THE EVALUATION BOTTLENECK: YOUR BIGGEST JUDGE IS NOT YOUR BEST JUDGE

Anonymous authors
Paper under double-blind review

ABSTRACT

Agentic LLM systems generate outputs faster than they can be reliably reviewed: generation is cheap, evaluation is not. The common responses are to make the judge bigger or more specialized. We ask a more basic architectural question: where should specialization live in an evaluation system—in the judge itself, or in the rule that decides when its verdict can be trusted? We test both placements. In the first, we split a judge's training data across expert evaluators, one per family of criteria. In the second, we keep judges of different sizes frozen and train a small *trust head* that either accepts a cheap judge's verdict or defers to a stronger one, with every automatic release audited against a predeclared risk target. The findings reverse intuition. Specializing the judge backfires: splitting supervision across eight experts costs 10 accuracy points and shrinks the fraction of judgments that can be safely released more than fourfold—a loss that vanishes once the experts start from a shared trained judge. Specializing the trust decision wins: on RewardBench 2, a 15-parameter trust head routes a cascade of small-to-large judges to an accuracy 4.7 points *above* the largest judge alone at $0.42\times$ its compute, passing exact 95% risk audits in all 20 runs, while naive confidence-based escalation yields neither an accuracy gain nor compute savings. The design rule is simple: share judgment learning by default, and specialize *when to trust*, not *what judges know*—provided the earlier judges are genuinely cheap.

1 INTRODUCTION

Language-model agents produce drafts, plans, and tool trajectories much faster than any careful reviewer—human or model—can inspect them. In a demo this imbalance is invisible; in a deployed system, where an output must also satisfy a local policy, a professional standard, or a task-specific rubric, review becomes the limiting resource. We call this the *evaluation bottleneck*.

The standard response is LLM-as-a-judge: put a large model behind the agent and let it score outputs (Kim et al., 2024a;b). This helps, but it does not settle the operational problem. The judge is called at roughly the rate of generation, so its cost scales with the bottleneck it is meant to relieve; a scalar score does not record which rule governed the decision; and aggregate accuracy says nothing about which *individual* judgments are reliable enough to leave the review queue. In practice, an evaluator that knows when to abstain is often more useful than a slightly more accurate evaluator that does not.

Practitioners facing this bottleneck reach for two intuitive levers. The first is **specialize the judge**: fine-tune expert evaluators for each family of criteria, on the theory that a rubric-specific judge beats a generalist. The second is **scale the judge**: route everything to the biggest model available, on the theory that its judgment dominates. This paper tests both levers under controlled conditions with held-out risk audits, and finds that neither puts effort in the right place: the first actively hurts, and the second is dominated by something far cheaper. What succeeds is a third placement of specialization that neither lever touches: a small learned rule that decides, per example, whether a cheap judge's verdict can be *trusted* or must be escalated to a stronger one. In the selective-prediction literature this accept-or-escalate decision is called *deferral* (Chow, 1970; Cortes et al., 2016; Madras et al., 2018; Mozannar & Sontag, 2020); we use "trust decision" and "deferral" interchangeably, and Table 1 fixes all vocabulary used in the paper.

Our headline finding gives the paper its title. On RewardBench 2 (Malik et al., 2026), a cascade of 0.6B, 4B, and 8B reward models governed by a logistic trust head reaches 89.40% top-choice

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

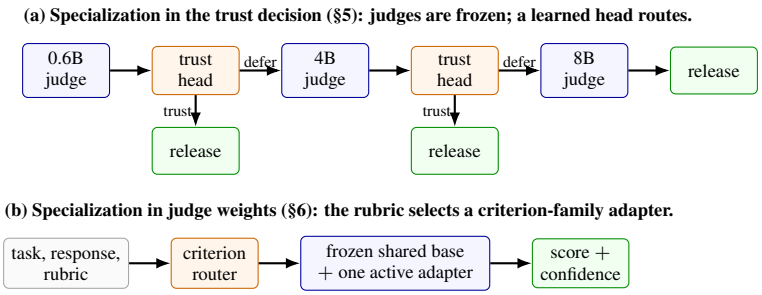

Figure 1: Two placements of specialization in an evaluation control layer. (a) A trust head decides per example whether a cheap judge's ranking can be released or must be deferred to a larger judge; reward scores are never modified, and thresholds are calibrated and audited on disjoint data. (b) A rubric-derived router activates one criterion-family LoRA adapter on a shared frozen base. Placement (a) beats the largest judge at $0.42\times$ compute; placement (b), applied too early, loses 10 points to a single shared judge.

accuracy at $0.415\times$ the compute of the 8B judge alone—which scores 84.75% on the same metric. The biggest judge is not the best judge: its smaller siblings are right on 5.32% of the tasks it gets wrong, and the trust head finds those tasks using only stage-local score geometry, never treating the terminal model as an oracle. Crucially, the *naive* version of this idea does not work. Escalating on raw score margin—the cascade everyone builds first—yields 84.81% at $0.944\times$ compute: no accuracy gain, no compute savings. The entire effect lives in what signal governs the handoff, not in the existence of stages. And every reported cascade passes an exact one-sided 95% Clopper–Pearson audit on early-exit error, so the trust decision carries a statistical certificate, not a heuristic.

