Transferability for General Reasoning: An Automated Curriculum for Multi-Domain RLVR

Yongjin Yang[1] Jiarui Liu[2] Yinghui He[3] Lechen Zhang[4]
Bernhard Schölkopf[5,6] Zhijing Jin[1,6,7]
Jinesis Lab, University of Toronto & Vector Institute[1]
Carnegie Mellon University[2] Princeton University[3]
University of Illinois Urbana-Champaign[4] ELLIS Institute Tübingen[5]
Max Planck Institute for Intelligent Systems[6] EuroSafeAI[7]
{yjyang,zjin}@cs.toronto.edu

github.com/YangYongJin/transfer-aware-curriculum

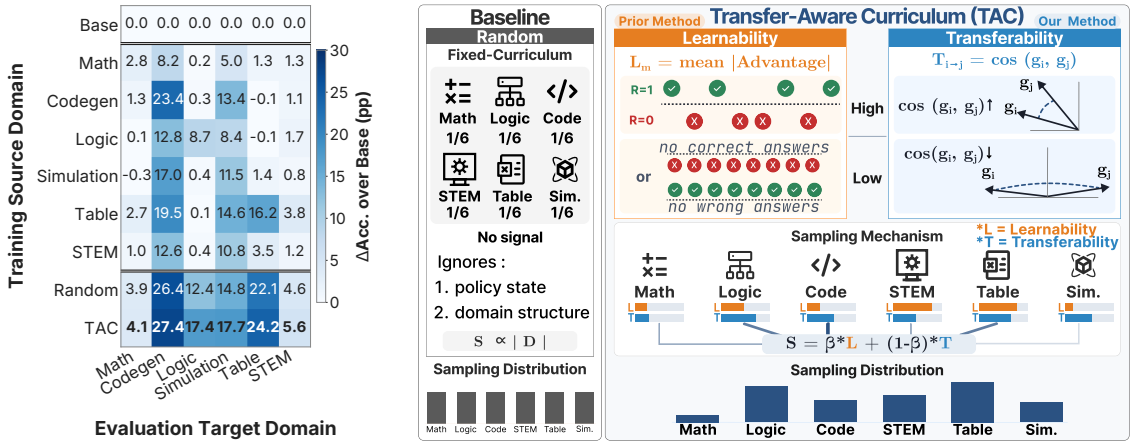
Reinforcement learning with verifiable rewards (RLVR) has been extended from single-domain training to multi-domain reasoning suites spanning mathematics, programming, and science. However, the training curriculum (how often each domain is sampled) is typically fixed or hand-tuned, even though reasoning skills transfer unevenly across domains. Existing learnability-based curricula adapt to where the policy is currently improving, but are blind to whether a gradient step on the selected domain benefits the remaining domains. In this paper, we propose Transfer-Aware Curriculum (`TAC`), a bandit-style online curriculum that prioritizes domains whose updates broadly benefit the rest of the training suite. `TAC` repurposes signals already produced by RL training: per-domain advantages capture local learnability, and projected gradients, taken from the GRPO step being computed, estimate cross-domain transferability via gradient-geometry alignment, at negligible cost ($<1\%$ wall-clock overhead). Across a six-domain reasoning suite, `TAC` achieves the best macro-averaged accuracy on both `Qwen3-1.7B` and `Llama3.2-3B`, outperforming proportional random sampling, a hand-designed schedule, and a learnability-only bandit, and improving over the last of these by up to 2.8 points (10% relative). Ablations show performance degrades sharply when the transferability term is removed, and `TAC` remains robust on imbalanced training mixtures where learnability-only curricula over-commit to dominant domains. Our findings establish cross-domain transferability as a key signal for curriculum design in multi-domain RLVR.

1 Introduction

Reinforcement learning with verifiable rewards (RLVR) has become a central tool for improving the reasoning capabilities of large language models (LLMs), producing substantial gains on benchmarks where correctness can be checked automatically [17, 50]. Motivated by these successes, recent work has extended RL training beyond single-domain setups toward broader multi-domain reasoning suites spanning domains such as mathematics, programming, and science [36, 11]. The goal of this line of work is a single policy that reasons competently across heterogeneous tasks rather than one specialized in a narrow slice.

Building a strong multi-domain reasoner, however, is not merely a matter of pooling data from every domain. Recent analyses show that reasoning training transfers unevenly across tasks, with effects depending strongly on the source domain [20, 30, 11]. We observe the same pattern in our own setting (Figure 1a): RL-training on different single domains yields markedly different transfer profiles across the remaining domains. For example, given the same training budget, RL on *table* improves *simulation* accuracy by 14.6 percentage points, while RL on *math* improves it by only 5.0. Methodological responses have so far focused on optimization- and loss-level interventions: per-domain gradient alignment [31] or per-task loss reweighting for multi-task GRPO [45]. The orthogonal question of *which domain to sample at each step*, and how that choice should depend on cross-domain transferability, has received far less attention.

Existing curricula for LLM reasoners [8, 55, 22] operate at the sampling level, but answer only half of this question. They prioritize domains whose on-policy advantages [8, 55] or reward variances indicate active learning [22], telling us *which domain the policy can presently learn from*, but not whether a gradient step on that domain also *benefits the remaining training domains*. A highly learnable domain may be narrowly



(a) Cross-domain transfer matrix.

(b) TAC vs. baseline and prior-work selection signals.

Figure 1: (a) Each cell shows the accuracy gain (pp) on a target domain after RL-training on a single source domain (1,000 queries, two epochs). Off-diagonal transfer varies sharply by source. The bottom rows show that TAC improves over *Random* across all six domains. **(b)** TAC combines a learnability signal with a gradient-cosine transferability signal that captures whether updates on a domain align with those of the other domains, mixed into per-arm scores via a single coefficient β.

scoped, so that gains on it fail to transfer; a less obviously learnable domain may nonetheless yield updates whose direction broadly improves the policy. A curriculum blind to cross-domain transfer can over-commit to locally rich but globally narrow domains, leaving a large share of the achievable cross-domain improvement on the table.

Contributions. In this paper, we propose Transfer-Aware Curriculum (TAC), a curriculum for multi-domain RL that introduces *cross-domain transferability* as a first-class signal for domain selection. While prior curricula prioritize domains the policy can presently learn from, TAC additionally asks whether a gradient step on a selected domain benefits the *remaining* training domains, and steers sampling accordingly. Figure 1a previews the result: uniform multi-domain sampling (*Random*) already outperforms any single-source curriculum on every target, and TAC widens this gap further across all six domains. The two signals come apart in practice: the most *learnable* domain is often not the most *transferable*. Notably, *math*, the domain RLVR leans on most, ranks among the *least* transferable in our suite, and TAC down-weights it accordingly. Our specific contributions are as follows:

1. **Transfer-Aware Bandit Curriculum.** We formulate multi-domain RL training as a multi-armed bandit over domains (§2.3), whose per-arm feedback combines an on-policy learnability term (§3.1) derived from GRPO advantages with a novel gradient-based transferability term, mixed via a single coefficient β (§3.3). The resulting method, TAC, jointly prioritizes domains that are currently learnable *and* broadly beneficial to the rest of the training set.

2. **Self-Supervised Transferability Signal.** We introduce a gradient-geometry estimator of cross-domain transferability (§3.2). At each training step, we maintain a per-domain exponential moving average of projected gradients; every K_c steps, the pairwise cosine similarities between these EMAs serve as the transferability signal fed back to the bandit. The signal is computed entirely from gradients already produced by RL training, requiring no held-out probes, extra rollouts, or oracle annotations, and adapts with the policy as it evolves.

3. **Empirical Gains on Multi-Domain Reasoning.** Across a six-domain reasoning suite spanning mathematics, programming, logic, simulation, tables, and stem, TAC consistently outperforms proportional sampling, a hand-designed math-to-others schedule, and a learnability-only bandit, improving macro-averaged accuracy across 14 evaluation benchmarks by 1.6–2.8 points (up to 10% relative) on `Qwen3-1.7B` and `Llama3.2-3B` at $<1\%$ wall-clock overhead (§4, Appendix C.2). Ablations isolate the transferability term, and TAC remains robust under data-budget skew where learnability-only curricula over-commit to dominant domains.

2 Preliminaries

2.1 Multi-Domain Reinforcement Learning for Reasoning

Let $\mathcal{D} = \{D_1, \ldots, D_M\}$ be a collection of reasoning domains, where each D_m consists of prompt–answer pairs (x, y^*) with verifiable targets. Given a query x, the policy π_θ generates a response $o \sim \pi_\theta(\cdot \mid x)$, evaluated by a sparse correctness reward $r(o, y^*) = \mathbb{I}[\mathrm{Ans}(o) = y^*]$. Multi-domain RL seeks a single policy maximizing expected reward under a sampling distribution $\mu \in \Delta^{M-1}$ over domains:

$$J(\theta;\, \mu) = \sum_{m=1}^{M} \mu_m \, \mathbb{E}_{(x,y^*)\sim D_m,\, o\sim\pi_\theta(\cdot|x)} \big[r(o, y^*) \big]. \tag{1}$$

2.2 Group Relative Policy Optimization (GRPO)

We optimize θ with Group Relative Policy Optimization (GRPO; [50]). For a query x, GRPO draws K rollouts $\{o^{(k)}\}_{k=1}^{K}$ from the old policy $\pi_{\theta_{\text{old}}}$, where each $o^{(k)} = (o_1^{(k)}, \ldots, o_{|o^{(k)}|}^{(k)})$ is the token sequence of a sampled response, and computes group-normalized advantages

$$A^{(k)} = \frac{r^{(k)} - \bar{r}}{\sigma_r + \epsilon}, \qquad \bar{r} = \tfrac{1}{K}\textstyle\sum_k r^{(k)}, \quad \sigma_r = \sqrt{\tfrac{1}{K}\textstyle\sum_k (r^{(k)} - \bar{r})^2}, \tag{2}$$

yielding the clipped surrogate objective

$$\mathcal{L}_{\text{GRPO}}(\theta) = -\mathbb{E}_x \left[\frac{1}{K} \sum_k \sum_t \min\!\big(p_t^{(k)} A^{(k)},\ \mathrm{clip}(p_t^{(k)}, 1 \pm \varepsilon)\, A^{(k)} \big) \right], \tag{3}$$

where $p_t^{(k)} = \pi_\theta(o_t^{(k)} \mid x, o_{<t}^{(k)}) / \pi_{\theta_{\text{old}}}(o_t^{(k)} \mid x, o_{<t}^{(k)})$ is the per-token importance ratio. We follow the DAPO recipe [60] and omit the KL regularizer used in the original formulation.

