^{(T)} - y_j| - (\varepsilon_j + \delta_j) \le \left[\prod_{t=0}^{T-1}(1-\theta_j^{(t)})\right]\left[|q_j^{(0)} - y_j| - (\varepsilon_j + \delta_j)\right].$$

For $x \in [0,1]$, it is clear that $1 - x \le e^{-x}$. Hence

$$0 \le \prod_{t=0}^{T-1}(1-\theta_j^{(t)}) \le \exp\left(-\sum_{t=0}^{T-1}\theta_j^{(t)}\right).$$

The assumed divergence of $\sum_{t=0}^{\infty}\theta_j^{(t)}$ therefore implies that the product tends to zero. Taking the upper limit in the preceding error bound gives the result $\limsup_{T\to\infty}|q_j^{(T)} - y_j| \le \varepsilon_j + \delta_j$. □

E Additional experimental details

E.1 Evaluation metrics

We use the same set of metrics for simulation and real-data evaluation whenever applicable. The *original accuracy* measures the agreement between the initial judge-induced consistency signal and the ground-truth consistency label. In simulation, this label is generated by the simulator; in real data, it is derived from human preference annotations. The *adjusted accuracy* measures the agreement after applying AURA.

We define the accuracy difference as

$$\Delta_{\mathrm{acc}} = \mathrm{Acc}_{\mathrm{adj}} - \mathrm{Acc}_{\mathrm{orig}}.$$

A positive value indicates that the refinement improves agreement with the target human-consistency label. We also report the *flip rate*, defined as the fraction of examples whose hard assignment changes after refinement:

$$\mathrm{Flip} = \frac{1}{n}\sum_{i=1}^{n}\mathbf{1}\{\widehat{y}_i^{\mathrm{adj}} \ne \widehat{y}_i^{\mathrm{orig}}\}.$$

This metric measures how actively the method changes the initial signal. A useful refinement method should not only flip many labels, but should flip them in a way that improves adjusted accuracy.

For real-data experiments, we additionally report the final waiting-pool size, runtime, and peak memory usage. The waiting-pool size records the number of examples selected for verification but not yet incorporated into the refinement state at termination. Runtime and memory are used to quantify the computational cost of applying AURA to real LLM-as-a-judge data.

E.2 Simulation details

We construct a generative simulation that mirrors the refinement problem studied in Sections 3–4. Specifically, the simulator generates examples from two latent consistency groups with a adjustable separation, and then produces noisy comparison features and corrupted initial consistency signals. AURA only observes these noisy features and corrupted signals. This setup allows us to evaluate whether the proposed trust and transport updates can recover the underlying consistency structure from the information available to the refinement procedure.

For each simulated example, we first sample a binary consistency label $y_i \in \{0, 1\}$, following the definition in Section 3. We then generate a simulator-side latent coordinate $\boldsymbol{h}_i \in \mathbb{R}^d$, which is used only for data generation and is not observed by AURA. Conditional on $y_i = k$, we sample

$$\boldsymbol{h}_i \mid y_i = k \sim \mathcal{N}_d(\boldsymbol{\mu}_k, \boldsymbol{\Sigma}), \qquad k \in \{0, 1\},$$

where $\boldsymbol{\mu}_k \in \mathbb{R}^d$ is the group mean vector for label k, and $\boldsymbol{\Sigma} \in \mathbb{R}^{d\times d}$ is the covariance matrix. The distance between $\boldsymbol{\mu}_0$ and $\boldsymbol{\mu}_1$ controls how separable the human-consistent and human-inconsistent groups are.

The observed comparison feature is identified directly with the latent coordinate, $x_i = \boldsymbol{h}_i$; the covariance $\boldsymbol{\Sigma}$ already plays the combined role of geometric spread and observation noise, so no separate linear map or additive noise term is needed. Thus, x_i plays the same role as the fixed comparison feature in the real pipeline: it contains useful geometric information about consistency, but the conditional distributions of $x_i \mid y_i = 1$ and $x_i \mid y_i = 0$ remain noisy and overlapping rather than perfectly separable.

We then corrupt y_i to obtain an initial noisy consistency signal $\tilde{y}_i \in \{0, 1\}$, which plays the role of the raw LLM-induced signal before refinement. We consider two corruption mechanisms, following the common distinction between class-conditional and instance-dependent label-noise models (Xia et al., 2020; Berthon et al., 2021; Cheng et al., 2022). In the *class-dependent* (CD) setting, the corruption probability depends only on the true consistency label:

$$\mathbb{P}(\tilde{y}_i \neq y_i \mid y_i, x_i) = \mathbb{P}(\tilde{y}_i \neq y_i \mid y_i).$$

Thus, all examples with the same value of y_i share the same corruption rate, regardless of their location in the feature space. This provides a controlled baseline for testing whether AURA can correct label-level corruption.

In the *distribution-dependent* (DD) setting, the corruption probability is allowed to depend on the sample location. Writing $s(x_i) = (x_i - \bar{\boldsymbol{\mu}})^\top \boldsymbol{v}$ for the signed distance of x_i from the midpoint $\bar{\boldsymbol{\mu}} = (\boldsymbol{\mu}_0 + \boldsymbol{\mu}_1)/2$ along the unit axis $\boldsymbol{v} = (\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0)/\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\|$, we set

$$\mathbb{P}(\tilde{y}_i \neq y_i \mid y_i, x_i) = \mathrm{clip}\left(\sigma\big(-\gamma|s(x_i)|\big) \cdot \frac{\rho}{\sigma(-\gamma|s(\cdot)|)}, 0, 0.95\right),$$

where σ is the logistic function, $\gamma > 0$ controls how sharply corruption concentrates near the class boundary, ρ is the target corruption rate, and the denominator rescales the batch so its average matches ρ, the same nominal rate used in the CD setting. Examples near the midpoint between the two groups are corrupted with higher probability, and examples far from it with lower probability, reflecting that judge errors concentrate on ambiguous, boundary-region comparisons rather than being uniform within each class.

Finally, we construct the verified sets $\mathcal{V}^+$ and $\mathcal{V}^-$ by revealing only a subset of labels to the refinement procedure. We consider two verification mechanisms. The first is *selected completely at random* (SCAR), under which the probability that an example is human-verified is a constant ν

that does not depend on its features. The second is *selected at random* (SAR), under which the verification probability instead decreases with distance from the midpoint $\bar{\boldsymbol{\mu}}$, $\mathbb{P}(\text{verified} \mid x_i) = \text{clip}\Big(\frac{1}{1+\|x_i-\bar{\boldsymbol{\mu}}\|} \cdot \frac{\nu}{(1+\|\cdot-\bar{\boldsymbol{\mu}}\|)^{-1}},\, 0,\, 0.9\Big)$, again rescaled to match the same average rate ν as SCAR. Under SAR, verification is therefore biased toward examples near the decision boundary, better reflecting selective human review than uniform sampling. Human resolution is further restricted to a random κ fraction of the unlabeled pool (an Elkan–Noto-style limited-oracle constraint (Elkan & Noto, 2008)); queries outside this fraction enter the waiting pool rather than resolving immediately.

Unless otherwise specified, we set the latent dimension $d = 2$, class-centre distance $\|\boldsymbol{\mu}_1 - \boldsymbol{\mu}_0\| = 4$, covariance $\boldsymbol{\Sigma} = \left[\begin{smallmatrix} 1.5 & 0.3 \\ 0.3 & 1.5 \end{smallmatrix}\right]$, corruption sharpness $\gamma = 1.2$, target corruption rate $\rho = 0.30$, initial verification rate $\nu = 0.15$, and library fraction $\kappa = 0.40$. Each run has 160 trusted seed-positive examples, 168 hidden-positive and 312 negative unlabeled examples (640 total), and downstream query selection uses the top 15 candidates by query score per round. Each noise–verification configuration is repeated over 5 independent runs. We report four metrics: original accuracy, the accuracy of the corrupted initial signal before refinement; adjusted accuracy, the accuracy after applying AURA; accuracy difference, the adjusted accuracy minus the original accuracy; and flip rate, the fraction of examples whose hard assignment changes after refinement.

E.3 Simulation sensitivity to latent geometry

We further examine how the latent geometry affects refinement, repeating the sweep across all four noise-and-verification settings (CD/DD $\times$ SCAR/SAR). We assess sensitivity to two geometric factors: the class-center distance $\|\mu_1 - \mu_0\|$, which controls how separable the two consistency groups are, and the covariance scale, which controls how much the two groups overlap for a fixed distance. Each configuration is repeated over 5 independent runs, and we report original accuracy, adjusted accuracy, accuracy gain, and hidden-positive recall (the fraction of held-out hidden positive examples, i.e. excluding the trusted seed set, correctly recovered as positive).

