Which Data Is Worth Training On?

A self-improving predictor that tells you if RL data will improve your model — *before* you train

---

TAP · Inference-Time Compute Hackathon

June 20, 2026

The Problem (in one slide)

- Labs are buying huge amounts of post-training data from experts.
- But most of it barely moves the model — and the only way to know today is to train on it and check.
- That means burning GPUs and hours on data that may not help at all.

The question

Can we tell whether a dataset will improve the model cheaply, before training?

"lift" = how much accuracy goes up after training on the data.

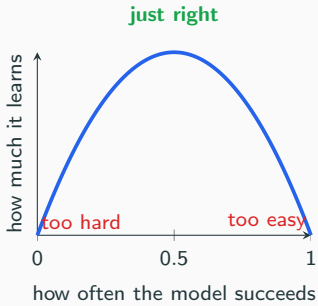
Our Idea: a side-model that predicts the lift

We train a separate, cheaper, self-improving model that learns one function:

$$f(\text{GRPO features}) \rightarrow \text{predicted lift}$$

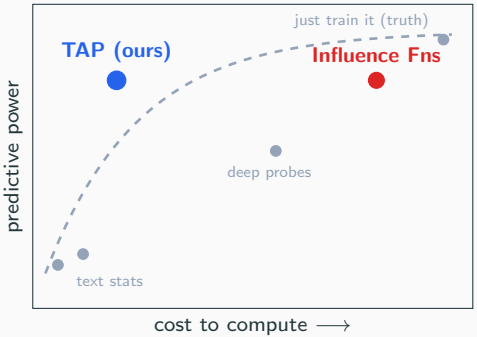
From the traces of a few RL rollouts, it tells us how much the LLM would improve if trained on that data — without ever training the LLM.

- Self-improving: every real RL run is a new lesson for it.
- Universal: works for any subject we hill-climb on with RLVR.
- Cheap: a few rollouts — no gradient steps on the big model.



Why it's learnable: the top feature — how much the model's attempts **disagree** — peaks right where learning happens.

Cheap and Accurate — Comparing to the SOTA

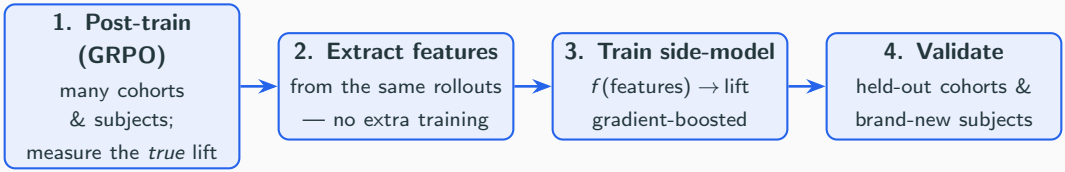


Usually cheap means weak. Our disagreement signal is cheap *and* strong — it's literally what RL learns from.

vs. Influence Functions (SOTA)

- +68% lift over random (+118% on unseen)
- Cost = ϵ — inference-only, no training
- IF pays training-scale cost (gradients / Hessians)

Our Process: turning RL runs into a predictor



- **The features (step 2), all cheap:** whether the attempts disagree (the top one), whether the model already knows it, and its relevance to the target.
- **Scope:** proved on Qwen2.5-Math, then generalized to Qwen3-1.7B across 4 subjects; next is the live RL loop.
- **The point:** built entirely from runs we do anyway — no extra LLM training to make a prediction.

A Worked Example: one cohort, start to finish

1. A cohort is just a handful of problems. **2.** The model attempts each 8 times; we only mark right/wrong:

"If $3x + 7 = 22$**, find** x**"**
✓✗✓✓✗✓✗✓ 5/8 ⇒ they disagree: LEARN
"What is $2 + 2$**?"**
✓✓✓✓✓✓✓✓ 8/8 ⇒ already known: skip
"Prove the Riemann Hypothesis"
✗✗✗✗✗✗✗✗ 0/8 ⇒ no foothold: skip

The signal: how many problems land in the "disagree" zone — fed (with the other features) to our model.

3. What we predict — the cohort's NLL-lift:

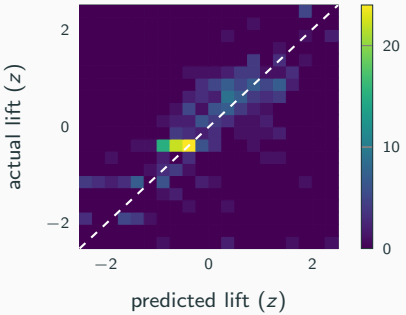
NLL = the model's *loss* (how surprised it is by the right answer). A cohort that truly teaches the model lowers this loss. We predict that drop — smooth and reliable, where small-probe accuracy is too jumpy.

4. Train & check:

cohort	predicted	actual
mostly disagree	+2.1	+2.3
some disagree	+1.0	+0.8
scrambled	0.0	0.0

predicted ≈ actual NLL-lift (loss drop)

Did It Work?



predicted vs. actual lift, all 320 cohorts — density hugs the diagonal.

Across 4 subjects — science, code, math, general knowledge (Qwen3-1.7B):

- 78% of the time, shown two datasets, our model ranks them in the correct order.
- Pick the top 10% by our predictor ⇒ +68% more improvement than random.
- That's nearly 70% of the way to a perfect, noise-free oracle — and it beats picking by the noisy measured results.

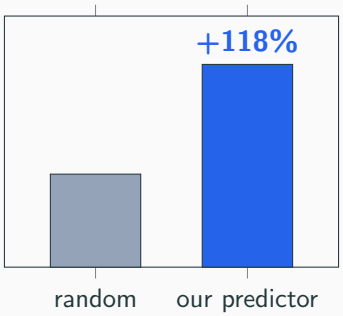
320 datasets, leave-one-out. "Improvement" = capability gain (NLL-lift); oracle = noise-free (seed cross-fit). Near the measurement ceiling — more runs →∼0.9.