The specialization lever fails outright. On 99,952 rubric-conditioned examples from the Feedback Collection (Kim et al., 2024a), dividing one training corpus among eight criterion-family LoRA "judgelets" loses 10.05 accuracy points relative to a single shared judge and collapses audited coverage at a 5% risk target from 24.44% to 5.43%. A battery of controls shows the loss is not capacity (a rank-64 monolith matching the bank's stored parameters is 11.16 points *better*), not optimizer steps, and not learning rate. It is fragmented supervision: specializing too early divides the examples needed to learn the common task of judging. Initializing every specialist from a shared trained judge recovers test accuracy to 76.85%, 19.94 points above learning-rate-matched scratch training.

Together the two studies answer an architectural question that the community's instincts get backwards: *where should domain specialization live in an evaluation system—in the judge's weights, or in the rule that decides when its judgment can be trusted?* Our answer is conditional but directional. Specializing weights fragments learning; specializing the trust decision exploits complementary errors among already-trained judges—provided the early judges are genuinely cheap, a condition we show can fail.

Contributions.

1. **A risk-audited trust study** on RewardBench 2 with exhaustive stage subsets, four confidence baselines, disjoint fit/calibration/test partitions, and exact held-out risk audits, showing that a learned trust head—but not margin-based confidence—moves the accuracy–compute frontier, beating the largest judge at $0.415\times$ its cost (§5).
2. **Measured boundary conditions**: two settings where the same recipe fails to yield a Pareto improvement—near-equal-cost stages, and an accuracy rescue that costs more compute—formalized by a rescue/harm decomposition of when a cascade can beat its terminal judge (§5.3).
3. **A controlled specialization study** with locked criterion shift, a response-only control, a counterfactual rubric intervention, and capacity-, step-, and learning-rate-matched controls, isolating fragmented supervision—not sparsity, capacity, or optimization—as the cause of the specialist bank's failure, and shared pre-adaptation as the fix (§6).

2 TERMINOLOGY AND NOTATION

Evaluation-routing work draws vocabulary from several subfields (selective prediction, mixtures of experts, reward modeling), so we fix terms up front.

Table 1: Terminology and notation used throughout the paper.

Term / symbol	Meaning
Judge	A model that scores or ranks candidate outputs (an LLM evaluator or reward model).
Rubric (criterion) r	The written rule governing a judgment; the unit of policy identity in our splits.
Monolith	A single judge trained on the full corpus (one LoRA adapter, rank 8 or 64).
Judgelet	A criterion-family LoRA adapter on a frozen shared base; one of K in a bank, one active per example.
Router $g(r)$	TF-IDF + k-means assignment of a rubric to one of K criterion families.
Deferral (trust decision)	The per-example choice to *accept* a judge's verdict or *defer* it to a stronger stage (Chow, 1970; Madras et al., 2018).
Trust head (risklet)	A logistic probe on a stage's score geometry that predicts whether its top choice is correct; it accepts or defers, never altering scores.
Release / coverage Γ	A judgment is *released* when accepted automatically; coverage is the released fraction.
Selective risk R	Error rate among released judgments; we target 5% with one-sided 95% upper bounds.
Cascade	Judges ordered by cost; every example enters the cheapest stage, deferred examples continue.
C_{norm}	Parameter compute normalized so that running the terminal judge on every example equals 1 (Eq. 3).
Rescue / harm	Released early verdicts that are right where the terminal judge is wrong / wrong where it is right (Eq. 4).
Locked test / criterion shift	Untouched test split / examples whose complete rubric strings were held out of all training and calibration.

3 RELATED WORK

LLM judges and specialized evaluators. Prometheus trains open evaluators conditioned on references and user-defined rubrics (Kim et al., 2024a), and Prometheus 2 extends this to direct assessment and pairwise ranking (Kim et al., 2024b). Small-judge comparisons find pronounced domain dependence (Laddha et al., 2026); we start from that heterogeneity but share a base model across criterion-family adapters. Quantitative LLM Judges align a judge's score to human scores post hoc (Sahoo et al., 2025); our post-hoc heads instead predict correctness for routing.

Sparse experts and parameter-efficient adaptation. LoRA represents a downstream update with low-rank matrices on a frozen base (Hu et al., 2022); sparse mixture-of-experts models grow stored capacity while activating a subset per input (Fedus et al., 2022). Judgelets combine these deployment properties at coarser granularity: a rubric-derived router selects one complete adapter, so the assignment is inspectable and constant for all examples governed by the same criterion. Mixture-of-LoRA methods report gains from learned or task-labeled routing (Li et al., 2024; Feng et al., 2024); our controlled result identifies a failure regime when independently optimized experts divide a common judgment task.

Selective prediction, learning to defer, and cascades. The reject option dates to Chow (1970), with modern treatments in learning with rejection (Cortes et al., 2016) and selective classification (Geifman & El-Yaniv, 2017). Learning to defer trains the deferral policy jointly with a downstream expert (Madras et al., 2018; Mozannar & Sontag, 2020); our deferral target is another model rather than a human. For LLM evaluation specifically, Trust or Escalate provides cascaded selective evaluation with calibrated human-agreement guarantees (Jung et al., 2025), SCOPE studies risk-controlled pairwise judging (Badshah et al., 2026), and Xu et al. (2025) route uncertain reward-model comparisons to a larger generative judge. Cascades with confidence tuning and learned epistemic-risk estimators improve cost–accuracy tradeoffs (Zellinger & Thomson, 2024; Rabanser et al., 2025; Melo et al., 2026); cheap-first pipelines are older still (Viola & Jones, 2004). Uncertainty routing per se is therefore not our claim. Our question is orthogonal: given a fixed evaluation workload, does domain specialization belong in judge parameters or in the release boundary—and under what measured conditions does each placement pay?