We first vary the class-center distance from 1 to 10 while holding the covariance at its default scale (Figure 4 shows DD+SAR in detail; Figure 6 compares the accuracy gain across all four settings). At distance 1, the two groups are highly overlapping and refinement is actively harmful, giving a negative gain, since local-neighborhood and anchor evidence are unreliable when the class regions overlap. As the distance increases from 2 to 4, refinement becomes clearly beneficial, with the gain peaking around distance 3–4 in every setting. Beyond this range, the original signal's own accuracy keeps rising, because the corruption mechanism itself becomes less confusing at large separations, leaving progressively less room for correction; the gain declines accordingly, turning slightly negative again by distance 9–10 under DD, while staying near its peak under CD, whose corruption mechanism does not similarly resolve at large distances.

We then vary the covariance scale from 0.25 to 4 while holding the class-center distance fixed at 4 (Figure 5 shows DD+SAR in detail; Figure 7 compares all four settings). The gain decreases monotonically as the covariance scale increases in every setting: tighter, more separable groups yield the largest improvement, while the most diffuse setting again yields a small negative gain under DD and a small positive gain under CD. Together, these sweeps indicate that AURA is most effective in a middle regime of latent geometry, where the two consistency groups are neither so overlapping that local geometry is uninformative, nor so separated that little uncertainty remains to resolve, and that this qualitative pattern is not an artifact of any single noise or verification mechanism.

E.4 Hyperparameter sensitivity

AURA is organized around several hyperparameters that control the relabeling threshold, the pseudo-label weight, the transport fraction, the anchor fraction, the human-query top-K, and the number of stable rounds used for stopping. We assess their sensitivity on Mistral-7B-Instruct-v0.3 at the nominal 20% verification setting, varying one hyperparameter at a time while holding the others at their default values, and averaging accuracy over three seeds.

The fixed transport fraction has by far the largest effect (a 6–9 percentage-point spread across the tested range) because it directly controls how much evidence the transport operator propagates from anchors to uncertain examples; forcing it away from the default dynamically-estimated schedule substantially degrades accuracy. Query top-K also has a moderate effect (4 percentage points),

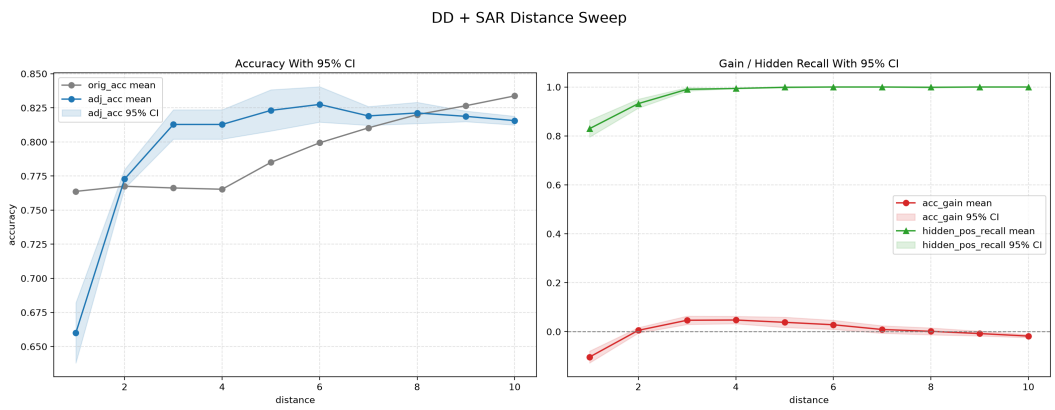

Figure 4: Sensitivity of AURA to the class-center distance under DD+SAR, holding the covariance scale fixed at its default value. Shaded bands show 95% confidence intervals over five seeds per distance value.

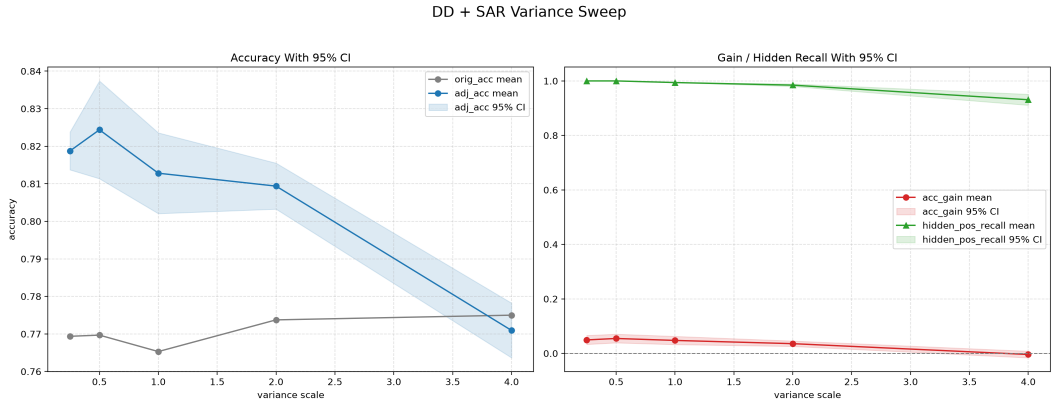

Figure 5: Sensitivity of AURA to the latent covariance scale under DD+SAR, holding the class-center distance fixed at 4. Shaded bands show 95% confidence intervals over five seeds per covariance scale.

since it controls how many candidates are queried each round and therefore how much of the adaptive verification budget is actually spent (Section 6.2). The remaining hyperparameters (relabeling threshold, pseudo-label weight, anchor fraction, stopping rounds) have comparatively small effects, within about 1.5 percentage points, across the tested ranges. Note that the "Fixed transport fraction" row uses a diagnostic ablation that disables AURA's dynamic transport-fraction estimator; in the default method, this fraction is re-estimated at each iteration rather than fixed.

E.5 Real-data details

We evaluate AURA on real LLM-as-a-judge data constructed from MT-Bench and Chatbot Arena (Zheng et al., 2023; Chiang et al., 2024). For the main objective-data experiments, we sample pairwise comparisons from three question types: coding, math reasoning, and structured factual questions. We additionally include one MT-Bench reference setting, where examples are sampled from MT-Bench without question-type stratification.

We consider five judge models, including three API-based models and two locally deployed open-weight models. The API judges are GPT-5.4, GPT-5.4-mini, and Gemini-2.5-Flash (DeepMind, 2024). The local judges are Qwen2.5-7B-Instruct (Team, 2024) and Mistral-7B-Instruct-v0.3 (Jiang et al., 2023). For each judge, we use its preference prediction as the initial LLM-induced signal and evaluate the refined output against available human preference labels.

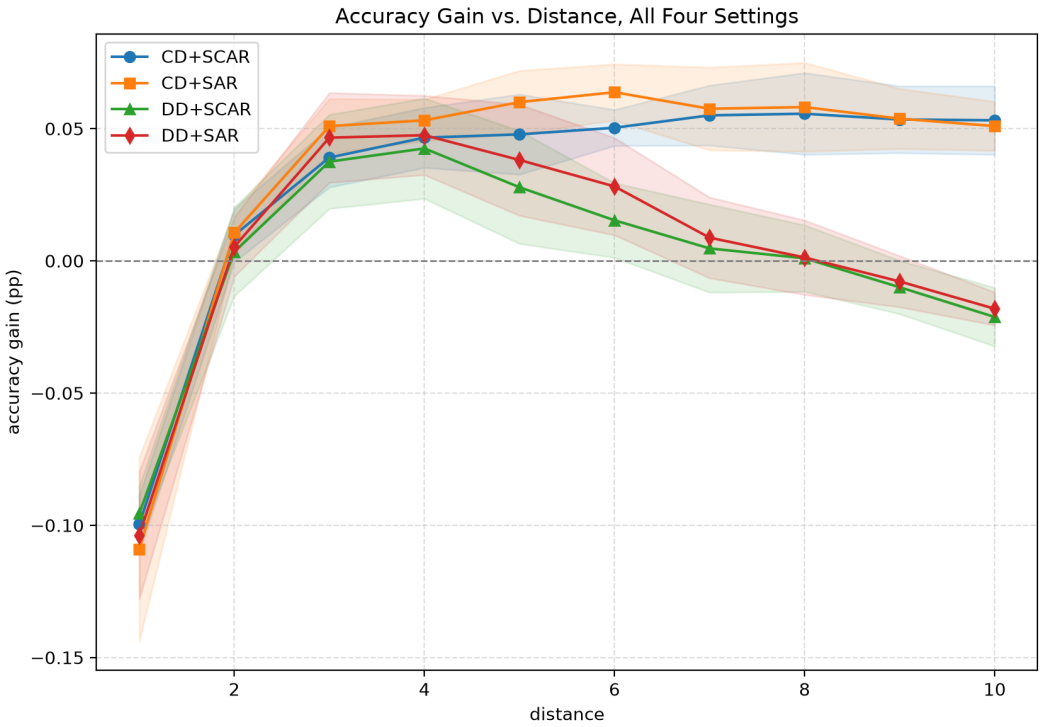

Figure 6: Accuracy gain versus class-center distance, compared across all four noise-and-verification settings. The peak-around-distance-3–4 pattern holds in every setting.