2.3 Curriculum as a Multi-Armed Bandit

A fixed sampling mixture μ is generally suboptimal: under a given policy, the optimization signal each domain provides, and how well its updates transfer to the others, drifts as training proceeds. We therefore adapt $\mu^{(t)}$ online from the training history $\mathcal{H}_{t-1}$.

Bandit formulation. We cast domain selection as a multi-armed bandit in which each domain D_m corresponds to an arm. At each step t the curriculum (i) samples an arm m_t from a distribution over per-arm value estimates $\{Q_m^{(t-1)}\}$; (ii) draws a single-domain minibatch $\mathcal{B}_t \subset D_{m_t}$ and applies one GRPO update; (iii) observes a scalar feedback signal $S_{m_t}^{(t)}$; and (iv) refreshes the pulled arm's estimate via an exponential moving average,

$$Q_{m_t}^{(t)} = (1 - \alpha)\, Q_{m_t}^{(t-1)} + \alpha\, S_{m_t}^{(t)}, \tag{4}$$

with bandit learning rate $\alpha \in (0, 1]$. By default the pulled arm is refreshed and the others retain their previous estimates; `TAC` extends step (iv) to also refresh unsampled arms whose transferability has been recomputed, while leaving the EMA form of Eq. (4) unchanged (§3.3). The EMA absorbs the non-stationarity that arises as π_θ evolves.

Arm selection. We sample arms from a Boltzmann distribution over UCB-augmented values:

$$\mu_m^{(t)} \;\propto\; \exp\left(\frac{1}{\tau} \left[Q_m^{(t-1)} + \frac{c}{\sqrt{n_m^{(t-1)} + 1}} \right] \right), \tag{5}$$

where $n_m^{(t-1)} = \sum_{s<t} \mathbb{I}[m_s = m]$ is the visit count, $c > 0$ scales the exploration bonus following the UCB form [3], and τ is a softmax temperature. We use a deliberately soft temperature $\tau = 0.85$, which keeps the UCB exploration bonus influential relative to the small Q-value gaps, so domains whose value dipped early are revisited rather than starved.

Design degree of freedom. Under this template, the curriculum is fully specified by the per-arm feedback signal $S_m^{(t)}$: the bandit machinery in Eqs. (4)–(5) is invariant to its choice, while $S_m^{(t)}$ determines what the curriculum optimizes for. Prior bandit curricula instantiate $S_m^{(t)}$ as a *learnability* term [8], prioritizing domains the policy is presently improving on but blind to whether updates on the chosen domain benefit the others. In Section 3 we keep the same template and introduce a feedback signal that pairs learnability with a gradient-based measure of *cross-domain transferability*.

3 Method

Selecting which domain to train on at each step of multi-domain RL hinges on two questions. First, how much *optimization signal* does the domain currently provide? Second, and central to our work, how well does the resulting gradient step *transfer* to the remaining domains? A curriculum tracking only the first over-commits to locally rich but globally narrow domains; one tracking only the second starves the policy of usable signal.

We propose Transfer-Aware Curriculum (`TAC`), a curriculum addressing both within the bandit framework of Section 2.3. `TAC` specifies $S_m^{(t)}$ as a weighted combination of *(1)* a **local learnability term** (§3.1) derived from on-policy GRPO advantages, and *(2)* a **global transferability term** (§3.2) estimating how well an update on D_m aligns with updates on the others. The full procedure is summarized in Figure 1b and Algorithm 1; details follow in Section 3.3.

3.1 Local Learnability Signal

For a minibatch $\mathcal{B}_t \subset D_{m_t}$ of size B, the mean absolute GRPO advantage

$$L_{m_t}^{(t)} \;=\; \frac{1}{BK}\sum_{b=1}^{B}\sum_{k=1}^{K}\left|A_b^{(k)}\right| \tag{6}$$

serves as an on-policy proxy for learnability [8, 55]: with binary rewards, it is large when the rollout group contains a mix of successes and failures, the regime of active improvement, and collapses to zero when all K rollouts share the same reward, either because D_m is saturated or because it is presently beyond the policy's capacity. Because its raw scale varies across domains and training stages, we feed a running z-score into the curriculum,

$$\hat{L}_{m_t}^{(t)} \;=\; \frac{L_{m_t}^{(t)} - \mu_L^{(t)}}{\sigma_L^{(t)} + \epsilon}, \tag{7}$$

where $\mu_L^{(t)}$ and $\sigma_L^{(t)}$ are EMAs of the observed L values' mean and standard deviation; during the first few observations, before these statistics are reliable, we feed the raw $L_{m_t}^{(t)}$ to the curriculum instead of the z-score (Appendix B).

3.2 Gradient-Based Transferability

A good curriculum should not only select domains on which the policy is currently learning, but also favor domains whose gradient updates are directionally aligned with those of the other training domains. We estimate this alignment directly from training gradients.

Projected-gradient representation. Full gradient vectors are high-dimensional and expensive to compare across domains. Following Panigrahi et al. [41], we sketch gradients into a shared low-dimensional space via TRAK-style random projections [43]. Let $P \in \mathbb{R}^{d\times r}$ be the fixed projection matrix with $r \ll d$, where d is the dimensionality of a designated parameter subset (the last N transformer layers). Given the GRPO gradient $g_t = \nabla_\theta \mathcal{L}_{\mathrm{GRPO}}(\theta^{(t)}; \mathcal{B}_t)$ restricted to that subset, the projected representation is the unit-normalized sketch

$$\mathbf{v}_t \;=\; \frac{P^\top g_t}{\left\|P^\top g_t\right\|_2} \;\in\; \mathbb{R}^r. \tag{8}$$

The ℓ_2 normalization makes the per-domain state below a pure average of update *directions*: without it, a single step with an unusually large gradient dominates the accumulated state for many steps, and the cosine comparisons then mostly reflect that one outlier. We deliberately apply no response-length rescaling to $\mathbf{v}_t$: response length varies systematically across domains, so magnitude corrections based on it inject a persistent domain-level bias into the accumulated direction rather than removing noise. The projection matrix P is generated once with a fixed seed and reused throughout training; in practice we project the gradient of the final PPO mini-batch of each step, and the smoothing below absorbs the additional variance (Appendix B).

Per-domain gradient state. For each domain D_m we maintain an exponential moving average of its projected gradient, updated only when D_m is selected:

$$\mathbf{h}_{m_t}^{(t)} \;=\; \gamma\, \mathbf{h}_{m_t}^{(t-1)} + (1-\gamma)\, \mathbf{v}_t, \tag{9}$$

with decay $\gamma \in [0, 1)$; the first observation for a domain initializes its state directly, and unselected domains retain their previous EMAs. Each $\mathbf{h}_m^{(t)}$ thus accumulates a smoothed representation of the gradient directions most recently induced by D_m.

Pairwise transferability. A domain scores highly if its gradient state is aligned with those of the other active training domains. Every K_c steps, we recompute the raw pairwise transfer of *every* domain with an initialized gradient state as the mean cosine similarity against all others,

$$\rho_m^{(t)} \;=\; \frac{1}{|\mathcal{A}_t| - 1} \sum_{\substack{j \in \mathcal{A}_t \\ j \neq m}} \frac{\langle \mathbf{h}_m^{(t)},\, \mathbf{h}_j^{(t)} \rangle}{\|\mathbf{h}_m^{(t)}\|_2\, \|\mathbf{h}_j^{(t)}\|_2 + \epsilon}, \tag{10}$$

where $\mathcal{A}_t$ is the set of domains with initialized gradient states. Note that $\rho_m^{(t)}$ changes at every comparison even for domains that were not sampled in between, because the *other* domains' gradient states have moved.

Temporal smoothing. To damp this step-level variance, each domain's raw cosine is first passed through a per-domain EMA with smoothing coefficient $\delta \in [0, 1)$, refreshed for all $m \in \mathcal{A}_t$ at comparison steps:

$$\tilde{\rho}_m^{(t)} \;=\; \begin{cases} \delta\, \tilde{\rho}_m^{(t-1)} + (1-\delta)\, \rho_m^{(t)}, & m \in \mathcal{A}_t \text{ at refresh steps } (t \bmod K_c = 0), \\ \tilde{\rho}_m^{(t-1)}, & \text{otherwise;} \end{cases} \tag{11}$$

between comparisons the smoothed estimate carries the signal forward.