Reward-model evaluation. RewardBench 2 evaluates factuality, focus, instruction following, math, safety, and ties (Malik et al., 2026). Skywork-Reward-V2 supplies a controlled 0.6B-to-8B ladder trained with one data recipe (Liu et al., 2026), letting model size vary without changing the family.

4 PROBLEM SETTING

Judging. Each example carries a task x, a candidate response y, an optional reference a, a rubric r from a collection $\mathcal{R} = \{r_1, \ldots, r_m\}$, and an ordinal label $z \in \{1, \ldots, L\}$. A monolithic judge models $p(z \mid x, y, a, r)$ with one set of trainable parameters. A specialist bank assigns each rubric to a family $g(r) \in \{1, \ldots, K\}$; family k owns a low-rank adapter ϕ_k on a frozen shared base θ:

$$p_k(z \mid x, y, a, r) \;=\; M_{\theta,\phi_k}(x, y, a, r), \qquad k = g(r). \tag{1}$$

The bank stores K adapters but activates one per example; $K{=}1$ recovers the monolith.

Trusting. For a confidence function c_k and threshold t_k, a family releases an automatic judgment when $c_k \geq t_k$. With $\hat{z}$ its predicted score, its selective risk and coverage are

$$R_k(t_k) \;=\; \Pr(\hat{z} \neq z \mid c_k \geq t_k), \qquad \Gamma_k(t_k) \;=\; \Pr(c_k \geq t_k). \tag{2}$$

Thresholds are chosen on calibration data and never retuned on development, locked-test, criterion-shift, or external data.

Cascading. For judges $M_1, \ldots, M_J$ ordered by cost, every example enters M_1 and deferred examples continue. With q_j the fraction reaching stage j and P_j its parameter count, normalized parameter compute is

$$C_{\text{norm}} \;=\; \frac{\sum_{j=1}^{J} q_j P_j}{P_J}. \tag{3}$$

C_{norm} compares architectures; it is not an energy estimate, and measured throughput is reported separately.

When can a cascade beat its biggest judge? Only through complementary errors. For two stages, let A denote early acceptance and C_j correctness of stage j. Relative to always using stage 2,

$$\Delta_{\text{acc}} \;=\; \underbrace{\Pr(A \cap C_1 \cap \neg C_2)}_{\text{rescue}} \;-\; \underbrace{\Pr(A \cap \neg C_1 \cap C_2)}_{\text{harm}}. \tag{4}$$

Routing is worthwhile only if released early verdicts retain more helpful small-judge disagreements than harmful ones, *and* the first stage is cheap enough that $P_1 + (1 - \Pr(A))\, P_2 < P_2$. This decomposition separates selective prediction (don't release errors) from the stronger claim we test: that a cascade can improve accuracy and compute simultaneously.

5 STUDY 1: LEARN WHEN TO TRUST THE JUDGE

We first test the trust placement, where our headline result lives: frozen judges of increasing size, and a learned rule deciding which verdicts to release.

5.1 SETUP

Judges and data. We use the four Skywork-Reward-V2 Qwen3 reward models (0.6B, 1.7B, 4B, 8B) (Liu et al., 2026), which score 61.39, 68.11, 75.44, and 77.91 on the official RewardBench 2 macro metric.[1] The cascade study uses micro top-choice accuracy on the 1,763 non-tie tasks; the 102 tie tasks remain in the official score but are excluded from head fitting, calibration, and binary cascade accuracy. All results average over 20 locked train/calibration/test repartitions (mean test size 616.8).

Trust heads. Reward-model margins are not comparably calibrated across skills, so for each non-terminal judge we fit a logistic *risklet* predicting whether that stage's top-ranked candidate is correct. Its inputs are the largest scores, gap, range, mean, variance, candidate count, winning position, and a one-hot skill family—14 inputs and an intercept, so its arithmetic cost is negligible relative to a 0.6B forward pass. The risklet either accepts the current ranking or defers the example; reward scores are never modified.

[1] The exact checkpoints are indexed in the official Skywork-Reward-V2 model collection. We reproduce the official subset metric on all 1,865 tasks (8,977 candidate completions).

Table 2: RewardBench 2 routing on 1,763 non-tie tasks, mean over 20 locked repartitions. "Exact pass" counts runs whose one-sided 95% Clopper–Pearson upper bound on aggregate early-exit error is at most 5%. Compute is normalized to running 8B on every task (Eq. 3).

Policy and stages	**Early cov. (%)**	**Accuracy (%)**	**Exact pass**	C_{norm}
8B only	0.00	84.75	–	1.000
Raw margin, 4B–8B	55.61	84.81	20/20	0.944
Isotonic margin, 4B–8B	48.86	84.84	19/20	1.011
Geometry probe, 0.6B–4B–8B	80.35	89.37	20/20	0.413
Risklet, 0.6B–8B	71.52	88.71	20/20	**0.360**
Risklet, 0.6B–4B–8B	**80.20**	**89.40**	**20/20**	0.415
Risklet, all four stages	80.20	89.40	20/20	0.476

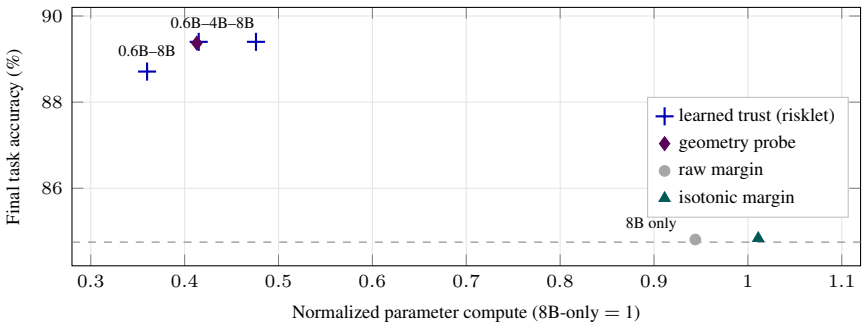

Figure 2: Learned trust routing forms a separate accuracy–compute frontier (mean operating points from Table 2; full stage-subset sweep and repartition variability in the artifact). Margin-based policies cluster at the 8B-only point; learned probes beat the largest judge at less than half its compute.