Table 5: Hyperparameter sensitivity on MISTRAL-7B-INSTRUCT-V0.3 at the nominal 20% verification setting. Accuracy (%) is averaged over three seeds; values within each cell follow the order of the corresponding entries in "Values Tested."

Hyperparameter Sensitivity (20% Verification)			
Hyperparameter	**Values Tested**	**Math Acc. (%)**	**Factual Acc. (%)**
Relabeling threshold	0.30 / 0.45 / 0.60	64.7 / 64.4 / 64.4	66.3 / 65.3 / 65.1
Pseudo-label weight	0 / 0.1 / 0.35 / 0.6	64.2 / 63.7 / 64.4 / 64.3	65.4 / 65.8 / 65.3 / 66.1
Fixed transport fraction	0.5 / 0.7 / 0.9 / 0.95	63.9 / 61.7 / 58.1 / 57.5	65.5 / 63.4 / 58.3 / 56.8
Anchor fraction	0.10 / 0.25 / 0.40	65.6 / 64.4 / 64.8	64.8 / 65.3 / 65.9
Query top-K	10 / 50 / 100	61.8 / 64.4 / 65.8	61.8 / 65.3 / 66.1
Stopping rounds	1 / 2 / 3	63.8 / 64.4 / 65.2	65.0 / 65.3 / 65.5

The main real data evaluation consists of 5 judge models $\times$ 3 question types, together with the additional MT-Bench reference setting. Each configuration is repeated over five independent runs. Comparison features are extracted using SKYWORK-REWARD-LLAMA-3.1-8B-V0.2 (Skywork AI, 2024), which is kept fixed throughout refinement. We evaluate two encoder configurations, using latent dimensions 16 and 128, to examine whether the refinement behaviour is sensitive to representation capacity. We provide a representative hyperparameter configuration for a standard MT-Bench run in Table 6, corresponding to the actual setup used in that experiment.

We also compare against four lightweight baselines: logistic regression, MLP classifier, random forest (Breiman, 2001), and label propagation (Zhu & Ghahramani, 2002). These baselines use the same comparison features and verified examples, but do not use the full trust-state update or conservative transport mechanism of AURA.

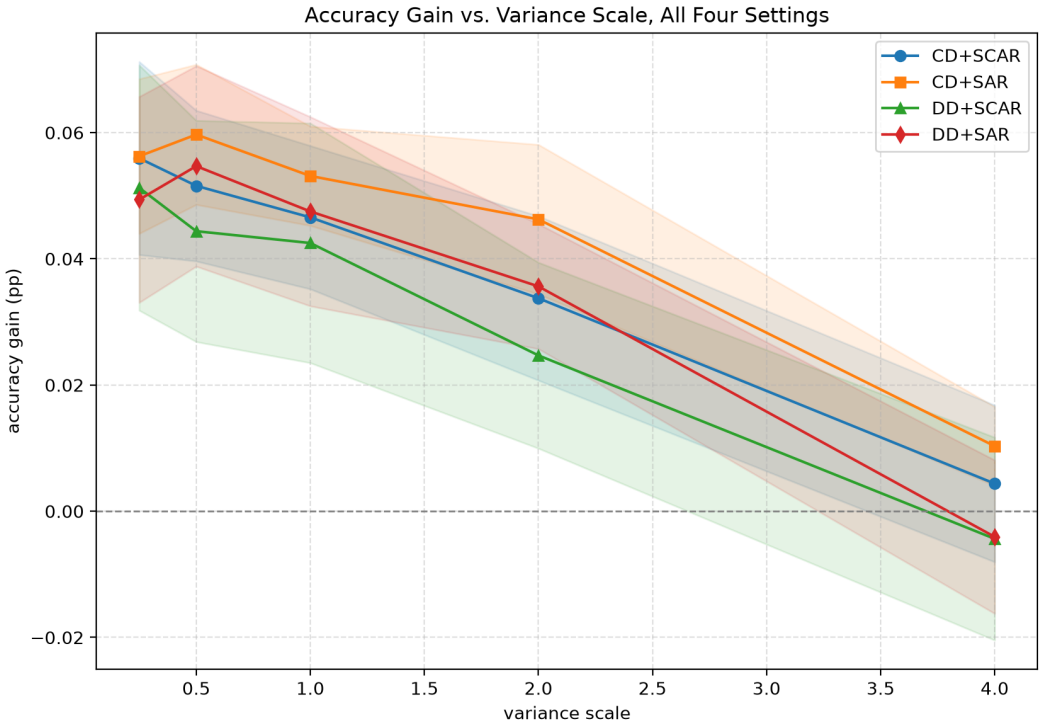

Figure 7: Accuracy gain versus latent covariance scale, compared across all four noise-and-verification settings. The monotonic decline holds in every setting.

Table 6: Hyperparameters for the standard heavy old-style real-data runs. Values are taken from the selected best-combo configuration used for the 16-dataset, 5-seed experiment.

Component	Hyperparameters	Values
Encoder	backbone; loss	`skywork`; `cls_geo_share`
Encoder size	hidden dim; latent dim	256; 16
Optimization	batch size; lr; weight decay	32; 10^{-4}; 5×10^{-4}
Training schedule	max epochs; patience; warmup	40; 8; 8
Curriculum	alt. rounds; epochs/round; pseudo weight	3; 2; 0.35
Initialization	library frac.; seed frac.; min pos/neg	0.20; 0.25; 24/16
Risk estimation	Beta prior; max iter.; tol	$(1, 19)$; 100; 10^{-5}
Neighborhood risk	k; kernel scale	30; 3.0
Trusted partition	threshold; budget scale; max frac.	0.20; 0.20; 0.20
Wash step	latent dim; kNN k; posterior threshold	32; 25; 0.55
Wash risk gate	pre-wash risk threshold; gate alpha	0.60; 1.5
Transport	frac.; min/max; relabel threshold	0.95; 0.40/0.95; 0.45
Query selection	top-k; Fisher L_2; max iter.	50; 10^{-3}; 200
Waiting pool	EMA decay; min hits	0.7; 2
Stopping	max iter.; stable rounds	24; 2
Stopping thresholds	gap; change; size; Δgap	0.35; 0.02; 0.015; 0.05

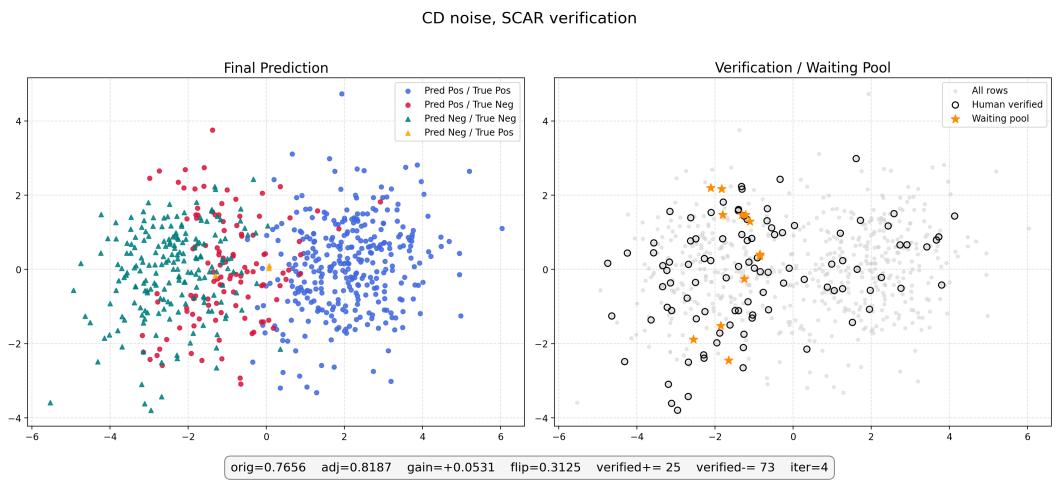

Figure 8: Simulation embedding result under class-dependent (CD) noise with SCAR verification.