Cross-domain normalization. What matters for domain selection is not a domain's absolute cosine, which is uniformly small in the projected subspace (Appendix C.3), but how it compares to the *other* domains right now. We therefore map the smoothed cosines to a bounded relative score via min–max normalization with a floored, EMA-smoothed scale:

$$s^{(t)} \;=\; \delta_s\, s^{(t-1)} + (1-\delta_s)\Big(\max_{j \in \mathcal{A}_t} \tilde{\rho}_j^{(t)} - \min_{j \in \mathcal{A}_t} \tilde{\rho}_j^{(t)}\Big), \qquad T_m^{(t)} \;=\; \mathrm{clip}\left(\frac{\tilde{\rho}_m^{(t)} - \min_{j \in \mathcal{A}_t} \tilde{\rho}_j^{(t)}}{\max\big(s^{(t)},\, s_{\min}\big)},\, 0,\, 1\right), \tag{12}$$

with scale-EMA decay δ_s and floor $s_{\min}$ ($s^{(0)}$ is initialized to the first observed range). This pins the worst-transferring domain near 0 and the best near 1, yielding a bounded, monotone ranking signal. The floored EMA scale guards against a specific failure mode: the cross-domain spread occasionally collapses when all domains' smoothed cosines briefly cluster, and dividing by the raw spread would then produce a one-step spike that, at bandit learning rate α, contaminates the Q-values for many subsequent steps.

3.3 Curriculum Signal and Algorithm

The `TAC` feedback signal combines the learnability and transferability terms:

$$S_{m_t}^{(t)} \;=\; \beta\, \hat{L}_{m_t}^{(t)} \;+\; (1-\beta)\, T_{m_t}^{(t)}, \tag{13}$$

where $\beta \in [0, 1]$ interpolates between pure-learnability ($\beta = 1$) and pure-transferability ($\beta = 0$) selection. The two terms play complementary roles on deliberately different scales: $T_m^{(t)}$ is a bounded relative ranking in $[0, 1]$ that orders domains by current cross-domain alignment, while $\hat{L}_{m_t}^{(t)}$ is an unbounded z-score that injects the locally available learning signal; β balances the two empirically (§4.1, Table 6).

Algorithm 1: Transfer-Aware Curriculum for RL (TAC)

Require: Domains $\{D_m\}_{m=1}^{M}$; policy π_θ; projection matrix P; hyperparameters $\alpha, \beta, \gamma, \delta, \delta_s, \kappa, c, \tau, r, N, K_c, W$ (see Table 6 for values)

Initialize $\mathbf{h}_m^{(0)} \leftarrow \mathbf{0}$ and $T_m^{(0)} \leftarrow 0$ for all m; initialize $Q_m^{(0)}$ via Eq. (14)
Run W-round round-robin warmup over all M domains (bandit and normalizers update live)
for step $t = 1, 2, \ldots$ **do**
  Sample m_t via Eq. (5); draw $\mathcal{B}_t \subset D_{m_t}$
  Compute K rollouts and $\hat{L}_{m_t}^{(t)}$ (Eqs. 6–7)
  Compute gradient g_t, form unit sketch $\mathbf{v}_t$ (Eq. 8), update $\mathbf{h}_{m_t}^{(t)}$ (Eq. 9)
  if $t \bmod K_c = 0$ **then** update $\{\rho_m^{(t)}\}_{m \in A_t}$, $\{\tilde{\rho}_m^{(t)}\}_{m \in A_t}$ (Eqs. 10–11) and $\{T_m^{(t)}\}_{m \in A_t}$ (Eq. 12)
  Sampled arm: compute $S_{m_t}^{(t)}$ (Eq. 13); update $Q_{m_t}^{(t)}$ (Eq. 4); cache $\hat{L}_{m_t}^{(t)}$
  if $t \bmod K_c = 0$ **then** update $Q_m^{(t)}$ (Eq. 4) for each unsampled m with cached $\hat{L}_m$, using $S_m^{(t)} = \beta \hat{L}_m + (1-\beta)\, T_m^{(t)}$
  Apply one GRPO step on $\mathcal{B}_t$
end for

Two-phase value update. A standard bandit refreshes only the arm it pulls. But TAC recomputes transferability for *every* domain at each comparison step (Eq. 12), so it also has fresh feedback for the arms it did not pull, and uses it. Each step, the sampled arm m_t is updated as usual via Eq. (4) with a fresh learnability term and the current $T_{m_t}^{(t)}$. Then, at every comparison step, each *unsampled* arm is updated by the same rule, combining its fresh $T_m^{(t)}$ with its most recent learnability, cached from the last time it was pulled. Arms never yet pulled have no cached learnability and are left untouched. This lets a domain's transferability reallocate sampling mass even while the bandit is not actively revisiting it. We detail the caching and normalizer bookkeeping in Appendix B.

Initialization and warmup. With $p_m = |D_m| / \sum_j |D_j|$, Q-values are initialized to centered log-proportions,

$$Q_m^{(0)} \;=\; \kappa \left(\log p_m - \tfrac{1}{M} \textstyle\sum_j \log p_j \right), \tag{14}$$

with scale κ, so that initial sampling follows $\mu_m^{(0)} \propto |D_m|^{\kappa/\tau}$, proportional to domain size on a softened (log-scaled) footing. In the balanced training setting, where the per-domain cap binds for every domain, this reduces exactly to uniform initialization ($Q_m^{(0)} = 0$). At the start of each epoch, the sampler then runs W rounds of round-robin warmup over all M domains (order shuffled per round) before applying Eq. (5). Warmup only forces *which* arm is drawn: the bandit, the normalizers, and the visit counts all update live during warmup. This guarantees that every domain contributes multiple gradient-state updates before transferability comparisons drive sampling, and that the learnability normalizer is past its own warmup—with every arm's cached $\hat{L}_m$ refreshed to a properly normalized value—by the time bandit control begins. The UCB bonus in Eq. (5) then carries the ongoing exploration burden after warmup, occasionally reviving arms whose Q-values dipped early so that their cached learnability is refreshed.

Data-exhaustion policy. Each domain maintains an independent index pool per epoch. When the bandit selects a domain whose pool is exhausted, we reshuffle that domain's data and resume sampling from it, rather than re-drawing m_t from the remaining domains. This lets high-Q domains be over-sampled within an epoch while guaranteeing no data is permanently withheld across epochs.

The full procedure, using the bandit template of Eqs. (4) and (5) with the feedback signal $S_{m_t}^{(t)}$ from Eq. (13), is summarized in Algorithm 1. Additional implementation details (warmup schedule, choice of N, r, K_c) are in Section 4.1 and Appendix B.

Interpretation. The gradient-cosine signal admits a local first-order justification under the GRPO surrogate loss minimized at each step, in the spirit of gradient-based influence analyses [24, 44, 43, 56]. A Taylor expansion of $\mathcal{L}_{\mathrm{GRPO}}$ on domain D_j around a step $-\eta g_t$ taken on D_{m_t} gives

$$\mathcal{L}(\theta - \eta g_t;\, D_j) \;\approx\; \mathcal{L}(\theta;\, D_j) \;-\; \eta \, \langle g_t,\, g_j \rangle, \tag{15}$$

so $\langle g_t, g_j \rangle$ predicts, to first order, the loss change on D_j from a step on D_{m_t} [44]. The cosine in Eq. (10) is the direction-only version of this predictor, mirroring the gradient-alignment view used in multi-task optimization [61, 33]: it preserves the sign of the predicted improvement while normalizing away gradient magnitudes that vary substantially across domains and training stages—a view reinforced by the unit normalization of $\mathbf{v}_t$ in Eq. (8). We compute it in a low-dimensional sketch where random projections preserve inner products in expectation [23, 43], and substitute the EMAs $\mathbf{h}_m^{(t)}, \mathbf{h}_j^{(t)}$ for instantaneous gradients to reduce step-level variance. Averaging over $j \neq m_t$ gives the aggregate cross-domain improvement summarized by $\rho_{m_t}^{(t)}$. Two properties are worth noting: the signal is *adaptive*, since the gradient states $\{\mathbf{h}_m\}$ evolve with π_θ so $\{T_m^{(t)}\}$ tracks the *current* gradient geometry rather than a static notion of similarity; and *relative*, since the normalization in Eq. (12) rewards a domain for transferring well compared with the other active domains, not for having large gradients. Together with the learnability term, Eq. (13) thus implements a compact answer to the question *"does a step on D_m both carry local learning signal and point in a direction aligned with the rest of the training set?"*

4 Experiments

4.1 Experimental Setup

Datasets. We train and evaluate on the GURU multi-domain reasoning suite [11], spanning six reasoning domains: *math*, *codegen*, *logic*, *simulation*, *table*, and *stem*. We follow the original GURU sources except *stem*, where we replace WebInstruct [36] with OpenScienceReasoning-2[1], a science-reasoning dataset with higher-quality traces. To isolate curriculum design from raw data imbalance, we cap each domain at 1,000 queries; without this cap, math and stem dominate training by two orders of magnitude. We additionally construct an *imbalanced* data budget (1,500 for math and stem, 500 for simulation and table, 1,000 elsewhere) to stress-test the curriculum under realistic source-size skew (§4.2). For evaluation, we use held-out benchmarks spanning all six domains: MATH-500 [19], AIME [2] (math); HumanEval [7], MBPP [4] (codegen); zebra puzzles [32], ARC-AGI [12, 13] (logic); CodeI/O [27], CruxEval [16] (simulation); HiTab [10], MultiHierTT [63], FinQA [9] (table); GPQA-Diamond [46], SuperGPQA [14] (stem). Training data details are in Appendix B.1; evaluation pipeline in Appendix B.3; full per-domain prompts in Appendix A.

Models. We evaluate on `Qwen3-1.7B-Base` [59] and `Llama3.2-3B-Instruct` [15]. We use the instruction-tuned Llama variant because its base model produces no correct rollouts on several training domains, collapsing GRPO's advantage signal to zero. Scaling results on `Qwen3-{0.6, 4}B-Base`, confirming `TAC`'s gains hold across model sizes, are in Appendix C.4.