Audits. Fitting, threshold selection, and evaluation use three disjoint partitions. We require a 3% one-sided upper bound on calibration error, leaving a predeclared buffer before the 5% held-out target; the untouched test partition is audited with an exact one-sided 95% Clopper–Pearson bound on aggregate early-exit error. As a stage-selection check we enumerate all seven compute-ordered subsets terminating at 8B, with a calibration-only pruning rule: drop any non-terminal stage that accepts no calibration examples. Baselines—raw margin, logistic margin calibration, isotonic margin calibration, and a geometry-only probe without the skill indicator—use the same partitions and audit.

5.2 Results: margins fail, learned trust beats the biggest judge

The naive cascade buys nothing. Raw-margin escalation from 4B to 8B releases 55.61% of tasks early yet lands at 84.81% accuracy and 0.944 compute—effectively the 8B-only operating point. Logistic calibration preserves the margin ordering and finds the same optimum; isotonic calibration introduces ties and no gain, and even exceeds 8B-only compute (Table 2). If cascading were the trivial "cheap filter first" idea, this is where the story would end.

A learned trust head moves the frontier. The three-stage risklet cascade releases 80.20% of tasks before the terminal judge, reaching 89.40% accuracy at 0.415 compute—4.66 points above 8B alone at 41.5% of its cost, with the gain ranging from 3.21 to 6.19 points across the 20 repartitions (a stability range over reused tasks, not an independent-sample interval). Every run passes the exact risk audit. Complementary errors make this possible: relative to 8B alone, early exits rescue 5.32% of tasks and harm 0.66% (Eq. 4). An oracle over all four judges reaches 91.80%, and one restricted to the deployed 0.6B–4B–8B subset reaches 90.97%, so the learned head captures most of the available headroom. Unlike margin rules, the risklet uses multivariate score geometry to find cases where a smaller judge is right—including some the 8B judge misses—so the terminal model is never treated as an oracle. Removing the skill one-hot changes accuracy by only 0.03 points: nearly all the signal is stage-local score geometry, not the benchmark's skill label.

Stage membership is part of the fitted system. The 1.7B judge accepted no calibration example in any run, so the pruning rule removes it before locked evaluation; it would also have accepted no test example while still charging every example a forward pass. The two-stage 0.6B–8B cascade is the minimum-compute point (0.360); adding 4B costs 0.055 compute for 0.69 accuracy points. Which judges to include is a calibration-data decision, not an implementation detail.

5.3 Boundary conditions: when the recipe does not pay

Equation 4 has two requirements—complementary errors *and* a genuinely cheap first stage—and each can fail independently.

Complementary but not cheap. A diagnostic cascade of our Study-2 rank-8 and rank-64 judges (same 0.6B base, nearly identical inference cost) releases 27.98% at the first stage and moves accuracy from 77.50% to 77.52% while requiring $1.720\times$ the forward passes of rank-64 alone. Complementary errors are insufficient when the first decision is not materially cheaper.

Rescuing accuracy at a compute loss. An independent OpenAssistant DeBERTa base–large pair (OpenAssistant, 2023)—official RewardBench 2 scores of only 26.07 and 33.72, but strongly complementary errors—shows the mirror image. On the same 20 repartitions, a risklet accepts 30.05% at the base stage and lifts task accuracy from 40.14% (large alone) to 48.78%, an 8.64-point rescue, passing the exact audit in 19/20 runs (the geometry-only probe matches the accuracy and passes 20/20). But the 184M base is already 42% the size of the 435M terminal model, so $C_{\mathrm{norm}} = 1.122$: the accuracy effect transfers across model families, the Pareto improvement does not. The recipe's payoff is conditional on the cost ratio between stages, which must be measured, not assumed.

6 Study 2: Should the Judge Itself Be Specialized?

The alternative placement writes domain knowledge into judge weights. Organizations already maintain rubrics, checklists, and corrected decisions; the natural architecture compiles them into criterion-family experts. We test this with controls that separate sparsity, capacity, optimization, and supervision.

6.1 Setup

Data and splits. The corpus is 99,952 Feedback Collection examples spanning 996 criteria (Kim et al., 2024a), partitioned with rubric text as the unit of policy identity. A deterministic hash reserves 206 complete criteria (20,696 examples) as *criterion shift*; no example carrying one of these criteria touches training, calibration, or development. The remainder splits into 47,638 training, 9,441 calibration, 9,597 development, and 12,580 locked-test examples, with all responses sharing a criterion–instruction pair assigned to the same split. Feedback-Bench contributes 1,000 external examples; all 626 of its criterion strings occur somewhere in Feedback Collection, but 122 were excluded from adapter training and account for 203 external examples, which we report as the adapter-held subset. A content-hash audit found two duplicated pairs (four rows) with conflicting labels, neither crossing a split, criterion, or routed family.