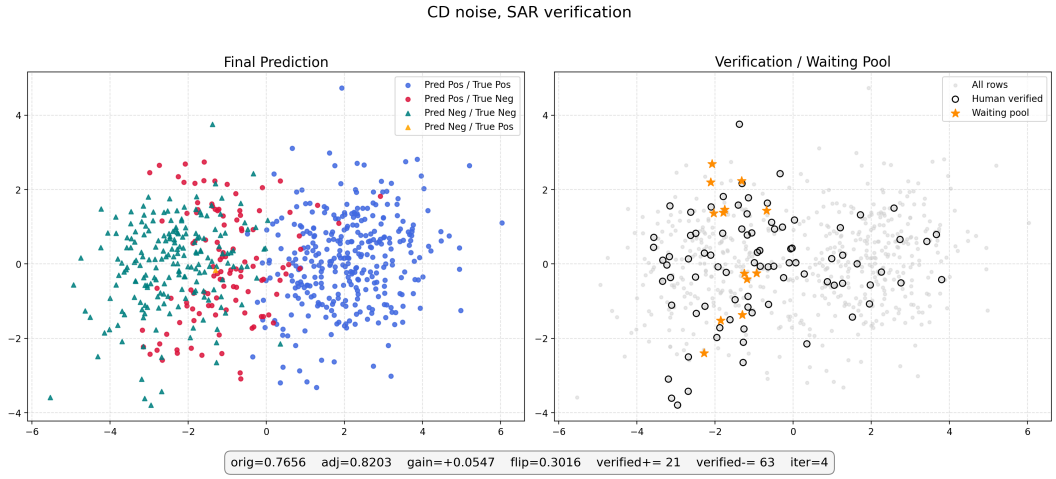

Figure 9: Simulation embedding result under class-dependent (CD) noise with SAR verification.

E.6 Additional simulation result

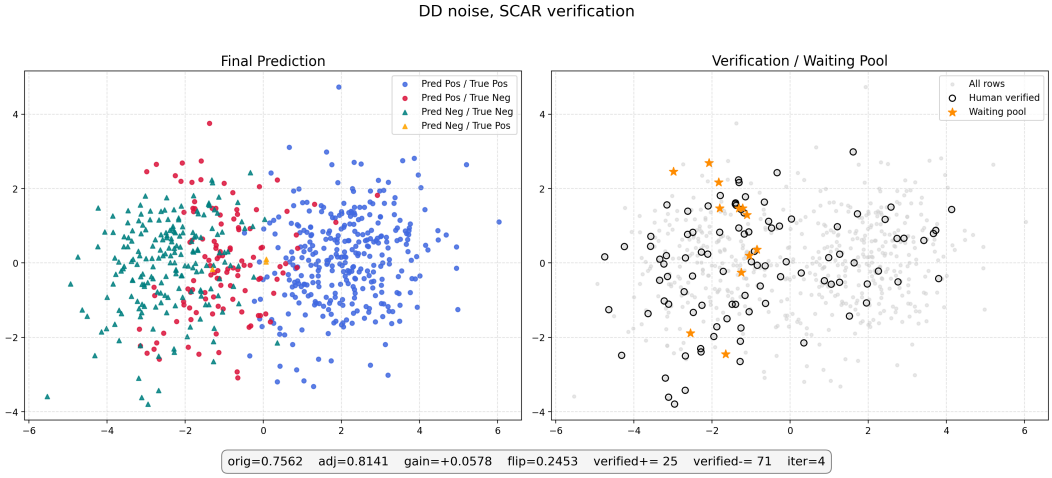

Figure 10: Simulation embedding result under distribution-dependent (DD) noise with SCAR verification.

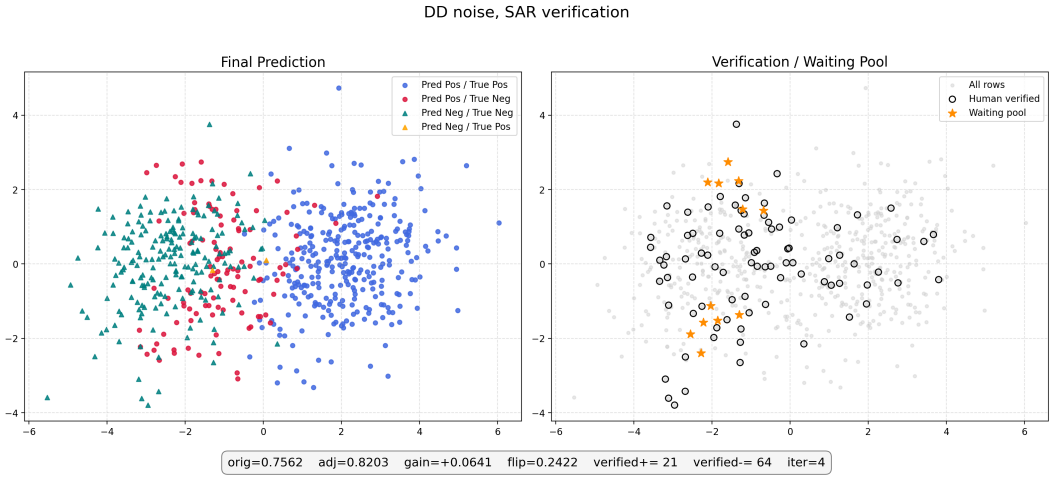

Figure 11: Simulation embedding result under distribution-dependent (DD) noise with SAR verification. This is the same run visualized in Figure 3 in the main text, shown here at full size alongside the other three settings for comparison.

Table 7: Real-data results with 3% verified data for baseline methods (Coding and MT-Bench). Accuracy and gain values are reported as mean values with range in parentheses, where the range is computed as maximum minus minimum across five random seeds. Data is the number of pairwise comparisons. Human Verif. Req. denotes the number of human-verified examples required by each method, averaged over five seeds with range in parentheses. Orig. Acc. and Adj. Acc. denote the original LLM-judge accuracy and the adjusted accuracy after refinement, respectively. Bold values indicate AURA results. Green cells highlight the highest Adj. Acc. or Gain between the best baseline and AURA within each model and question type.

Real-Data Results with 3% Verified Baseline Data (Coding and MT-Bench)

Question Type	Method	Data	Orig. Acc.	Adj. Acc.	Gain	Human Verif. Req.
			Model: GPT-5.4			
Coding	Logistic Regression	2949	63.92%	57.23% (3.57)	-6.69% (3.57)	89 (0)
Coding	Multilayer Perceptron	2949	63.92%	57.96% (9.13)	-5.96% (9.13)	89 (0)
Coding	Random Forest	2949	63.92%	61.82% (4.09)	-2.10% (4.09)	89 (0)
Coding	Label Propagation	2949	63.92%	56.34% (18.15)	-7.58% (18.15)	89 (0)
Coding	nnPU	2949	63.92%	62.45% (3.25)	-1.47% (3.25)	89 (0)
Coding	AURA	2949	63.92%	**67.20% (2.00)**	**+3.28% (2.00)**	100.6 (5)
			Model: GPT-5.4-mini			
Coding	Logistic Regression	2971	63.24%	56.34% (3.78)	-6.91% (3.78)	89 (0)
Coding	Multilayer Perceptron	2971	63.24%	59.22% (9.58)	-4.02% (9.58)	89 (0)
Coding	Random Forest	2971	63.24%	61.47% (1.77)	-1.77% (1.77)	89 (0)
Coding	Label Propagation	2971	63.24%	54.55% (19.81)	-8.70% (19.81)	89 (0)
Coding	nnPU	2971	63.24%	63.03% (0.32)	-0.21% (0.32)	89 (0)
Coding	AURA	2971	63.24%	**66.27% (1.55)**	**+3.03% (1.55)**	102.2 (9)
MT-Bench	Logistic Regression	2565	77.82%	74.27% (4.34)	-3.55% (4.34)	77 (0)
MT-Bench	Multilayer Perceptron	2565	77.82%	74.12% (14.39)	-3.70% (14.39)	77 (0)
MT-Bench	Random Forest	2565	77.82%	77.46% (1.25)	-0.36% (1.25)	77 (0)
MT-Bench	Label Propagation	2565	77.82%	64.91% (51.69)	-12.91% (51.69)	77 (0)
MT-Bench	nnPU	2565	77.82%	77.81% (0.00)	-0.01% (0.00)	77 (0)
MT-Bench	AURA	2575	77.82%	**81.01% (0.94)**	**+3.19% (0.94)**	87.4 (7)
			Model: Gemini			
Coding	Logistic Regression	2820	63.55%	56.37% (3.03)	-7.18% (3.03)	85 (0)
Coding	Multilayer Perceptron	2820	63.55%	59.33% (5.96)	-4.21% (5.96)	85 (0)
Coding	Random Forest	2820	63.55%	61.63% (4.17)	-1.92% (4.17)	85 (0)
Coding	Label Propagation	2820	63.55%	46.87% (26.29)	-16.68% (26.29)	85 (0)
Coding	nnPU	2820	63.55%	63.43% (0.14)	-0.12% (0.14)	85 (0)
Coding	AURA	2820	63.55%	**67.17% (2.84)**	**+3.62% (2.84)**	96.6 (8)
			Model: Qwen			
Coding	Logistic Regression	8884	53.73%	53.99% (1.90)	+0.27% (1.90)	266 (0)
Coding	Multilayer Perceptron	8884	53.73%	53.25% (6.15)	-0.48% (6.15)	266 (0)
Coding	Random Forest	8884	53.73%	56.70% (2.23)	+2.98% (2.23)	266 (0)
Coding	Label Propagation	8884	53.73%	50.87% (4.55)	-2.86% (4.55)	266 (0)
Coding	nnPU	8884	53.73%	55.74% (1.65)	+2.01% (1.65)	266 (0)
Coding	AURA	8884	53.73%	**77.74% (9.78)**	**+24.02% (9.78)**	278.8 (14)
			Model: Mistral			
Coding	Logistic Regression	8240	50.29%	53.24% (3.77)	+2.95% (3.77)	247 (0)
Coding	Multilayer Perceptron	8240	50.29%	54.12% (6.04)	+3.83% (6.04)	247 (0)
Coding	Random Forest	8240	50.29%	57.20% (4.63)	+6.90% (4.63)	247 (0)
Coding	Label Propagation	8240	50.29%	50.89% (1.83)	+0.60% (1.83)	247 (0)
Coding	nnPU	8240	50.29%	55.38% (3.31)	+5.09% (3.31)	247 (0)
Coding	AURA	8240	50.29%	**72.91% (0.61)**	**+22.62% (0.61)**	261.8 (10)

E.7 Additional real-data results

Table 8: Real-data results with 3% verified data for baseline methods (Math). Values follow the same reporting and highlighting conventions as Table 7.