Baselines. We compare `TAC` against four baselines under identical data and compute budgets: *(i)* **Base**, the model without RL training; *(ii)* **Random**, which draws each single-domain batch from a domain sampled with fixed probability proportional to its pool size (uniform across domains when pools are balanced); this is the standard pooled multi-domain baseline, with no online adaptation; *(iii)* **Math-to-Others** (M2O) [40], a manually designed two-stage curriculum that first trains on math and then on the remaining domains; and *(iv)* **SEC** [8], an advantage-based learnability curriculum corresponding to `TAC` with $\beta = 1$. Math-to-Others tests hand-designed vs. adaptive scheduling; SEC isolates the contribution of transferability.

Implementation Details. We train all methods with GRPO using $K = 4$ rollouts per query, a per-step batch size of 64 queries from a single domain, and 2 epochs total. The bandit uses $\alpha = 0.3$, $c = 0.2$, softmax temperature $\tau = 0.85$, and five rounds of round-robin warmup per epoch ($W = 5$). For the transferability signal, we sketch gradients with a Rademacher JL projection [43] of dimension $r = 4096$ on the last $N = 4$ transformer layers, taken from the final PPO mini-batch of each step; we update per-domain gradient EMAs with $\gamma = 0.8$, recompute pairwise cosines every $K_c = 2$ steps, temporally smooth them with $\delta = 0.8$, and map them to per-arm scores via the min–max cross-domain normalization of Eq. (12) (scale-EMA decay $\delta_s = 0.9$, floor $s_{\min} = 0.01$). The learnability and transferability terms are mixed with $\beta = 0.2$. Full tables, compute details, and complexity analysis are in Appendix B.2 and Appendix C.2.

Table 1: Pass@1 accuracy (%) across 14 held-out benchmarks. *Base* is the off-the-shelf model; *Random*, *M2O*, and *SEC* are curriculum baselines. Each entry is mean±std over three training seeds, with most benchmarks additionally averaged over 4 evaluation runs.

		Qwen3-1.7B					Llama3.2-3B				
Domain	**Benchmark**	Base	Random	M2O	SEC	**TAC**	Base	Random	M2O	SEC	**TAC**
Codegen	HumanEval	$27.7_{\pm 0.3}$	$65.2_{\pm 1.9}$	$65.0_{\pm 1.8}$	$64.8_{\pm 0.3}$	$\mathbf{66.7}_{\pm 1.3}$	$57.0_{\pm 1.2}$	$57.0_{\pm 3.7}$	$54.6_{\pm 2.8}$	$54.6_{\pm 2.8}$	$\mathbf{58.4}_{\pm 2.0}$
	MBPP	$36.9_{\pm 0.1}$	$52.3_{\pm 1.3}$	$52.0_{\pm 0.8}$	$52.0_{\pm 1.0}$	$\mathbf{52.7}_{\pm 2.4}$	$52.7_{\pm 0.3}$	$50.5_{\pm 0.7}$	$51.5_{\pm 0.6}$	$51.1_{\pm 1.3}$	$\mathbf{52.8}_{\pm 0.6}$
Logic	ARC-AGI	$0.1_{\pm 0.0}$	$1.0_{\pm 0.4}$	$0.6_{\pm 0.1}$	$0.5_{\pm 0.2}$	$\mathbf{1.4}_{\pm 0.8}$	$1.0_{\pm 0.0}$	$0.8_{\pm 0.7}$	$0.4_{\pm 0.3}$	$0.1_{\pm 0.1}$	$\mathbf{1.2}_{\pm 0.1}$
	Zebra	$1.1_{\pm 0.1}$	$24.9_{\pm 6.5}$	$28.5_{\pm 1.4}$	$26.7_{\pm 3.1}$	$\mathbf{34.7}_{\pm 0.9}$	$1.4_{\pm 0.3}$	$30.5_{\pm 3.7}$	$30.9_{\pm 2.0}$	$\mathbf{36.5}_{\pm 1.1}$	$35.8_{\pm 0.6}$
Math	MATH	$52.8_{\pm 0.8}$	$59.6_{\pm 1.1}$	$59.8_{\pm 0.8}$	$59.1_{\pm 0.5}$	$\mathbf{60.1}_{\pm 0.6}$	$34.4_{\pm 0.6}$	$41.9_{\pm 3.3}$	$46.1_{\pm 2.0}$	$43.1_{\pm 2.8}$	$\mathbf{46.9}_{\pm 0.8}$
	AIME	$4.3_{\pm 0.1}$	$5.3_{\pm 1.0}$	$5.2_{\pm 0.4}$	$\mathbf{5.4}_{\pm 0.6}$	$5.1_{\pm 0.6}$	$2.2_{\pm 0.1}$	$3.4_{\pm 0.8}$	$4.9_{\pm 1.0}$	$3.6_{\pm 0.3}$	$\mathbf{5.2}_{\pm 0.9}$
Simulation	CodeI/O	$3.2_{\pm 0.3}$	$\mathbf{5.2}_{\pm 0.6}$	$4.2_{\pm 0.3}$	$4.4_{\pm 0.6}$	$4.8_{\pm 0.6}$	$2.3_{\pm 0.3}$	$\mathbf{6.7}_{\pm 3.3}$	$2.0_{\pm 0.7}$	$2.7_{\pm 1.1}$	$3.3_{\pm 0.1}$
	CruxEval-I	$15.1_{\pm 0.2}$	$37.8_{\pm 2.8}$	$42.3_{\pm 1.8}$	$\mathbf{42.9}_{\pm 0.4}$	$42.2_{\pm 2.1}$	$31.4_{\pm 0.3}$	$39.7_{\pm 3.4}$	$36.1_{\pm 0.7}$	$35.8_{\pm 3.9}$	$\mathbf{40.6}_{\pm 2.5}$
	CruxEval-O	$15.0_{\pm 0.2}$	$34.8_{\pm 5.3}$	$\mathbf{41.5}_{\pm 0.9}$	$35.8_{\pm 4.0}$	$39.5_{\pm 3.6}$	$27.6_{\pm 0.5}$	$36.2_{\pm 1.7}$	$\mathbf{37.4}_{\pm 4.6}$	$34.0_{\pm 4.6}$	$35.5_{\pm 4.6}$
STEM	GPQA	$24.2_{\pm 0.3}$	$30.0_{\pm 2.8}$	$25.8_{\pm 1.1}$	$30.7_{\pm 1.2}$	$\mathbf{31.9}_{\pm 0.5}$	$23.6_{\pm 0.5}$	$28.0_{\pm 0.8}$	$27.7_{\pm 0.6}$	$26.2_{\pm 0.5}$	$\mathbf{28.7}_{\pm 3.2}$
	SuperGPQA	$18.5_{\pm 0.2}$	$21.9_{\pm 0.9}$	$21.7_{\pm 0.5}$	$21.9_{\pm 0.9}$	$\mathbf{22.1}_{\pm 0.5}$	$21.0_{\pm 0.4}$	$18.8_{\pm 1.3}$	$\mathbf{21.5}_{\pm 0.8}$	$18.0_{\pm 1.7}$	$21.3_{\pm 0.5}$
Table	MultiHierTT	$14.7_{\pm 0.1}$	$25.5_{\pm 1.2}$	$24.8_{\pm 0.4}$	$26.5_{\pm 0.8}$	$\mathbf{28.7}_{\pm 0.9}$	$16.6_{\pm 0.4}$	$21.3_{\pm 2.1}$	$23.0_{\pm 0.3}$	$17.8_{\pm 4.8}$	$\mathbf{24.8}_{\pm 2.0}$
	FinQA	$9.7_{\pm 0.3}$	$24.3_{\pm 2.5}$	$24.7_{\pm 0.8}$	$23.2_{\pm 1.3}$	$\mathbf{25.6}_{\pm 1.0}$	$15.6_{\pm 0.3}$	$14.2_{\pm 1.6}$	$19.2_{\pm 3.3}$	$15.4_{\pm 8.2}$	$\mathbf{21.7}_{\pm 1.0}$
	HiTab	$13.7_{\pm 0.3}$	$54.5_{\pm 0.6}$	$51.5_{\pm 2.1}$	$53.4_{\pm 2.0}$	$\mathbf{56.5}_{\pm 1.6}$	$16.6_{\pm 0.3}$	$60.5_{\pm 1.6}$	$60.0_{\pm 1.3}$	$56.5_{\pm 1.9}$	$\mathbf{61.3}_{\pm 0.6}$
All (macro avg.)		$17.8_{\pm 0.1}$	$31.8_{\pm 0.5}$	$32.0_{\pm 0.2}$	$32.1_{\pm 0.6}$	$\mathbf{33.9}_{\pm 0.2}$	$22.2_{\pm 0.2}$	$29.2_{\pm 0.2}$	$29.7_{\pm 0.3}$	$28.5_{\pm 1.5}$	$\mathbf{31.3}_{\pm 0.6}$

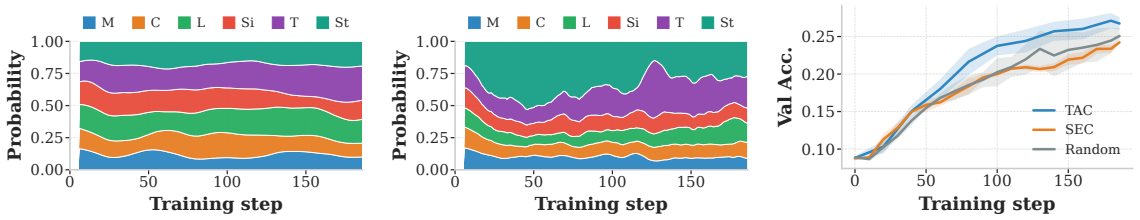
4.2 Results

Main Results. Table 1 compares TAC with the baselines across both backbones. TAC achieves the best macro-averaged accuracy on both models, improving over the strongest baseline by $+1.8$ pp ($+5.6\%$ relative) on `Qwen3-1.7B` (over SEC) and $+1.6$ pp ($+5.4\%$ relative) on `Llama3.2-3B` (over M2O), and ranking first on 10/14 benchmarks on both.