Judges. All judges share the 0.6B Skywork-Reward-V2 Qwen3 base with its scalar reward head replaced by a five-class score head. The router represents criterion strings with word and bigram TF-IDF features fitted only on training-visible criteria; k-means defines K families and assigns new rubrics by the fitted vectorizer and centroids. We chose a lexical router because its assignments are directly inspectable; routing quality is not the experimental variable. We evaluate $K \in \{1, 4, 8\}$, with $K{=}1$ the monolith. Each rank-8 judgelet trains 5.05M parameters (0.84% of the instantiated evaluator); a rank-64 monolith approximately matches the bank's aggregate stored capacity (40.38M vs. 40.41M). Inputs place the criterion and five score descriptions before the task, response, and reference; because over 95% of inputs exceed 512 tokens while under 3.5% exceed 1,024, primary models use 1,024 tokens, with a 512-token variant retained as a truncation intervention and the rubric-free response-only control at 512.

Controls. The primary $K{=}8$ bank matches the monolith's aggregate optimizer budget (1,494 total family updates vs. 1,489 monolithic updates, unevenly distributed (98–489 updates per expert)). Three further banks isolate competing explanations: (i) *500 each*: every family cycled to exactly 500

Table 3: Direct-assessment results (accuracy %; higher is better except MAE). "κ" is quadratic-weighted kappa; "Shift" is held-criterion accuracy; "External" is Feedback-Bench.

Judge	Dev	Test	MAE	κ	Shift	External
Response only, 512	74.75	74.27	.281	.916	74.75	75.70
Monolith, rank 8	76.78	76.38	.250	.929	77.85	77.60
Monolith, rank 64	79.09	**77.50**	**.237**	**.934**	**78.97**	**82.10**

updates (4,000 total), an optimization control; (ii) *shared init.*: each expert initialized from the trained rank-8 monolith, then one family epoch at learning rate 5×10^{-5}, preserving shared judgment learning before the split; (iii) *matched LR*: scratch training for the same 1,494 family updates at 5×10^{-5}, isolating initialization within the family-adaptation recipe. Appendix A records every recipe.

Interventions and audits. The counterfactual rubric intervention replaces the criterion and all five score descriptions with a rubric sampled from another cluster in the same split, leaving task, response, reference, label, router assignment, and row identity unchanged: a response-only judge is invariant, a rubric-sensitive one should degrade. Release policies compare a global confidence threshold, per-family thresholds, and a Bonferroni-adjusted family policy, each selected by scanning the calibration risk–coverage curve; because the scan is adaptive, realized risk is evaluated on untouched splits with one-sided 95% Wilson upper bounds rather than treating the search as a prospective guarantee. Comparisons use paired bootstrap intervals with criterion ID as the resampling unit.

6.2 The rubric helps—and the judge really reads it

On the locked test, the full-context monolith beats the response-only control by 2.11 points (paired criterion-bootstrap interval $+1.48$ to $+2.75$ over 765 criterion IDs) and lowers MAE by 0.030; the gap grows to $+3.10$ points on held criteria ($+2.47$ to $+3.74$).[2] The external comparison is less decisive: a 1.9-point accuracy gain with interval -0.5 to $+4.4$ (p=0.138), though MAE falls from 0.270 to 0.239 (p=0.033); full-context accuracy is 79.30% on the 797 examples with training-visible criteria versus 70.94% on the 203 adapter-held examples. Raising monolith rank from 8 to 64 adds 1.11 test points, 1.12 shift points, and 4.50 external points (Table 3).

Two interventions confirm the rubric is causally used rather than decorative. *Counterfactual rubrics*: swapping in a wrong rubric while fixing every other field drops locked-test accuracy from 76.38% to 73.72% (-2.66 points; -3.26 to -2.09) and held-criterion accuracy by a similar margin—and the wrong-rubric judge falls *below* the response-only control, ruling out an explanation from extra benign context. *Truncation*: running the same adapter at 512 tokens cuts test accuracy to 73.82% (-2.56 paired points), statistically indistinguishable from response-only (-0.45; p=0.147), even as throughput rises from roughly 58 to 110 examples/s. The rubric benefit is real, but it lives in context the judge must actually see—and the strong response-only baseline shows the data also contain a response shortcut, which motivates the audited release policies below.

6.3 Specializing weights fragments supervision

The specialist bank fails badly. The eight-family bank's locked-test accuracy is 66.34%—10.05 points below the rank-8 monolith (criterion-bootstrap interval -10.87 to -9.22)—with MAE rising from 0.250 to 0.375 and a matching -10.13-point held-criterion gap. Expert quality is uneven: a few reach 71–72%, several sit between 56% and 61%. Four families narrow the loss without removing it (-4.57 points), and the pattern in Table 4 is monotone under independent one-epoch training: active adapter size fixed, stored parameters rising, accuracy falling as data are split more finely. The bank is not ignoring its rubrics—counterfactual replacement still costs it a further 1.72 points—but it learns the *common* task of judging less well once supervision is divided.

Ruling out the mundane explanations. *Capacity*: rank 64 stores 40.38M adapter parameters against the bank's 40.41M yet is 11.16 test points better ($+10.28$ to $+12.07$), so stored capacity is not the limiting factor—the capacity-matched comparison points in the *opposite* direction from the sparse bank. *Optimizer steps*: cycling every expert to 500 updates spends 2.7× the update budget to recover

[2]Throughout, "points" are percentage points and intervals are 95% intervals unless stated otherwise.