Real-Data Results with 3% Verified Baseline Data (Math)						
Question Type	Method	Data	Orig. Acc.	Adj. Acc.	Gain	Human Verif. Req.
Model: GPT-5.4						
Math	Logistic Regression	2817	67.20%	61.05% (4.43)	-6.15% (4.43)	85 (0)
Math	Multilayer Perceptron	2817	67.20%	62.00% (8.09)	-5.20% (8.09)	85 (0)
Math	Random Forest	2817	67.20%	65.92% (2.93)	-1.28% (2.93)	85 (0)
Math	Label Propagation	2817	67.20%	67.07% (0.48)	-0.13% (0.48)	85 (0)
Math	nnPU	2817	67.20%	67.23% (0.05)	+0.03% (0.05)	85 (0)
Math	AURA	2817	67.20%	**70.05% (1.17)**	**+2.85% (1.17)**	95.8 (5)
Model: GPT-5.4-mini						
Math	Logistic Regression	2911	62.35%	58.55% (2.12)	-3.80% (2.12)	87 (0)
Math	Multilayer Perceptron	2911	62.35%	56.88% (12.64)	-5.47% (12.64)	87 (0)
Math	Random Forest	2911	62.35%	63.68% (2.41)	+1.33% (2.41)	87 (0)
Math	Label Propagation	2911	62.35%	59.89% (12.82)	-2.46% (12.82)	87 (0)
Math	nnPU	2911	62.35%	62.53% (0.65)	+0.18% (0.65)	87 (0)
Math	AURA	2911	62.35%	**65.62% (1.79)**	**+3.27% (1.79)**	99 (5)
Model: Gemini						
Math	Logistic Regression	2770	66.35%	59.71% (6.44)	-6.64% (6.44)	83 (0)
Math	Multilayer Perceptron	2770	66.35%	59.86% (11.46)	-6.50% (11.46)	83 (0)
Math	Random Forest	2770	66.35%	65.63% (3.13)	-0.72% (3.13)	83 (0)
Math	Label Propagation	2770	66.35%	56.17% (26.65)	-10.19% (26.65)	83 (0)
Math	nnPU	2770	66.35%	66.19% (0.26)	-0.16% (0.26)	83 (0)
Math	AURA	2770	66.35%	**68.79% (1.52)**	**+2.43% (1.52)**	96.2 (4)
Model: Qwen						
Math	Logistic Regression	9145	55.10%	55.71% (1.54)	+0.61% (1.54)	274 (0)
Math	Multilayer Perceptron	9145	55.10%	54.28% (10.39)	-0.82% (10.39)	274 (0)
Math	Random Forest	9145	55.10%	59.70% (2.19)	+4.60% (2.19)	274 (0)
Math	Label Propagation	9145	55.10%	53.82% (2.15)	-1.28% (2.15)	274 (0)
Math	nnPU	9145	55.10%	58.88% (0.90)	+3.78% (0.90)	274 (0)
Math	AURA	9145	55.10%	**79.27% (5.47)**	**+24.17% (5.47)**	291.2 (10)
Model: Mistral						
Math	Logistic Regression	8361	50.46%	56.73% (2.75)	+6.27% (2.75)	251 (0)
Math	Multilayer Perceptron	8361	50.46%	54.30% (5.60)	+3.84% (5.60)	251 (0)
Math	Random Forest	8361	50.46%	60.53% (1.94)	+10.07% (1.94)	251 (0)
Math	Label Propagation	8361	50.46%	51.55% (3.11)	+1.09% (3.11)	251 (0)
Math	nnPU	8361	50.46%	55.24% (4.09)	+4.78% (4.09)	251 (0)
Math	AURA	8361	50.46%	**71.75% (6.54)**	**+21.29% (6.54)**	266 (23)

Table 9: Real-data results with 3% verified data for baseline methods (Factual). Values follow the same reporting and highlighting conventions as Table 7.

Real-Data Results with 3% Verified Baseline Data (Factual)						
Question Type	Method	Data	Orig. Acc.	Adj. Acc.	Gain	Human Verif. Req.
Model: GPT-5.4						
Factual	Logistic Regression	2842	64.39%	58.72% (7.65)	-5.68% (7.65)	85 (0)
Factual	Multilayer Perceptron	2842	64.39%	59.01% (7.83)	-5.38% (7.83)	85 (0)
Factual	Random Forest	2842	64.39%	63.40% (2.97)	-0.99% (2.97)	85 (0)
Factual	Label Propagation	2842	64.39%	54.87% (23.10)	-9.52% (23.10)	85 (0)
Factual	nnPU	2842	64.39%	64.39% (0.02)	-0.00% (0.02)	85 (0)
Factual	AURA	2842	64.39%	**67.66% (2.22)**	**+3.27% (2.22)**	95.6 (9)
Model: GPT-5.4-mini						
Factual	Logistic Regression	2920	62.47%	57.20% (4.31)	-5.26% (4.31)	88 (0)
Factual	Multilayer Perceptron	2920	62.47%	56.79% (5.19)	-5.68% (5.19)	88 (0)
Factual	Random Forest	2920	62.47%	61.52% (4.03)	-0.95% (4.03)	88 (0)
Factual	Label Propagation	2920	62.47%	53.23% (18.50)	-9.24% (18.50)	88 (0)
Factual	nnPU	2920	62.47%	62.44% (0.11)	-0.03% (0.11)	88 (0)
Factual	AURA	2920	62.47%	**66.07% (2.09)**	**+3.60% (2.09)**	99 (17)
Model: Gemini						
Factual	Logistic Regression	2768	66.37%	59.14% (4.13)	-7.22% (4.13)	83 (0)
Factual	Multilayer Perceptron	2768	66.37%	62.22% (7.82)	-4.15% (7.82)	83 (0)
Factual	Random Forest	2768	66.37%	65.29% (2.20)	-1.08% (2.20)	83 (0)
Factual	Label Propagation	2768	66.37%	54.91% (26.63)	-11.45% (26.63)	83 (0)
Factual	nnPU	2768	66.37%	66.03% (0.75)	-0.34% (0.75)	83 (0)
Factual	AURA	2768	66.37%	**69.21% (1.73)**	**+2.85% (1.73)**	94 (12)
Model: Qwen						
Factual	Logistic Regression	9333	55.94%	56.15% (2.88)	+0.21% (2.88)	280 (0)
Factual	Multilayer Perceptron	9333	55.94%	56.51% (5.39)	+0.57% (5.39)	280 (0)
Factual	Random Forest	9333	55.94%	60.15% (2.14)	+4.21% (2.14)	280 (0)
Factual	Label Propagation	9333	55.94%	51.89% (6.01)	-4.05% (6.01)	280 (0)
Factual	nnPU	9333	55.94%	59.23% (1.63)	+3.29% (1.63)	280 (0)
Factual	AURA	9333	55.94%	**80.31% (4.76)**	**+24.37% (4.76)**	292.8 (11)
Model: Mistral						
Factual	Logistic Regression	8260	51.05%	56.11% (2.91)	+5.06% (2.91)	248 (0)
Factual	Multilayer Perceptron	8260	51.05%	53.90% (7.81)	+2.84% (7.81)	248 (0)
Factual	Random Forest	8260	51.05%	59.96% (2.97)	+8.91% (2.97)	248 (0)
Factual	Label Propagation	8260	51.05%	50.87% (2.60)	-0.19% (2.60)	248 (0)
Factual	nnPU	8260	51.05%	56.32% (3.73)	+5.27% (3.73)	248 (0)
Factual	AURA	8260	51.05%	**72.37% (6.49)**	**+21.31% (6.49)**	259 (7)

Table 10: Real-data results with 20% verified data for baseline methods (Coding and MT-Bench). Values follow the same reporting and highlighting conventions as Table 7.