The two adaptive baselines behave inconsistently. M2O improves over Random on both backbones ($+0.2$ on Qwen, $+0.5$ on Llama), confirming that prior domain knowledge yields a useful schedule, but its static two-stage structure cannot react to mid-training shifts in cross-domain transfer. SEC is far less stable: it is the strongest baseline on `Qwen3-1.7B` ($+0.3$ over Random) yet *underperforms* Random on `Llama3.2-3B` (-0.7), and carries the highest macro variance of any method (± 1.5 on Llama). As the Curriculum Dynamics analysis below shows, a learnability-only bandit anchors onto whichever domain produces the strongest early advantage signal, regardless of whether its updates benefit the rest of the training set, a gamble that pays off on one backbone but not the other. TAC's transferability term avoids this trap.

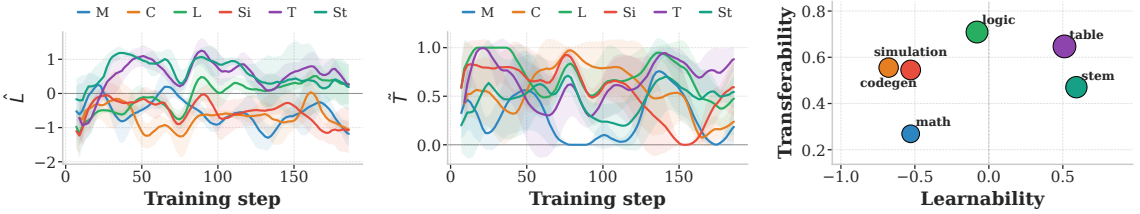
The gains are broad rather than concentrated. TAC's largest per-benchmark improvements span logic, math, and table reasoning: Zebra rising $+9.8$ over Random and $+8.0$ over SEC on Qwen, MATH rising $+5.0$ over Random on Llama, and FinQA $+7.5$ over Random with MultiHier $+3.5$ on Llama ($+3.2$ on Qwen). They also extend to stem ($+1.9$ on GPQA over Random for Qwen), indicating the transferability signal is not biased toward any particular target. The curriculum analysis flags *math* and *codegen* as the *least* transferable domains: their training gradients align weakly, even negatively, with every other domain (Appendix C.3), an effect we attribute to base models such as `Qwen3` being heavily pretrained on math and code. Yet TAC, which *down-weights* both relative to uniform sampling, still improves their benchmarks. We read this as capability-level spillover, where investing the schedule in the broadly-transferable domains reinforces general verifiable-task solving and exposes non-math reasoning styles that carry over to math and code at evaluation, even though the local gradient-cosine signal does not directly credit it. The only benchmarks where TAC is not first on *either* backbone, CodeI/O and CruxEval-O, both fall in the simulation domain, where local learnability and transferability are both relatively flat (Figure 3) and the curriculum has limited leverage. The

[1] https://huggingface.co/datasets/nvidia/OpenScienceReasoning-2



(a) Sampling under TAC. **(b)** Sampling under SEC ($\beta = 1$). **(c)** Macro validation accuracy.

Figure 2: Curriculum dynamics. (a)–(b) Per-domain sampling probability $\mu_m^{(t)}$ under TAC and SEC; colors index the six training domains (*math, codegen, logic, simulation, table, stem*). **(c)** Macro-averaged validation accuracy across all evaluation benchmarks.



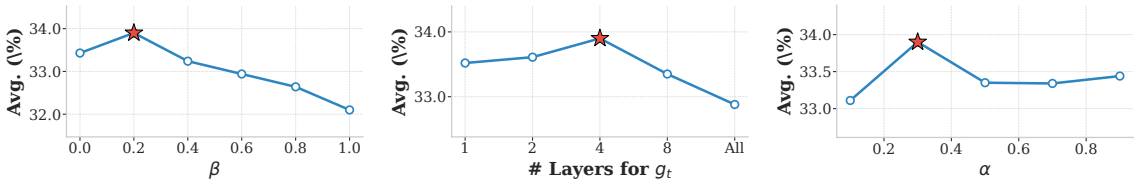
(a) Learnability $\hat{L}_m^{(t)}$. **(b)** Transferability $T_m^{(t)}$. **(c)** Training-step average of $(\hat{L}_m, T_m)$.

Figure 3: Learnability and transferability favor different domains. (a) Per-domain learnability $\hat{L}_m^{(t)}$ across training under TAC, recovered from the bandit's composite score by inverting the β-mixture $S_m = \beta\,\hat{L}_m + (1-\beta)\,T_m$ for $\hat{L}_m$ (the normalized mean $|A|$ of Eq. (6)), so that **(a)** and **(b)** are measured under the *same* TAC curriculum. **(b)** Per-domain transferability $T_m^{(t)}$ under TAC, computed via Eqs. (10)–(12). **(c)** $(\hat{L}_m, T_m)$ per domain, averaged over *all* training steps; marker area is proportional to each domain's mean sampling probability under TAC. Off-axis placement shows the two signals rank domains differently. All curves and markers are averaged over 3 seeds.

gains also transfer across families: although `Llama3.2-3B` starts substantially weaker than `Qwen3-1.7B` (e.g., 34.4 vs. 52.8 on MATH), TAC improves over Random by an identical $+2.1$ macro points on both backbones, which we attribute to the transferability signal being computed from each model's own training gradients. Examples of the resulting qualitative differences in model rollouts are provided in Appendix D.

Curriculum Dynamics: TAC Adapts; SEC Anchors. Because the curriculum is online, *when* each method samples each domain matters as much as what. Both TAC and SEC begin near proportional initialization and behave similarly for $\sim$30 steps. Around step 30–50, *stem* learnability spikes (Figure 3a, peaking near step $\sim$35 before it is trained down) and SEC's bandit collapses onto it: by step $\sim$60, stem occupies about half of SEC's sampling mass (Figure 2b) and stays the dominant domain thereafter. TAC up-weights stem only transiently (Figure 2a), peaking near step $\sim$60, then releases it as *table* overtakes every other domain in transferability (Figure 3b); TAC redirects the freed budget toward *table*, which reaches its largest sampling share by the end of training. The decisive point is that the most *learnable* domain (stem) and the most *transferable* domain (table) are not the same: SEC, ranking domains by learnability alone, pours most of its mass into stem, whereas TAC, rewarding transferability on top of learnability, shifts the budget onto table (strong on *both* signals) and logic. The validation curves (Figure 2c) trace the same timeline: the three methods are indistinguishable until step $\sim$40, after which TAC pulls ahead while SEC tracks Random closely, indicating that without a transferability-sensitive signal the bandit machinery alone does not improve over uniform sampling.

Learnability and Transferability Favor Different Domains. Because both signals are read from TAC's own run, learnability recovered as the normalized mean $|A|$ ($\hat{L}$) and transferability as T, Figures 3a and 3b track $\hat{L}$ and T under a single curriculum, and the domains they favor separate *progressively*. **Early** ($t \lesssim 50$)**:** learnability spikes for stem (and table) while T is uniformly high and not yet discriminative, so SEC and TAC make similar decisions. **Mid** ($t \approx 50$–100)**:** stem's $\hat{L}$ stays high but begins to fall as it is trained down, table's $\hat{L}$ holds, and T differentiates sharply: table climbs to the highest transferability while *codegen* and *math* drift to the bottom, so TAC's sampling peels away from SEC's toward table. **Late** ($t \gtrsim 100$)**:** stem's $\hat{L}$ keeps



(a) Mixing coefficient β. **(b)** Number of layers N. **(c)** Bandit learning rate α.

Figure 4: Hyperparameter ablations. Macro-averaged accuracy (unweighted mean over the six domain scores) while varying **(a)** the mixing coefficient β ($\beta=0$: pure transferability; $\beta=1$: pure learnability); **(b)** the number of trailing transformer layers N used for the projected gradient (Eq. 8); and **(c)** the bandit EMA learning rate α (Eq. 4). Shaded regions denote standard deviation over three seeds.

falling while *logic*'s climbs, so the two orderings stay misaligned rather than converging. Averaged over the whole run (Figure 3c, marker area $\propto$ sampling share), this leaves stem at high $\hat{L}$ but only moderate T, *logic* at low $\hat{L}$ yet high T, table high on *both* axes and sampled most, and *math* low on both and sampled least. A learnability-only curriculum locks onto the high-$\hat{L}$ stem and under-weights the equally- or more-transferable table and logic; `TAC` uses T to correct this.

This online ranking is consistent with the offline transfer matrix in Figure 1a, where *table* as a single source yields among the largest off-diagonal accuracy gains and *math* the smallest, though the two need not match exactly: the matrix measures end-to-end accuracy transfer from training each domain in isolation, whereas `TAC`'s signal is an online, first-order estimate of gradient alignment that evolves with the policy. The per-pair gradient cosines in Appendix C.3 confirm the same structure: *math* and *codegen* align weakly, often negatively, with every other domain and most weakly with *each other*, while *table* behaves as a hub that aligns positively with simulation, stem, and logic. We attribute the math/codegen isolation to a pretraining effect: base models, and `Qwen3` in particular, are saturated with math and code, so their RL updates point in idiosyncratic directions that are near-orthogonal to the shared reasoning gradients the other domains move along, sampling them more sharpens an already-strong skill rather than benefiting the rest.