Table 4: Criterion-family granularity and optimization controls. Adapter counts in millions. Coverage is the locked-test release rate under the calibration-selected Bonferroni policy at a 5% risk target; † marks a policy whose one-sided risk upper bound exceeds 5%.

Design	Stored	Active	Test	Shift	External	Coverage (%)
Monolith r8	5.05	5.05	76.38	77.85	77.60	24.44
K=4, scratch	20.21	5.05	71.81	73.78	74.90	17.79
K=8, scratch	40.41	5.05	66.34	67.72	69.10	5.43
K=8, scratch, LR 5×10^{-5}	40.41	5.05	56.91	58.47	58.00	0.00
K=8, 500 each	40.41	5.05	72.56	73.74	79.40	11.76†
K=8, shared init.	40.41	5.05	76.85	78.14	78.90	24.90
Monolith r64	40.38	40.38	**77.50**	**78.97**	**82.10**	**30.14**

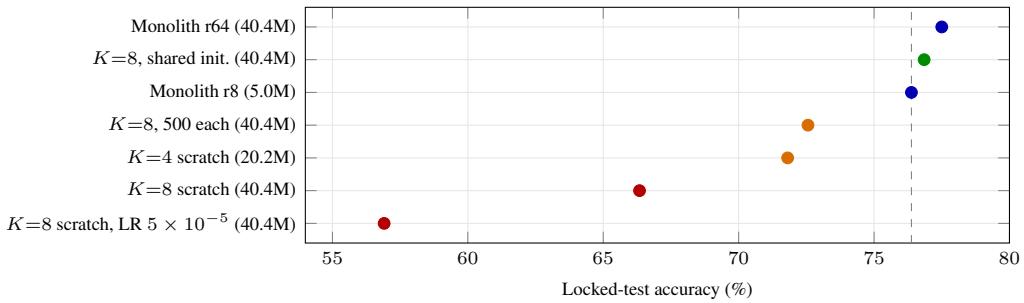

Figure 3: Locked-test accuracy across specialization and optimization controls (Table 4); dashed line marks the rank-8 monolith. Fragmenting supervision across scratch-trained specialists costs up to 10 points at fixed active capacity; shared initialization before the split recovers the loss. Criterion-bootstrap intervals for each contrast are given in the text.

6.22 points, still 3.82 short of the rank-8 monolith; under-training matters but is not the whole story. *Learning rate*: scratch training at the shared-init rate (5×10^{-5}) collapses to 56.91%, so the shared-init advantage below cannot be an artifact of its rate. This low-rate scratch run clearly underfits and says nothing about good scratch rates in general; what it rules out is the specific confound.

Shared pre-adaptation nearly closes the gap. Initializing each expert from the trained rank-8 monolith and running one family epoch yields 76.85% test accuracy—10.51 points above independent training and 0.47 points above the monolithic initialization itself ($+0.13$ to $+0.82$)—with 78.14% on held criteria and 78.90% externally. A wrong rubric still costs it 3.28 points, so the recovery does not come from ignoring family-specific rule text. (On the external set the 500-each control is 0.50 points higher, but the interval spans -2.65 to $+1.70$; that comparison is unresolved and should not be read as reversing the locked-test result.) The mechanism, then, is supervision: specializing weights *before* the shared judgment task is learned divides the examples needed to learn it; specializing *after* is nearly free.

6.4 Trust is fragmented too: audited coverage

The bottleneck is only relieved by judgments that can *safely* leave the review queue, so audited coverage is the deployment-relevant metric—and it exposes failures that accuracy hides. **Global calibration overstates safe coverage**: the global threshold accepts 37.47% of locked-test examples but its 5.38% risk upper bound misses the 5% target, while per-family thresholds pass at 32.06% and the conservative Bonferroni policy at 24.44% (26.13% coverage with a 3.05% bound under criterion shift; Table 5). **Fragmentation collapses trustworthy coverage**: for the eight-family bank, neither global nor ordinary family calibration passes at all, and the passing Bonferroni policy releases only 5.43% of examples—a $4.5\times$ collapse against the monolith—while shared initialization restores 24.90% at a 3.48% bound; the 500-each bank selects 11.76% but misses its bound, the low-rate scratch bank releases nothing, and rank 64 reaches 30.14%. **Audits do not survive corrupted rubrics**: under wrong-rubric replacement the monolith's ordinary family policy sees its risk bound rise to 5.30% even as coverage falls, and only the Bonferroni policy stays under target at 20.95%

Table 5: Locked-test release policies with calibration-only thresholds (5% selective-risk target; one-sided 95% Wilson upper bounds).

Judge and policy	Coverage (%)	Risk (%)	Upper (%)	Pass
Response only, global	29.93	4.91	5.53	No
Response only, family Bonf.	18.01	2.69	3.31	Yes
Rubric monolith, global	37.47	4.84	5.38	No
Rubric monolith, family	32.06	4.04	4.58	Yes
Rubric monolith, family Bonf.	24.44	2.99	3.54	Yes

coverage; truncation likewise cuts Bonferroni coverage from 24.44% to 19.77%. A release rule audited with valid rule text cannot be assumed safe after that text is corrupted.

7 Discussion: A Conditional Design Rule

The two studies dissociate two kinds of specialization that are often grouped together, and the resulting rule is directional but conditional:

Share judgment learning by default. Specializing judge weights too early divides the supervision needed to learn the common task of judging; the cost appears in accuracy (10 points) and, more damagingly for deployment, in audited coverage ($4.5\times$). If per-domain adaptation is required, pre-adapt on the shared task first—shared initialization recovers nearly the entire capacity-matched gap while remaining rubric-sensitive.