Real-Data Results with 20% Verified Baseline Data (Coding and MT-Bench)						
Question Type	Method	Data	Orig. Acc.	Adj. Acc.	Gain	Human Verif. Req.
Model: GPT-5.4						
Coding	Logistic Regression	2949	63.92%	59.54% (3.56)	-4.38% (3.56)	590 (0)
Coding	Multilayer Perceptron	2949	63.92%	60.83% (2.80)	-3.09% (2.80)	590 (0)
Coding	Random Forest	2949	63.92%	63.23% (1.87)	-0.69% (1.87)	590 (0)
Coding	Label Propagation	2949	63.92%	60.50% (9.92)	-3.42% (9.92)	590 (0)
Coding	nnPU	2949	63.92%	63.86% (0.10)	-0.06% (0.10)	590 (0)
Coding	AURA	2949	63.92%	**67.20% (2.00)**	**+3.28% (2.00)**	100.6 (5)
Model: GPT-5.4-mini						
Coding	Logistic Regression	2971	63.24%	59.20% (0.84)	-4.04% (0.84)	594 (0)
Coding	Multilayer Perceptron	2971	63.24%	61.40% (4.29)	-1.85% (4.29)	594 (0)
Coding	Random Forest	2971	63.24%	63.27% (1.22)	+0.03% (1.22)	594 (0)
Coding	Label Propagation	2971	63.24%	54.38% (11.11)	-8.87% (11.11)	594 (0)
Coding	nnPU	2971	63.24%	63.43% (0.73)	+0.19% (0.73)	594 (0)
Coding	AURA	2971	63.24%	**66.27% (1.55)**	**+3.03% (1.55)**	102.2 (9)
MT-Bench	Logistic Regression	2565	77.82%	75.39% (3.61)	-2.43% (3.61)	513 (0)
MT-Bench	Multilayer Perceptron	2565	77.82%	77.66% (0.54)	-0.16% (0.54)	513 (0)
MT-Bench	Random Forest	2565	77.82%	77.28% (1.80)	-0.54% (1.80)	513 (0)
MT-Bench	Label Propagation	2565	77.82%	76.74% (3.61)	-1.07% (3.61)	513 (0)
MT-Bench	nnPU	2565	77.82%	77.83% (0.00)	+0.01% (0.00)	513 (0)
MT-Bench	AURA	2575	77.82%	**81.01% (0.94)**	**+3.19% (0.94)**	87.4 (7)
Model: Gemini						
Coding	Logistic Regression	2820	63.55%	59.72% (2.57)	-3.83% (2.57)	564 (0)
Coding	Multilayer Perceptron	2820	63.55%	61.76% (6.25)	-1.79% (6.25)	564 (0)
Coding	Random Forest	2820	63.55%	63.39% (1.68)	-0.16% (1.68)	564 (0)
Coding	Label Propagation	2820	63.55%	58.65% (6.12)	-4.89% (6.12)	564 (0)
Coding	nnPU	2820	63.55%	63.51% (0.12)	-0.04% (0.12)	564 (0)
Coding	AURA	2820	63.55%	**67.17% (2.84)**	**+3.62% (2.84)**	96.6 (8)
Model: Qwen						
Coding	Logistic Regression	8884	53.73%	58.47% (1.59)	+4.75% (1.59)	1777 (0)
Coding	Multilayer Perceptron	8884	53.73%	57.44% (1.83)	+3.72% (1.83)	1777 (0)
Coding	Random Forest	8884	53.73%	59.20% (0.96)	+5.47% (0.96)	1777 (0)
Coding	Label Propagation	8884	53.73%	53.77% (2.11)	+0.04% (2.11)	1777 (0)
Coding	nnPU	8884	53.73%	58.54% (0.48)	+4.81% (0.48)	1777 (0)
Coding	AURA	8884	53.73%	**77.74% (9.78)**	**+24.02% (9.78)**	278.8 (14)
Model: Mistral						
Coding	Logistic Regression	8240	50.29%	57.52% (1.18)	+7.22% (1.18)	1648 (0)
Coding	Multilayer Perceptron	8240	50.29%	56.82% (2.64)	+6.53% (2.64)	1648 (0)
Coding	Random Forest	8240	50.29%	58.88% (1.27)	+8.59% (1.27)	1648 (0)
Coding	Label Propagation	8240	50.29%	53.83% (1.94)	+3.53% (1.94)	1648 (0)
Coding	nnPU	8240	50.29%	57.97% (0.59)	+7.68% (0.59)	1648 (0)
Coding	AURA	8240	50.29%	**72.91% (0.61)**	**+22.62% (0.61)**	261.8 (10)

Table 11: Real-data results with 20% verified data for baseline methods (Math). Values follow the same reporting and highlighting conventions as Table 7.

Real-Data Results with 20% Verified Baseline Data (Math)						
Question Type	Method	Data	Orig. Acc.	Adj. Acc.	Gain	Human Verif. Req.
Model: GPT-5.4						
Math	Logistic Regression	2817	67.20%	62.92% (3.77)	-4.28% (3.77)	564 (0)
Math	Multilayer Perceptron	2817	67.20%	65.45% (2.75)	-1.75% (2.75)	564 (0)
Math	Random Forest	2817	67.20%	67.98% (1.38)	+0.78% (1.38)	564 (0)
Math	Label Propagation	2817	67.20%	65.28% (2.66)	-1.92% (2.66)	564 (0)
Math	nnPU	2817	67.20%	67.32% (0.19)	+0.12% (0.19)	564 (0)
Math	AURA	2817	67.20%	**70.05% (1.17)**	**+2.85% (1.17)**	95.8 (5)
Model: GPT-5.4-mini						
Math	Logistic Regression	2911	62.35%	62.18% (2.23)	-0.17% (2.23)	582 (0)
Math	Multilayer Perceptron	2911	62.35%	63.31% (1.50)	+0.97% (1.50)	582 (0)
Math	Random Forest	2911	62.35%	65.47% (2.49)	+3.12% (2.49)	582 (0)
Math	Label Propagation	2911	62.35%	62.46% (1.80)	+0.11% (1.80)	582 (0)
Math	nnPU	2911	62.35%	63.31% (0.77)	+0.96% (0.77)	582 (0)
Math	AURA	2911	62.35%	**65.62% (1.79)**	**+3.27% (1.79)**	99 (5)
Model: Gemini						
Math	Logistic Regression	2770	66.35%	63.61% (2.44)	-2.74% (2.44)	554 (0)
Math	Multilayer Perceptron	2770	66.35%	64.50% (2.21)	-1.85% (2.21)	554 (0)
Math	Random Forest	2770	66.35%	67.45% (1.44)	+1.09% (1.44)	554 (0)
Math	Label Propagation	2770	66.35%	65.05% (3.70)	-1.31% (3.70)	554 (0)
Math	nnPU	2770	66.35%	66.31% (0.13)	-0.04% (0.13)	554 (0)
Math	AURA	2770	66.35%	**68.79% (1.52)**	**+2.43% (1.52)**	96.2 (4)
Model: Qwen						
Math	Logistic Regression	9145	55.10%	61.13% (1.22)	+6.03% (1.22)	1829 (0)
Math	Multilayer Perceptron	9145	55.10%	59.60% (3.35)	+4.50% (3.35)	1829 (0)
Math	Random Forest	9145	55.10%	62.85% (1.56)	+7.75% (1.56)	1829 (0)
Math	Label Propagation	9145	55.10%	55.35% (1.18)	+0.25% (1.18)	1829 (0)
Math	nnPU	9145	55.10%	60.27% (1.58)	+5.17% (1.58)	1829 (0)
Math	AURA	9145	55.10%	**79.27% (5.47)**	**+24.17% (5.47)**	291.2 (10)
Model: Mistral						
Math	Logistic Regression	8361	50.46%	61.63% (0.76)	+11.17% (0.76)	1672 (0)
Math	Multilayer Perceptron	8361	50.46%	60.61% (1.97)	+10.15% (1.97)	1672 (0)
Math	Random Forest	8361	50.46%	62.63% (1.38)	+12.17% (1.38)	1672 (0)
Math	Label Propagation	8361	50.46%	55.64% (0.78)	+5.18% (0.78)	1672 (0)
Math	nnPU	8361	50.46%	60.68% (0.62)	+10.22% (0.62)	1672 (0)
Math	AURA	8361	50.46%	**71.75% (6.54)**	**+21.29% (6.54)**	266 (23)

Table 12: Real-data results with 20% verified data for baseline methods (Factual). Values follow the same reporting and highlighting conventions as Table 7.