Ablation: Components of `TAC`. We ablate `TAC`'s three main hyperparameters: the mixing coefficient β, the bandit learning rate α, and the number of trailing layers N used for the projected gradient. Performance is stable within ± 0.2 points around our default settings (Figure 4). β peaks at $\beta \approx 0.2$ and degrades sharply at $\beta=1$ (Figure 4a), confirming the contribution of transferability; $\beta=0$ is competitive but slightly weaker, indicating learnability still adds signal. α peaks at 0.3 (Figure 4c): smaller values update too slowly given how fast the gradient geometry changes during RL, larger values amplify step-level noise. N has the smallest effect (Figure 4b): even a few trailing layers already match the full-model gradient, indicating the last layers carry most of the cross-domain alignment signal, consistent with gradient directions in trailing layers being highly correlated. Additional ablations on the projection family, projection dimension r, and the data-exhaustion strategy are in Appendix C.1.

Additional Study: Imbalanced Data Budget. The main results equalize the data budget across domains to isolate the curriculum's contribution from raw data skew. In practice, training pools are rarely balanced. We re-run training on an imbalanced data budget (1,500 for math and stem, 500 for simulation and table, 1,000 elsewhere) and evaluate on the same 14 benchmarks; Table 2 reports results for `Qwen3-1.7B`.

`TAC` again leads on macro-averaged accuracy (32.7, $+1.9$ over Random, $+1.5$ over SEC), and ranks first on 11 of 14 benchmarks. The gap over SEC is the more informative one: the centered log-proportional Q-initialization gives math and stem roughly twice (about $1.9\times$) the initial mass of simulation and table, and a learnability-only signal provides no corrective force when those large-but-narrow domains saturate. `TAC`'s transferability term supplies that correction, releasing weight from over-represented domains once their updates stop benefiting the rest, visible in the largest gains accruing to under-represented domains (CodeI/O $+2.1$ over Random, FinQA $+4.0$ over SEC). `TAC`'s gains are thus robust to data-budget skew, not an artifact of the balanced setting.

Table 2: Imbalanced data budget (`Qwen3-1.7B`). Math and STEM use 1500 training samples; Simulation and Table use 500; remaining domains use 1000. Evaluation benchmarks are unchanged. Mean over three seeds; best per column in bold.

Method	Codegen		Logic		Math		Simulation			STEM		Table			All
	HE	MBPP	ARC-AGI	Zebra	MATH	AIME	CodeI/O	CruxEval-I	CruxEval-O	GPQA	SuperGPQA	MH	FinQA	HiTab	*macro avg.*
Random	66.6	**53.8**	0.5	12.0	60.8	4.7	4.4	41.1	38.8	30.6	20.3	25.9	23.0	47.6	30.8
SEC	64.6	51.7	**1.6**	21.2	58.9	4.6	4.2	40.8	36.8	31.9	20.6	25.9	19.7	**50.8**	31.2
`TAC`	**67.1**	52.7	1.2	**23.6**	**61.1**	**5.9**	**6.5**	**42.3**	**40.0**	**33.0**	**22.2**	**26.5**	**23.7**	49.2	**32.7**

5 Related Work

Multi-Domain Reasoning for LLMs. Multi-domain training has roots in classical multi-task learning, including shared-representation approaches [5, 47], gradient-conflict mitigation [61, 33], and data-mixture optimization for language model pretraining [58]. The line we build on extends these ideas to RL training of LLM reasoners across multiple reasoning domains. General Reasoner [36] and GURU [11] introduce broad multi-domain reasoning benchmarks, including non-verifiable domains that rely on LLM-as-a-judge rewards [64], and show that math-only RL is insufficient for general-purpose reasoning. Huan et al. [20] and Li et al. [30] further examine when and how reasoning skills transfer across domains. Building on these benchmarks, methodological work has begun to address multi-domain training: Liang et al. [31] encourage gradient alignment across domains during RL, while Ramesh et al. [45] reweight per-task losses for balanced multi-task GRPO. Both modify the optimizer or loss; we instead target the sampling distribution, selecting domains by their joint learnability and cross-domain transferability.

Curriculum Learning for RL. Curriculum learning has long been studied in RL [38, 53, 54, 52], typically by estimating task difficulty or learning progress and then adaptively selecting tasks, goals, or environments to improve sample efficiency. Recent work extends this to RL training of LLM reasoners. Omni-Thinker [26] updates its sampling distribution using an accuracy-based backward-transfer signal computed on held-out evaluation probes; Pang et al. [40] propose a math-to-others curriculum that leverages math reasoning as a foundation for other tasks; and easy-to-hard curricula [42] have likewise been adopted. DUMP [55] and SEC [8] use advantage-based learnability to construct curricula automatically, with a similar idea applied to variance-based rewards [22]. `TAC` shares the bandit-curriculum framework and advantage-based learnability term of these methods; its contribution is a complementary cross-domain transferability signal, estimated directly from training gradients without additional rollouts or held-out probes, that selects domains whose updates are directionally aligned with the rest of the training set.

6 Conclusion

We introduce Transfer-Aware Curriculum (`TAC`), a bandit-style curriculum for multi-domain RL that combines local learnability with a gradient-geometry estimate of cross-domain transferability, computed entirely from training gradients and evolving with the policy as training proceeds. Across a six-domain reasoning suite, `TAC` consistently outperforms proportional sampling, a hand-designed math-to-others curriculum, and a learnability-only bandit on both `Qwen3-1.7B` and `Llama3.2-3B-Instruct`, in both balanced and imbalanced training mixtures. We hope these results motivate further work on cross-domain transferability as a first-class signal for curriculum design in multi-domain reasoning.

Limitations. `TAC` targets curriculum design, the sampling distribution over training domains, rather than the optimizer, the reward model, or the policy architecture. This scoping is deliberate, but it also means `TAC`'s gains are complementary to, not substitutes for, advances along those other axes, and the strongest multi-domain reasoning systems will likely combine adaptive curricula with progress on optimization and reward design. Our experiments are likewise restricted to the RL-with-verifiable-rewards (RLVR) regime, where advantage-based learnability and gradient-based transferability admit clean definitions; extending these signals to domains with non-verifiable or model-judged rewards [64] is left to future work.

Broader Impact. RL post-training of LLMs has so far concentrated on a narrow set of domains, most prominently mathematics and code generation, where verifiable rewards are easy to construct. `TAC` makes it

cheaper and more reliable to train across a broader portfolio of reasoning domains simultaneously, which we view as a step toward reasoning capability that generalizes beyond the math-and-code corner of the design space, into domains such as scientific reasoning, table understanding, and structured logic. By improving sample efficiency rather than introducing new data sources or deployment capabilities, TAC also lowers the compute barrier to training competent reasoning models, broadening access for academic and resource-constrained developers. We do not anticipate domain-specific harms beyond those already associated with RL fine-tuning of large language models.

Acknowledgement

We especially thank Roger Grosse, Minsu Kim, and Kimin Lee for providing important suggestions on this project. We also thank Ryan Faulkner for discussions on the RL setup. Finally, we sincerely thank Jimin Lee for her help with the paper and figures.

This work was supported in part by the German Federal Ministry of Education and Research (BMBF): Tübingen AI Center, FKZ: 01IS18039B; by the Machine Learning Cluster of Excellence, EXC number 2064/1 – Project number 390727645; by the NSERC Discovery Grant RGPIN-2025-06491; and by the University of Toronto's Acceleration Consortium, which receives funding from the Canada First Research Excellence Fund (CFREF). Resources used in preparing this research project were provided, in part, by the Digital Research Alliance of Canada; the Province of Ontario; the Government of Canada through CIFAR; and companies sponsoring the Vector Institute.

References

[1] Dimitris Achlioptas. Database-friendly random projections. In *Proceedings of the Twentieth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems*, 2001.

[2] Art of Problem Solving. AIME Problems and Solutions, 2025. URL https://artofproblemsolving.com/wiki/index.php/AIME_Problems_and_Solutions. Accessed: 2025-05-15.

[3] Peter Auer. Using confidence bounds for exploitation-exploration trade-offs. *Journal of Machine Learning Research*, 3(Nov):397–422, 2002.

[4] Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

[5] Rich Caruana. Multitask learning. *Machine learning*, 28(1):41–75, 1997.

[6] Moses Charikar, Kevin Chen, and Martin Farach-Colton. Finding frequent items in data streams. In *EATCS International Colloquium on Automata, Languages and Programming*, pages 693–703. Springer, 2002.

[7] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[8] Xiaoyin Chen, Jiarui Lu, Minsu Kim, Dinghuai Zhang, Jian Tang, Alexandre Piché, Nicolas Gontier, Yoshua Bengio, and Ehsan Kamalloo. Self-evolving curriculum for llm reasoning. *arXiv preprint arXiv:2505.14970*, 2025.

[9] Zhiyu Chen, Wenhu Chen, Charese Smiley, Sameena Shah, Iana Borova, Dylan Langdon, Reema Moussa, Matt Beane, Ting-Hao Huang, Bryan R Routledge, et al. Finqa: A dataset of numerical reasoning over financial data. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, 2021.

[10] Zhoujun Cheng, Haoyu Dong, Zhiruo Wang, Ran Jia, Jiaqi Guo, Yan Gao, Shi Han, Jian-Guang Lou, and Dongmei Zhang. Hitab: A hierarchical table dataset for question answering and natural language generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

[11] Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Nilabjo Dey, Yonghao Zhuang, Yuheng Zha, et al. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[12] Francois Chollet, Mike Knoop, Gregory Kamradt, and Bryan Landers. Arc prize 2024: Technical report. *arXiv preprint arXiv:2412.04604*, 2024.

[13] Francois Chollet, Mike Knoop, Gregory Kamradt, Bryan Landers, and Henry Pinkard. Arc-agi-2: A new challenge for frontier ai reasoning systems. *arXiv preprint arXiv:2505.11831*, 2025.