Specialize the trust decision—after measuring its risk and cost. A 15-parameter head over frozen judges' score geometry beats the biggest judge at $0.42\times$ compute with exact risk certificates, because judges of different sizes make complementary errors and the head can find them. But the same recipe yields nothing when cascading near-equal-cost judges, and yields accuracy without compute savings when the first stage is 42% the size of the last. Both conditions of Eq. 4—complementarity and a steep cost ratio—must hold, and both are cheap to measure before deployment.

Aggregate confidence is the wrong trust signal. Margins, calibrated or not, do not move the frontier; global thresholds that look fine on average violate the risk target; audited thresholds silently stop being valid when rubric text is corrupted. Trust must be learned per stage, calibrated per family, and audited on untouched data.

Limitations. Feedback Collection is synthetic, and the strong response-only control shows a model can exploit its regularities without reading the rubric—a property of the data our interventions expose but do not eliminate; neither substitutes for judgments by domain professionals on naturally occurring work. Criterion shift holds out complete rubric strings but does not simulate policy revision over time, adversarial rules, or population change; Feedback-Bench is external at the example level yet all its criteria occur in Feedback Collection, so we report the adapter-held subset and do not call it wholly new-criterion. Risk results are retrospective locked-split audits: Wilson bounds do not establish prospective guarantees under distribution shift, our threshold procedure scans candidate cutoffs, and high-impact applications need pre-specified risk control, monitoring, and mandatory human gates. The 20 RewardBench repartitions reuse the same tasks and measure sensitivity to partition assignment, not sampling uncertainty; we therefore use exact binomial bounds within each held-out partition and avoid treating across-seed quantiles as population intervals. The study also uses English lexical routing, one direct-assessment base family, and parameter count as a compute proxy; Appendix B states these boundaries in detail.

8 Conclusion

The evaluation bottleneck will not be closed by ever-bigger judges: the biggest judge is measurably not the best judge, and a learned trust decision over cheaper judges beats it at $0.42\times$ its cost with exact risk certificates, while specializing the judges themselves fragments the supervision that judgment learning needs. The design rule is conditional: share judgment learning until data justify a split, and specialize the audited trust boundary once its complementarity and cost ratio are measured.

Ethics Statement

This work uses public model outputs and benchmark annotations; it collects no new human-subject data. Automated evaluators may reduce review cost, but they may also encode contested policies, centralize managerial judgment, or invite more confidence than a benchmark audit warrants. Criteria should remain versioned and attributable, and affected users should have a route of appeal. Coverage measured on these benchmarks is not a reason to remove human review from legal, medical, employment, or other high-impact decisions.

AI Use Statement

In this work, we used generative AI tools to edit the manuscript for readability, to check compliance with the conference formatting instructions, and to format bibliographic entries and verify them against their original sources. We did not use generative AI tools to propose or refine the hypotheses, design the methodology or experiments, implement the methods, generate or clean the data, or interpret the results; the remaining required-disclosure tasks (synthetic-data generation, theoretical-model development, formulation or proof of mathematical claims, translation, and qualitative data analysis) are not applicable to this work. We reviewed every AI-assisted edit, and all quantitative claims in the paper were checked by the authors against the experiment logs. We take responsibility for the final content of this work, including text, claims, and artifacts produced with the aid of generative AI.

Reproducibility Statement

All splits use seed 20260728. Direct-assessment adapters train with batch size 4, gradient accumulation 8, and 1,024-token inputs unless marked otherwise; Appendix A records the initialization, learning rate, and optimizer updates for every specialization control. Experiments ran with PyTorch 2.8.0, CUDA 12.8, NCCL 2.27.3, and two RTX 5090 GPUs (32,607 MiB each). An anonymized artifact accompanies the submission with selected prediction-level outputs, split manifests, training metadata, threshold audits, paired-bootstrap results, all 20-seed outputs for each reported cascade configuration, and dataset file sizes with SHA-256 hashes.

References

Gene M. Amdahl. Validity of the single processor approach to achieving large scale computing capabilities. In *Proceedings of the April 18–20, 1967, Spring Joint Computer Conference*, pp. 483–485, 1967.

Sher Badshah, Ali Emami, and Hassan Sajjad. SCOPE: Selective conformal optimized pairwise LLM judging. In *International Conference on Machine Learning*, 2026.

C. K. Chow. On optimum recognition error and reject tradeoff. *IEEE Transactions on Information Theory*, 16(1):41–46, 1970.

Corinna Cortes, Giulia DeSalvo, and Mehryar Mohri. Learning with rejection. In *International Conference on Algorithmic Learning Theory*, pp. 67–82, 2016.

William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

Wenfeng Feng, Chuzhan Hao, Yuewei Zhang, Yu Han, and Hao Wang. Mixture-of-LoRAs: An efficient multitask tuning method for large language models. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 11371–11380, 2024.

Yonatan Geifman and Ran El-Yaniv. Selective classification for deep neural networks. In *Advances in Neural Information Processing Systems*, volume 30, 2017.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Jaehun Jung, Faeze Brahman, and Yejin Choi. Trust or escalate: LLM judges with provable guarantees for human agreement. In *International Conference on Learning Representations*, 2025.

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*, 2024a.

Seungone Kim, Juyoung Suk, Shayne Longpre, Bill Yuchen Lin, Jamin Shin, Sean Welleck, Graham Neubig, Moontae Lee, Kyungjae Lee, and Minjoon Seo. Prometheus 2: An open source language model specialized in evaluating other language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4334–4353, 2024b.