Real-Data Results with 20% Verified Baseline Data (Factual)						
Question Type	Method	Data	Orig. Acc.	Adj. Acc.	Gain	Human Verif. Req.
Model: GPT-5.4						
Factual	Logistic Regression	2842	64.39%	62.19% (2.02)	-2.20% (2.02)	568 (0)
Factual	Multilayer Perceptron	2842	64.39%	64.12% (2.07)	-0.27% (2.07)	568 (0)
Factual	Random Forest	2842	64.39%	65.82% (1.72)	+1.43% (1.72)	568 (0)
Factual	Label Propagation	2842	64.39%	61.84% (4.31)	-2.55% (4.31)	568 (0)
Factual	nnPU	2842	64.39%	65.03% (0.75)	+0.64% (0.75)	568 (0)
Factual	AURA	2842	64.39%	**67.66% (2.22)**	**+3.27% (2.22)**	95.6 (9)
Model: GPT-5.4-mini						
Factual	Logistic Regression	2920	62.47%	61.38% (3.51)	-1.09% (3.51)	584 (0)
Factual	Multilayer Perceptron	2920	62.47%	61.44% (3.98)	-1.03% (3.98)	584 (0)
Factual	Random Forest	2920	62.47%	64.77% (1.97)	+2.30% (1.97)	584 (0)
Factual	Label Propagation	2920	62.47%	59.41% (3.17)	-3.06% (3.17)	584 (0)
Factual	nnPU	2920	62.47%	63.16% (0.57)	+0.69% (0.57)	584 (0)
Factual	AURA	2920	62.47%	**66.07% (2.09)**	**+3.60% (2.09)**	99 (17)
Model: Gemini						
Factual	Logistic Regression	2768	66.37%	61.16% (2.71)	-5.21% (2.71)	553 (0)
Factual	Multilayer Perceptron	2768	66.37%	64.81% (6.41)	-1.55% (6.41)	553 (0)
Factual	Random Forest	2768	66.37%	66.58% (0.99)	+0.22% (0.99)	553 (0)
Factual	Label Propagation	2768	66.37%	61.95% (4.38)	-4.42% (4.38)	553 (0)
Factual	nnPU	2768	66.37%	66.33% (0.08)	-0.04% (0.08)	553 (0)
Factual	AURA	2768	66.37%	**69.21% (1.73)**	**+2.85% (1.73)**	94 (12)
Model: Qwen						
Factual	Logistic Regression	9333	55.94%	61.22% (0.87)	+5.27% (0.87)	1866 (0)
Factual	Multilayer Perceptron	9333	55.94%	60.10% (2.68)	+4.16% (2.68)	1866 (0)
Factual	Random Forest	9333	55.94%	63.01% (1.18)	+7.07% (1.18)	1866 (0)
Factual	Label Propagation	9333	55.94%	55.41% (3.48)	-0.53% (3.48)	1866 (0)
Factual	nnPU	9333	55.94%	60.63% (1.42)	+4.69% (1.42)	1866 (0)
Factual	AURA	9333	55.94%	**80.31% (4.76)**	**+24.37% (4.76)**	292.8 (11)
Model: Mistral						
Factual	Logistic Regression	8260	51.05%	61.67% (0.86)	+10.62% (0.86)	1652 (0)
Factual	Multilayer Perceptron	8260	51.05%	58.73% (3.65)	+7.68% (3.65)	1652 (0)
Factual	Random Forest	8260	51.05%	62.78% (1.07)	+11.73% (1.07)	1652 (0)
Factual	Label Propagation	8260	51.05%	54.42% (1.88)	+3.37% (1.88)	1652 (0)
Factual	nnPU	8260	51.05%	60.66% (0.60)	+9.61% (0.60)	1652 (0)
Factual	AURA	8260	51.05%	**72.37% (6.49)**	**+21.31% (6.49)**	259 (7)

Table 13: Real-data results with 80% verified data for baseline methods (Coding and MT-Bench). Values follow the same reporting and highlighting conventions as Table 7.

Real-Data Results with 80% Verified Baseline Data (Coding and MT-Bench)						
Question Type	Method	Data	Orig. Acc.	Adj. Acc.	Gain	Human Verif. Req.
Model: GPT-5.4						
Coding	Logistic Regression	2949	63.92%	62.17% (1.86)	-1.75% (1.86)	2359 (0)
Coding	Multilayer Perceptron	2949	63.92%	61.93% (3.73)	-1.99% (3.73)	2359 (0)
Coding	Random Forest	2949	63.92%	64.10% (2.37)	+0.18% (2.37)	2359 (0)
Coding	Label Propagation	2949	63.92%	60.07% (3.22)	-3.85% (3.22)	2359 (0)
Coding	nnPU	2949	63.92%	63.97% (0.26)	+0.05% (0.26)	2359 (0)
Coding	AURA	2949	63.92%	**67.20% (2.00)**	**+3.28% (2.00)**	100.6 (5)
Model: GPT-5.4-mini						
Coding	Logistic Regression	2971	63.24%	62.26% (4.88)	-0.99% (4.88)	2377 (0)
Coding	Multilayer Perceptron	2971	63.24%	61.48% (3.70)	-1.76% (3.70)	2377 (0)
Coding	Random Forest	2971	63.24%	63.20% (3.03)	-0.05% (3.03)	2377 (0)
Coding	Label Propagation	2971	63.24%	58.18% (3.54)	-5.06% (3.54)	2377 (0)
Coding	nnPU	2971	63.24%	63.77% (0.32)	+0.53% (0.32)	2377 (0)
Coding	AURA	2971	63.24%	**66.27% (1.55)**	**+3.03% (1.55)**	102.2 (9)
MT-Bench	Logistic Regression	2565	77.82%	78.67% (3.90)	+0.86% (3.90)	2052 (0)
MT-Bench	Multilayer Perceptron	2565	77.82%	78.91% (1.36)	+1.09% (1.36)	2052 (0)
MT-Bench	Random Forest	2565	77.82%	77.66% (6.04)	-0.16% (6.04)	2052 (0)
MT-Bench	Label Propagation	2565	77.82%	78.75% (2.92)	+0.94% (2.92)	2052 (0)
MT-Bench	nnPU	2565	77.82%	77.78% (0.00)	-0.04% (0.00)	2052 (0)
MT-Bench	AURA	2575	77.82%	**81.01% (0.94)**	**+3.19% (0.94)**	87.4 (7)
Model: Gemini						
Coding	Logistic Regression	2820	63.55%	61.67% (2.48)	-1.88% (2.48)	2256 (0)
Coding	Multilayer Perceptron	2820	63.55%	62.52% (4.26)	-1.03% (4.26)	2256 (0)
Coding	Random Forest	2820	63.55%	63.76% (3.19)	+0.21% (3.19)	2256 (0)
Coding	Label Propagation	2820	63.55%	58.72% (4.79)	-4.82% (4.79)	2256 (0)
Coding	nnPU	2820	63.55%	63.33% (0.23)	-0.22% (0.23)	2256 (0)
Coding	AURA	2820	63.55%	**67.17% (2.84)**	**+3.62% (2.84)**	96.6 (8)
Model: Qwen						
Coding	Logistic Regression	8884	53.73%	60.34% (1.80)	+6.61% (1.80)	7107 (0)
Coding	Multilayer Perceptron	8884	53.73%	58.74% (4.05)	+5.01% (4.05)	7107 (0)
Coding	Random Forest	8884	53.73%	60.30% (1.63)	+6.58% (1.63)	7107 (0)
Coding	Label Propagation	8884	53.73%	54.58% (1.91)	+0.85% (1.91)	7107 (0)
Coding	nnPU	8884	53.73%	58.91% (0.50)	+5.18% (0.50)	7107 (0)
Coding	AURA	8884	53.73%	**77.74% (9.78)**	**+24.02% (9.78)**	278.8 (14)
Model: Mistral						
Coding	Logistic Regression	8240	50.29%	59.14% (1.94)	+8.85% (1.94)	6592 (0)
Coding	Multilayer Perceptron	8240	50.29%	58.40% (2.06)	+8.11% (2.06)	6592 (0)
Coding	Random Forest	8240	50.29%	60.15% (1.40)	+9.85% (1.40)	6592 (0)
Coding	Label Propagation	8240	50.29%	56.36% (3.16)	+6.07% (3.16)	6592 (0)
Coding	nnPU	8240	50.29%	59.11% (0.99)	+8.82% (0.99)	6592 (0)
Coding	AURA	8240	50.29%	**72.91% (0.61)**	**+22.62% (0.61)**	261.8 (10)

Table 14: Real-data results with 80% verified data for baseline methods (Math). Values follow the same reporting and highlighting conventions as Table 7.