[14] Xeron Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, Zhenlin Wei, Chujie Zheng, Kaixin Deng, Shuyue Guo, Shian Jia, Sichao Jiang, Yiyan Liao, Rui Li, Qinrui Li, Sirun Li, Yizhi LI, Yunwen Li, dehua ma, Yuansheng Ni, Haoran Que, Qiyao Wang, Zhoufutu Wen, Siwei Wu, Tianshun Xing, Zhenzhu Yang, Zekun Moore Wang, Junting Zhou, yuelin bai, Xingyuan Bu, chenglin cai, Liang Chen, Yifan Chen, Cheng Chengtuo, Tianhao Cheng, Keyi Ding, Siming Huang, HUANG YUN, Yaoru Li, Yizhe Li, Zhaoqun Li, Tianhao Liang, Chengdong Lin, Hongquan Lin, Yinghao Ma, Z.Y. Peng, Zifan Peng, Qige Qi, Shi Qiu, Xingwei Qu, Shanghaoran Quan, Yizhou Tan, Zili Wang, Hao Wang, Yiya Wang, Yubo Wang, Jiajun Xu, Kexin Yang, Ruibin Yuan, Yuanhao Yue, Tianyang Zhan, Chun Zhang, Jinyang Zhang, Xiyue Zhang, Owen Xingjian

Zhang, Yue Zhang, Yongchi Zhao, Xiangyu Zheng, ChenghuaZhong, Yang Gao, Zhoujun Li, Dayiheng Liu, Qian Liu, Tianyu Liu, Shiwen Ni, Junran Peng, Yujia Qin, Wenbo Su, Guoyin Wang, Shi Wang, Jian Yang, Min Yang, Meng Cao, Xiang Yue, Zhaoxiang Zhang, Wangchunshu Zhou, Jiaheng Liu, Qunshu Lin, Wenhao Huang, and Ge Zhang. SuperGPQA: Scaling LLM evaluation across 285 graduate disciplines. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2025.

[15] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

[16] Alex Gu, Baptiste Rozière, Hugh Leather, Armando Solar-Lezama, Gabriel Synnaeve, and Sida I Wang. Cruxeval: A benchmark for code reasoning, understanding and execution. *arXiv preprint arXiv:2401.03065*, 2024.

[17] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[18] Jujie He, Jiacai Liu, Chris Yuhao Liu, Rui Yan, Chaojie Wang, Peng Cheng, Xiaoyu Zhang, Fuxiang Zhang, Jiacheng Xu, Wei Shen, Siyuan Li, Liang Zeng, Tianwen Wei, Cheng Cheng, Bo An, Yang Liu, and Yahui Zhou. Skywork open reasoner series. https://capricious-hydrogen-41c.notion.site/Skywork-Open-Reaonser-Series-1d0bc9ae823a80459b46c149e4f51680, 2025. Notion Blog.

[19] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[20] Maggie Huan, Yuetai Li, Tuney Zheng, Xiaoyu Xu, Seungone Kim, Minxin Du, Radha Poovendran, Graham Neubig, and Xiang Yue. Does math reasoning improve general llm capabilities? understanding transferability of llm reasoning. *arXiv preprint arXiv:2507.00432*, 2025.

[21] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code. In *The Thirteenth International Conference on Learning Representations*, 2025.

[22] Guochao Jiang, Wenfeng Feng, Guofeng Quan, Chuzhan Hao, Yuewei Zhang, Guohua Liu, and Hao Wang. Vcrl: Variance-based curriculum reinforcement learning for large language models. *arXiv preprint arXiv:2509.19803*, 2025.

[23] William B Johnson, Joram Lindenstrauss, et al. Extensions of lipschitz mappings into a hilbert space. *Contemporary Mathematics*, 26(189-206):1, 1984.

[24] Pang Wei Koh and Percy Liang. Understanding black-box predictions via influence functions. In *International Conference on Machine Learning*, 2017.

[25] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph Gonzalez, Hao Zhang, and Ion Stoica. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th Symposium on Operating Systems Principles*, 2023.

[26] Derek Li, Jiaming Zhou, Leo Maxime Brunswic, Abbas Ghaddar, Qianyi Sun, Liheng Ma, Yu Luo, Dong Li, Mark Coates, Jianye Hao, et al. Omni-thinker: Scaling multi-task rl in llms with hybrid reward and task scheduling. *arXiv preprint arXiv:2507.14783*, 2025.

[27] Junlong Li, Daya Guo, Dejian Yang, Runxin Xu, Yu Wu, and Junxian He. Codeio: Condensing reasoning patterns via code input-output prediction. In *International Conference on Machine Learning*, 2025.

[28] Kaixin Li. Verified taco problems. https://huggingface.co/datasets/likaixin/TACO-verified, 2024. URL https://huggingface.co/datasets/likaixin/TACO-verified.

[29] Wen-Ding Li, Keya Hu, Carter Larsen, Yuqing Wu, Simon Alford, Caleb Woo, Spencer M Dunn, Hao Tang, Michelangelo Naim, Dat Nguyen, et al. Combining induction and transduction for abstract reasoning. *arXiv preprint arXiv:2411.02272*, 2024.

[30] Yu Li, Zhuoshi Pan, Honglin Lin, Mengyuan Sun, Conghui He, and Lijun Wu. Can one domain help others? a data-centric study on multi-domain reasoning via reinforcement learning. *arXiv preprint arXiv:2507.17512*, 2025.

[31] Xize Liang, Lin Yang, Jie Wang, Rui Liu, Yang Lu, Jinliang Zeng, Hanzhu Chen, Dong Li, and Jianye HAO. Boosting multi-domain reasoning of LLMs via curvature-guided policy optimization. In *The Fourteenth International Conference on Learning Representations*, 2026.

[32] Bill Yuchen Lin, Ronan Le Bras, Kyle Richardson, Ashish Sabharwal, Radha Poovendran, Peter Clark, and Yejin Choi. Zebralogic: On the scaling limits of llms for logical reasoning. *arXiv preprint arXiv:2502.01100*, 2025.

[33] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. *Advances in Neural Information Processing Systems*, 2021.

[34] Michael Luo, Sijun Tan, Roy Huang, Xiaoxiang Shi, Rachel Xin, Colin Cai, Erran Li, Raluca Ada Popa, Ion Stoica, Ameen Patel, Alpay Ariyak, Qingyang Wu, Maurice Weber, and Ce Zhang. Deepcoder: A fully open-source 14b coder at o3-mini level. https://pretty-radio-b75.notion.site/DeepCoder-A-Fully-Open-Source-14B-Coder-at-O3-mini-Level-1cf81902c14680b3bee5eb349a512a51, 2025. Notion Blog.

[35] Michael Luo, Sijun Tan, Justin Wong, Xiaoxiang Shi, William Y. Tang, Manan Roongta, Colin Cai, Jeffrey Luo, Li Erran Li, Raluca Ada Popa, and Ion Stoica. Deepscaler: Surpassing o1-preview with a 1.5b model by scaling rl. https://pretty-radio-b75.notion.site/DeepScaleR-Surpassing-O1-Preview-with-a-1-5B-Model-by-Scaling-RL-19681902c1468005bed8ca303013a4e2, 2025. Notion Blog.

[36] Xueguang Ma, Qian Liu, Dongfu Jiang, Ge Zhang, Zejun MA, and Wenhu Chen. General-reasoner: Advancing LLM reasoning across all domains. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[37] MAA. American mathematics competitions - amc. https://maa.org/, 2023.

[38] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. Teacher–student curriculum learning. *IEEE transactions on neural networks and learning systems*, 31(9):3732–3740, 2019.

[39] Justus Mattern, Sami Jaghouar, Manveer Basra, Jannik Straube, Matthew Di Ferrante, Felix Gabriel, Jack Min Ong, Vincent Weisser, and Johannes Hagemann. Synthetic-1: Two million collaboratively generated reasoning traces from deepseek-r1. https://www.primeintellect.ai/blog/synthetic-1-release, 2025.

[40] Bo Pang, Deqian Kong, Silvio Savarese, Caiming Xiong, and Yingbo Zhou. Reasoning curriculum: Bootstrapping broad llm reasoning from math. *arXiv preprint arXiv:2510.26143*, 2025.

[41] Abhishek Panigrahi, Bingbin Liu, Sadhika Malladi, Sham M. Kakade, and Surbhi Goel. In good GRACEs: Principled teacher selection for knowledge distillation. In *The Fourteenth International Conference on Learning Representations*, 2026.

[42] Shubham Parashar, Shurui Gui, Xiner Li, Hongyi Ling, Sushil Vemuri, Blake Olson, Eric Li, Yu Zhang, James Caverlee, Dileep Kalathil, and Shuiwang Ji. Curriculum reinforcement learning from easy to hard tasks improves LLM reasoning. In *The Fourteenth International Conference on Learning Representations*, 2026.

[43] Sung Min Park, Kristian Georgiev, Andrew Ilyas, Guillaume Leclerc, and Aleksander Mądry. Trak: attributing model behavior at scale. In *Proceedings of the 40th International Conference on Machine Learning*, 2023.

[44] Garima Pruthi, Frederick Liu, Satyen Kale, and Mukund Sundararajan. Estimating training data influence by tracing gradient descent. *Advances in Neural Information Processing Systems*, 2020.

[45] Shyam Sundhar Ramesh, Xiaotong Ji, Matthieu Zimmer, Sangwoong Yoon, Zhiyong Wang, Haitham Bou Ammar, Aurelien Lucchi, and Ilija Bogunovic. Multi-task grpo: Reliable llm reasoning across tasks. *arXiv preprint arXiv:2602.05547*, 2026.

[46] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. In *First Conference on Language Modeling*, 2024.