Anish Laddha, Nitesh Pradhan, and Gaurav Srivastava. SLMJury: Can small language models judge as well as large ones? *arXiv preprint arXiv:2606.07810*, 2026.

Dengchun Li, Yingzi Ma, Naizheng Wang, Zhengmao Ye, Zhiyuan Cheng, Yinghao Tang, Yan Zhang, Lei Duan, Jie Zuo, Cal Yang, and Mingjie Tang. MixLoRA: Enhancing large language models fine-tuning with LoRA-based mixture of experts. *arXiv preprint arXiv:2404.15159*, 2024.

Chris Yuhao Liu, Liang Zeng, Yuzhen Xiao, Jujie He, Jiacai Liu, Chaojie Wang, Rui Yan, Wei Shen, Fuxiang Zhang, Jiacheng Xu, Yang Liu, and Yahui Zhou. Skywork-Reward-V2: Scaling preference data curation via human–AI synergy. In *International Conference on Learning Representations*, 2026.

David Madras, Toniann Pitassi, and Richard Zemel. Predict responsibly: Improving fairness and accuracy by learning to defer. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Saumya Malik, Valentina Pyatkin, Sander Land, Jacob Morrison, Noah A. Smith, Hannaneh Hajishirzi, and Nathan Lambert. RewardBench 2: Advancing reward model evaluation. In *International Conference on Learning Representations*, 2026.

Sébastien Melo, Gaël Varoquaux, and Marine Le Morvan. Epistemic uncertainty quantification to improve decisions from black-box models. In *International Conference on Learning Representations*, 2026.

Hussein Mozannar and David Sontag. Consistent estimators for learning to defer to an expert. In *International Conference on Machine Learning*, pp. 7076–7087, 2020.

OpenAssistant. DeBERTa-v3 reward-model checkpoints. `https://huggingface.co/OpenAssistant/reward-model-deberta-v3-base`, 2023. Base and large public model cards.

Stephan Rabanser, Nathalie Rauschmayr, Achin Kulshrestha, Petra Poklukar, Wittawat Jitkrittum, Sean Augenstein, Congchao Wang, and Federico Tombari. Gatekeeper: Improving model cascades through confidence tuning. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Aishwarya Sahoo, Jeevana Kruthi Karnuthala, Tushar Parmanand Budhwani, Pranchal Agarwal, Sankaran Vaidyanathan, Alexa Siu, Franck Dernoncourt, Jennifer Healey, Nedim Lipka, Ryan A. Rossi, Uttaran Bhattacharya, and Branislav Kveton. Quantitative LLM judges. *arXiv preprint arXiv:2506.02945*, 2025.

Paul Viola and Michael J. Jones. Robust real-time face detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

Zhenghao Xu, Qin Lu, Qingru Zhang, Liang Qiu, Ilgee Hong, Changlong Yu, Wenlin Yao, Yao Liu, Haoming Jiang, Lihong Li, Hyokun Yun, and Tuo Zhao. Ask a strong LLM judge when your reward model is uncertain. In *Advances in Neural Information Processing Systems*, volume 38, 2025.

Michael J. Zellinger and Matt Thomson. Efficiently deploying LLMs with controlled risk. *arXiv preprint arXiv:2410.02173*, 2024.

A Training Recipes for the Specialization Controls

Table 6: Training recipes for the direct-assessment specialization controls. "Shared updates" are full-corpus optimizer steps completed before any family split; "family updates" are summed across routed experts.

Design	Initialization	LR	Shared updates	Family updates
Monolith r8	base	2×10^{-4}	1,489	–
K=8, scratch	base	2×10^{-4}	0	1,494
K=8, scratch, matched LR	base	5×10^{-5}	0	1,494
K=8, 500 each	base	2×10^{-4}	0	4,000
K=8, shared init.	monolith r8	5×10^{-5}	1,489	1,494
Monolith r64	base	2×10^{-4}	1,489	–

B Additional Limitations

The compiler uses English lexical features and one base-model family; dense semantic routing, multilingual criteria, joint multi-expert training, and alternative parameter-efficient modules may change the specialization tradeoff. The ordinal score head produces no written evidence or critique, so released judgments carry a score and a confidence but no rationale. Risk results are retrospective locked-split audits; Wilson and Clopper–Pearson bounds quantify uncertainty within a held-out split but do not establish prospective guarantees under distribution shift, and our threshold procedure scans candidate cutoffs on calibration data, which is why we describe thresholds as calibration heuristics and audit realized risk on untouched data. Normalized parameter compute is not a measurement of energy or monetary cost, and the workflow calculation in Appendix C is not evidence of achieved organizational throughput.

C Workflow Sensitivity

Selective coverage is the fraction of decisions that can leave a review queue. For interpretation only, we substitute measured coverage c and accepted-set risk into an Amdahl-style sensitivity model (Amdahl, 1967),

$$S \;=\; \big\{(1-c) + c/s + e + crk\big\}^{-1},$$

where s is the automatic-review speedup, e the evaluator overhead, r the accepted-set risk, and k the escaped-error rework factor (e and crk are diagnostic extensions of the serial-fraction term). At s=20, e=0.02, and k=1, moving from the response-only Bonferroni policy to the rank-8 and rank-64 rubric-aware policies raises implied speedup from $1.171\times$ to $1.258\times$ and $1.345\times$. This is not a field measurement; the artifact provides the full sensitivity grid.