Real-Data Results with 80% Verified Baseline Data (Math)						
Question Type	Method	Data	Orig. Acc.	Adj. Acc.	Gain	Human Verif. Req.
Model: GPT-5.4						
Math	Logistic Regression	2817	67.20%	67.16% (1.06)	-0.04% (1.06)	2253 (0)
Math	Multilayer Perceptron	2817	67.20%	66.06% (3.72)	-1.14% (3.72)	2253 (0)
Math	Random Forest	2817	67.20%	67.84% (1.95)	+0.64% (1.95)	2253 (0)
Math	Label Propagation	2817	67.20%	65.64% (1.42)	-1.56% (1.42)	2253 (0)
Math	nnPU	2817	67.20%	67.45% (0.32)	+0.25% (0.32)	2253 (0)
Math	AURA	2817	67.20%	**70.05% (1.17)**	**+2.85% (1.17)**	95.8 (5)
Model: GPT-5.4-mini						
Math	Logistic Regression	2911	62.35%	66.01% (2.23)	+3.66% (2.23)	2329 (0)
Math	Multilayer Perceptron	2911	62.35%	65.15% (3.61)	+2.80% (3.61)	2329 (0)
Math	Random Forest	2911	62.35%	66.84% (3.78)	+4.49% (3.78)	2329 (0)
Math	Label Propagation	2911	62.35%	61.79% (3.95)	-0.56% (3.95)	2329 (0)
Math	nnPU	2911	62.35%	63.85% (1.46)	+1.50% (1.46)	2329 (0)
Math	AURA	2911	62.35%	**65.62% (1.79)**	**+3.27% (1.79)**	99 (5)
Model: Gemini						
Math	Logistic Regression	2770	66.35%	67.58% (3.07)	+1.23% (3.07)	2216 (0)
Math	Multilayer Perceptron	2770	66.35%	67.73% (2.89)	+1.37% (2.89)	2216 (0)
Math	Random Forest	2770	66.35%	68.19% (1.81)	+1.84% (1.81)	2216 (0)
Math	Label Propagation	2770	66.35%	66.03% (1.62)	-0.32% (1.62)	2216 (0)
Math	nnPU	2770	66.35%	67.15% (0.61)	+0.80% (0.61)	2216 (0)
Math	AURA	2770	66.35%	**68.79% (1.52)**	**+2.43% (1.52)**	96.2 (4)
Model: Qwen						
Math	Logistic Regression	9145	55.10%	61.90% (2.08)	+6.80% (2.08)	7316 (0)
Math	Multilayer Perceptron	9145	55.10%	62.07% (2.35)	+6.97% (2.35)	7316 (0)
Math	Random Forest	9145	55.10%	63.59% (1.97)	+8.49% (1.97)	7316 (0)
Math	Label Propagation	9145	55.10%	57.89% (2.95)	+2.79% (2.95)	7316 (0)
Math	nnPU	9145	55.10%	61.12% (1.67)	+6.02% (1.67)	7316 (0)
Math	AURA	9145	55.10%	**79.27% (5.47)**	**+24.17% (5.47)**	291.2 (10)
Model: Mistral						
Math	Logistic Regression	8361	50.46%	62.91% (2.39)	+12.45% (2.39)	6689 (0)
Math	Multilayer Perceptron	8361	50.46%	61.77% (4.31)	+11.31% (4.31)	6689 (0)
Math	Random Forest	8361	50.46%	64.10% (2.21)	+13.64% (2.21)	6689 (0)
Math	Label Propagation	8361	50.46%	57.97% (2.69)	+7.51% (2.69)	6689 (0)
Math	nnPU	8361	50.46%	62.26% (1.60)	+11.80% (1.60)	6689 (0)
Math	AURA	8361	50.46%	**71.75% (6.54)**	**+21.29% (6.54)**	266 (23)

Table 15: Real-data results with 80% verified data for baseline methods (Factual). Values follow the same reporting and highlighting conventions as Table 7.

Real-Data Results with 80% Verified Baseline Data (Factual)						
Question Type	Method	Data	Orig. Acc.	Adj. Acc.	Gain	Human Verif. Req.
Model: GPT-5.4						
Factual	Logistic Regression	2842	64.39%	65.53% (2.11)	+1.14% (2.11)	2274 (0)
Factual	Multilayer Perceptron	2842	64.39%	65.07% (5.63)	+0.68% (5.63)	2274 (0)
Factual	Random Forest	2842	64.39%	65.67% (2.99)	+1.28% (2.99)	2274 (0)
Factual	Label Propagation	2842	64.39%	62.04% (3.35)	-2.35% (3.35)	2274 (0)
Factual	nnPU	2842	64.39%	65.11% (0.79)	+0.72% (0.79)	2274 (0)
Factual	AURA	2842	64.39%	**67.66% (2.22)**	**+3.27% (2.22)**	95.6 (9)
Model: GPT-5.4-mini						
Factual	Logistic Regression	2920	62.47%	65.17% (5.82)	+2.71% (5.82)	2336 (0)
Factual	Multilayer Perceptron	2920	62.47%	64.55% (3.77)	+2.09% (3.77)	2336 (0)
Factual	Random Forest	2920	62.47%	66.16% (2.40)	+3.70% (2.40)	2336 (0)
Factual	Label Propagation	2920	62.47%	61.37% (2.91)	-1.10% (2.91)	2336 (0)
Factual	nnPU	2920	62.47%	64.42% (1.24)	+1.95% (1.24)	2336 (0)
Factual	AURA	2920	62.47%	**66.07% (2.09)**	**+3.60% (2.09)**	99 (17)
Model: Gemini						
Factual	Logistic Regression	2768	66.37%	65.28% (5.42)	-1.09% (5.42)	2215 (0)
Factual	Multilayer Perceptron	2768	66.37%	65.03% (4.16)	-1.34% (4.16)	2215 (0)
Factual	Random Forest	2768	66.37%	66.26% (2.53)	-0.11% (2.53)	2215 (0)
Factual	Label Propagation	2768	66.37%	63.94% (1.27)	-2.42% (1.27)	2215 (0)
Factual	nnPU	2768	66.37%	66.37% (0.00)	-0.00% (0.00)	2215 (0)
Factual	AURA	2768	66.37%	**69.21% (1.73)**	**+2.85% (1.73)**	94 (12)
Model: Qwen						
Factual	Logistic Regression	9333	55.94%	64.18% (1.77)	+8.24% (1.77)	7467 (0)
Factual	Multilayer Perceptron	9333	55.94%	63.12% (3.11)	+7.18% (3.11)	7467 (0)
Factual	Random Forest	9333	55.94%	64.24% (2.47)	+8.30% (2.47)	7467 (0)
Factual	Label Propagation	9333	55.94%	59.37% (2.04)	+3.43% (2.04)	7467 (0)
Factual	nnPU	9333	55.94%	62.60% (0.94)	+6.66% (0.94)	7467 (0)
Factual	AURA	9333	55.94%	**80.31% (4.76)**	**+24.37% (4.76)**	292.8 (11)
Model: Mistral						
Factual	Logistic Regression	8260	51.05%	62.60% (3.93)	+11.55% (3.93)	6608 (0)
Factual	Multilayer Perceptron	8260	51.05%	61.74% (4.00)	+10.69% (4.00)	6608 (0)
Factual	Random Forest	8260	51.05%	62.65% (2.54)	+11.60% (2.54)	6608 (0)
Factual	Label Propagation	8260	51.05%	57.99% (2.24)	+6.94% (2.24)	6608 (0)
Factual	nnPU	8260	51.05%	60.65% (0.84)	+9.60% (0.84)	6608 (0)
Factual	AURA	8260	51.05%	**72.37% (6.49)**	**+21.31% (6.49)**	259 (7)

Table 16: Runtime and memory usage for the real-data experiments, measured at the 20% verified-budget level. Time and memory are reported as mean values, with ranges in parentheses across ten random seeds.

Runtime and Memory Usage for Real-Data Experiments

Question Type	Model	# Data	Time (min)	Memory (GB)
Coding	GPT-5.4	2949	5.91 (4.84–7.99)	1.35 (1.31–1.41)
Coding	GPT-5.4-mini	2971	5.79 (4.59–7.65)	1.32 (1.28–1.40)
Coding	Gemini	2820	5.60 (4.38–6.91)	1.33 (1.30–1.42)
Coding	Qwen	8884	17.22 (13.53–26.79)	2.05 (2.04–2.10)
Coding	Mistral	8240	13.96 (11.32–15.81)	1.84 (1.82–1.85)
Math	GPT-5.4	2817	5.24 (4.55–6.64)	1.35 (1.30–1.40)
Math	GPT-5.4-mini	2911	5.34 (4.67–6.78)	1.34 (1.29–1.41)
Math	Gemini	2770	5.19 (4.28–6.04)	1.32 (1.30–1.40)
Math	Qwen	9145	18.19 (13.92–25.86)	2.11 (2.09–2.19)
Math	Mistral	8361	15.55 (11.77–17.90)	1.84 (1.83–1.85)
Factual	GPT-5.4	2842	5.33 (4.52–6.88)	1.31 (1.28–1.33)
Factual	GPT-5.4-mini	2920	5.47 (4.69–7.19)	1.32 (1.31–1.33)
Factual	Gemini	2768	5.71 (4.76–8.73)	1.33 (1.30–1.40)
Factual	Qwen	9333	17.95 (14.22–20.14)	2.17 (2.15–2.18)
Factual	Mistral	8260	16.15 (13.85–19.61)	1.81 (1.80–1.82)
MT-Bench	GPT-5.4-mini	2565	1.58 (1.47–1.63)	1.65 (1.64–1.66)

E.8 Runtime and memory usage