It Generalizes: works on subjects it's never seen

We trained the predictor on 3 subjects, then tested it on the 4th — completely unseen:

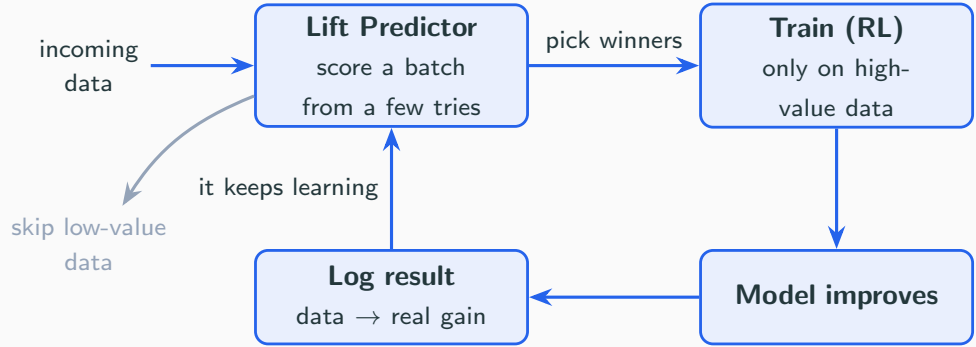
- It ranks the unseen subject's datasets almost as well as if it had trained on it — barely any drop.
- Because it learns one universal rule — more disagreement → more learning — not subject-specific tricks.
- In that unseen subject, its data picks beat random by +118% (right).

Payoff: forecast a brand-new domain's value with zero labels collected there.



improvement from picking data in a subject it has **never seen**

The Vision: a predictor inside the RL loop



Before spending GPUs on a batch, score it. Train the winners, skip the rest — and every real run makes the predictor smarter.

Bonus: If Inference Were Exponentially Cheap

What would you build if inference were exponentially cheap? Spend cheap inference to save expensive training.

- Our only cap is label noise — cured by more seeds (cheap re-runs); more seeds → sharper predictor (right).
- Spend them on the noisy subjects; the maxed ones (science) need none.
- Continual RL becomes practical: cheaply spot which of your users' real-world traces are high-lift and train on only those — the model keeps improving from live use while preventing loss of plasticity.

seeds →	3 (now)	10	∞
science *(maxed)*	0.92	0.96	0.98
code	0.68	0.82	0.90
math	0.78	0.91	$\sim$1.0
general	0.68	0.84	$\sim$1.0
avg	**0.78**	**0.89**	**$\sim$0.95**

more seeds → sharper ranking; 3 measured, rest projected.

Why It Matters

- Save compute — stop training on data that won't help.
- Buy smarter — score a vendor's dataset *before* paying to train on it.
- Sharper RL — a loop that automatically steers toward what improves the model.
- Compounding — it gets better the more it's used; every run is a new lesson.

Status

Validated across 4 subjects on Qwen3-1.7B · transfers to new subjects with no retraining · the loop is the next build.

Appendix: Predictor Features (23) & Importance

Learnability *(dominant group)*

- **adv_std** — spread of GRPO advantages (top feature)
- **frac_nondegenerate** — % prompts whose attempts disagree (r=0.98 to adv_std)
- adv_absmean; frac_all_correct (−); frac_all_wrong (−)
- reward_mean, reward_std, pass_rate
- group_passrate_mean, group_passrate_std

Relevance

- target_similarity — cohort vs eval probe

Shape & size

- len_mean, len_std, n_groups (cohort size)

Familiarity / confidence

- mean_logprob, surprisal_mean, entropy_mean
- logprob_p10 / p50 / p90
- confidence_slope (early vs late)
- redundancy_mean (mean token prob)
- min_logprob_mean (hardest token)

Importance (most → least)

1. frac_nondegenerate / adv_std — disagreement signal
2. frac_all_wrong, frac_all_correct — dead-ends
3. pass_rate, group_passrate_mean
4. familiarity / length — secondary

Monotone: disagreement ↑⇒ lift ↑; all-wrong / all-correct ↑⇒ lift ↓.

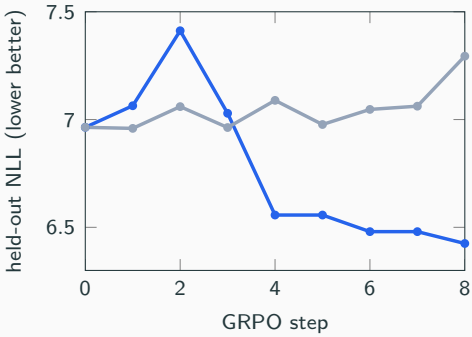
Model: gradient-boosted trees on these 23 (ridge benchmarked as baseline). Computed but excluded: n_rollouts, len_max, len_cv, reward_gap, distinct_frac.

Appendix: The Loop Works (v3 prototype)

A live loop, end to end: each step the online predictor re-scores 4 fresh cohorts against the *current* model, trains the winner, repeats — 8 cumulative GRPO steps (weights persist).

- online NLL -0.539 (improves) vs random $+0.331$ (worsens).
- Advantage at step 8: $+0.869$ nats.
- Predictor learns online: cold-starts as adv_std, ridge engages by step 4 (predicted $\approx$ realized).

Qwen3-1.7B · CompMath-MCQ · 20 cohorts (4 problems, 8 rollouts) · 64-item probe.



online (ours) drives loss down; **random** drifts up