[47] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017.

[48] Abulhair Saparov and He He. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022.

[49] Abulhair Saparov, Srushti Pawar, Shreyas Pimpalgaonkar, Nitish Joshi, Richard Yuanzhe Pang, Vishakh Padmakumar, Seyed Mehran Kazemi, Najoung Kim, and He He. Transformers struggle to learn to search. *arXiv preprint arXiv:2412.04703*, 2024.

[50] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[51] Guangming Sheng, Chi Zhang, Zilingfeng Ye, Xibin Wu, Wang Zhang, Ru Zhang, Yanghua Peng, Haibin Lin, and Chuan Wu. Hybridflow: A flexible and efficient rlhf framework. In *Proceedings of the Twentieth European Conference on Computer Systems*, 2025.

[52] Georgios Tzannetos, Bárbara Gomes Ribeiro, Parameswaran Kamalaruban, and Adish Singla. Proximal curriculum for reinforcement learning agents. *arXiv preprint arXiv:2304.12877*, 2023.

[53] Wei Wang, Ye Tian, Jiquan Ngiam, Yinfei Yang, Isaac Caswell, and Zarana Parekh. Learning a multi-domain curriculum for neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020.

[54] Xin Wang, Yudong Chen, and Wenwu Zhu. A survey on curriculum learning. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):4555–4576, 2021.

[55] Zhenting Wang, Guofeng Cui, Yu-Jhe Li, Kun Wan, and Wentian Zhao. Dump: Automated distribution-level curriculum learning for rl-based llm post-training. *arXiv preprint arXiv:2504.09710*, 2025.

[56] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. *arXiv preprint arXiv:2402.04333*, 2024.

[57] Yunhui Xia, Wei Shen, Yan Wang, Jason Klein Liu, Huifeng Sun, Siyue Wu, Jian Hu, and Xiaolong Xu. Leetcodedataset: A temporal dataset for robust evaluation and efficient training of code llms. *arXiv preprint arXiv:2504.14655*, 2025.

[58] Sang Michael Xie, Hieu Pham, Xuanyi Dong, Nan Du, Hanxiao Liu, Yifeng Lu, Percy S Liang, Quoc V Le, Tengyu Ma, and Adams Wei Yu. Doremi: Optimizing data mixtures speeds up language model pretraining. *Advances in Neural Information Processing Systems*, 2023.

[59] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

[60] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[61] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 2020.

[62] Yanli Zhao, Andrew Gu, Rohan Varma, Liang Luo, Chien-Chin Huang, Min Xu, Less Wright, Hamid Shojanazeri, Myle Ott, Sam Shleifer, et al. Pytorch fsdp: experiences on scaling fully sharded data parallel. *arXiv preprint arXiv:2304.11277*, 2023.

[63] Yilun Zhao, Yunxiang Li, Chenying Li, and Rui Zhang. Multihiertt: Numerical reasoning over multi hierarchical tabular and textual data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, 2022.

[64] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 2023.

Transferability for General Reasoning: An Automated Curriculum for Multi-Domain RLVR

Supplementary Material

A Experiment Prompts

For all experiments, we use the default prompt templates from the VERL[2] [51] repository.

A.1 Math

User Prompt (Math)

```
{{question}} Please output the final answer within \boxed{}.
```

A.2 Code Generation

LeetCode2K. The user prompt is the cleaned dataset query (after removing `### Answer:` and normalizing `### Format:`):

User Prompt (LeetCode2K)

```
{{query}}
```

LiveCodeBench (Function Call).

User Prompt (LiveCodeBench, Function Call)

```
You are an expert Python programmer. You will be given a question
(problem specification) and will generate a correct Python program
that matches the specification and passes all tests.

Below is the question:

{{problem_desc}}

You will use the following starter code to write the solution to the
problem and enclose your code within ```python delimiters.
```python
{{starter_code}}
```
```

LiveCodeBench (STDIN).

User Prompt (LiveCodeBench, STDIN)

```
You are an expert Python programmer. You will be given a question
(problem specification) and will generate a correct Python program
that matches the specification and passes all tests.

Below is the question:
```

[2] https://github.com/volcengine/verl

```
{{problem_desc}}

Read the inputs from stdin, solve the problem, and write the answer to
stdout (do not directly test on the sample inputs). Enclose your code
within delimiters. Ensure that when the python program runs, it reads
the inputs, runs the algorithm, and writes output to STDOUT.
```

PrimeIntellect. The user prompt is the dataset's `problem` field as-is:

User Prompt (PrimeIntellect)

```
{{problem}}
```

TACO. The TACO prompt template is:

User Prompt (TACO, base template)

```
Solve the programming task below in a Python markdown code block.

{{question}}

{{starter_code_instruction_or_stdin_instruction}}
```

The placeholder `{{starter_code_instruction_or_stdin_instruction}}` is replaced by one of the two snippets below depending on whether the problem provides starter code or expects STDIN-based I/O:

TACO instruction: with starter code

```
You will use the following starter code to write the solution to the
problem and enclose your code within ‘‘‘python delimiters’’’.
‘‘‘python
{{starter_code}}
’’’
```

TACO instruction: STDIN

```
Read the inputs from stdin, solve the problem, and write the answer to
stdout (do not directly test on the sample inputs). Enclose your code
within ‘‘‘python delimiters’’’.
```

A.3 Logic and Visual Reasoning

User Prompt (ARC-AGI and BARC)

```
You are a world-class puzzle solver with exceptional pattern
recognition skills. Your task is to analyze puzzles, spot patterns,
and provide direct solutions.

Given input-output grid pairs as reference examples, carefully observe
the patterns to predict the output grid for new test input. Each pair
follows the same transformation rule. Grids are 2D arrays. Here are
the input and output grids for the reference examples:
----------------------------------------
```

```
{{training_data}}
----------------------------------------
Now, solve the following puzzle based on its input grid by applying
the rules you have learned from the training data.
----------------------------------------
{{input_test_data}}
----------------------------------------
What is the output grid? Please put your answer within <answer> and
</answer> tags, your final answer should be only the output grid
(2d array).
```

User Prompt (Graph Logical Reasoning)

```
{{question}} Please put your answer within <answer> and </answer>
tags, for example <answer> fdebme </answer>.
```

User Prompt (Ordering Puzzle)

```
{{instruction}}. The constraints are: {{constraints}}. Please put
your answer within <answer> and </answer> tags, for example
<answer> ['pigeon', 'sparrow', 'quail'] </answer>.
```

User Prompt (Zebra Puzzle)

```
{{instruction}} The clues are: {{clues}}. Output the grid in the form
of a dictionary with keys as header containing a list of the attributes
and rows denoting each row of the final grid. Please return the final
answer in <answer> </answer> tags, for example <answer>
{"header": ["Position", "Nationality", "Job"],
 "rows": [["1", "british", "plumber"],
          ["2", "polish", "carpenter"]]} </answer>.
```

A.4 Simulation

User Prompt (CodeI/O, Input Prediction)

```
You are given a question that requires some input and output variables
as follows:

{{problem_description}}

The input and output requirements are as follows:

{{io_requirements}}

Given the following output:

{{given}}

Can you predict a feasible input without writing any code? Please
reason and put your final answer in the following json format:
```

```
"input": <your input>, where <your input> should be a dictionary, even
if there is only one input variable, with keys strictly matching the
input variables' names as specified. Please put your answer in
\boxed{} tags.

Tip: Here is a reference code snippet for this question. You can refer
to this code to guide your reasoning but not copy spans of code
directly.

{{refcode}}

Please output the final answer in JSON format.
```

User Prompt (CodeI/O, Output Prediction)

```
You are given a question that requires some input and output variables
as follows:

{{problem_description}}

The input and output requirements are as follows:

{{io_requirements}}

Given the following input:

{{given}}

Can you predict the output without writing any code? Please reason and
put your final answer in the following json format:
"output": <your output>, where <your output> should strictly match the
output requirement as specified. Please put your answer in
\boxed{} tags.

Tip: Here is a reference code snippet for this question. You can refer
to this code to guide your reasoning but not copy spans of code
directly.

{{refcode}}

Please output the final answer in JSON format.
```

A.5 Table Reasoning

User Prompt (HiTab)

```
You are given one or more tables. Use the information in the table to
answer the following question.

{{tables}}

The question is:
```

```
{{question}}

Please output the final answer within \boxed{}. If there are multiple
answers, please output them separated by |.
```

User Prompt (MultiHierTT)

```
You are given one or more tables. Use the information in the tables to
answer the following question.

{{tables}}

The question is:

{{question}}

Please output the final answer within \boxed{}.
```

A.6 STEM

For OPENSCIENCEREASONING-2, the user prompt is the source dataset's `input` field as-is, with no additional template wrapping.

B Implementation Details

B.1 Datasets

Training data. We follow the GURU multi-domain reasoning suite [11] and aggregate training data from six high-level domains, replacing only the *stem* source. GURU provides per-domain corpora that have already been deduplicated, heuristically filtered, and difficulty-filtered using a weak/strong model pass-rate scheme (see Cheng et al. [11] for the full pipeline). We use these curated subsets directly, without re-applying any of GURU's upstream filtering steps. Per-domain source files and raw sizes after GURU's curation are summarized in Table 3.

Per-domain sources. We use the following sources, all inherited from GURU [11] except where noted:

- *Math.* Aggregated from OR1 [18], DAPO [60], and DeepScaler [35], which themselves compile competition-style problems including AIME [2] and AMC [37].
- *Codegen.* LeetCode [57], TACO-Verified [28], PrimeIntellect [39], and LiveCodeBench [21], with the PrimeIntellect and LiveCodeBench subsets adopted from the pre-filtered DeepCoder release [34].
- *Logic.* ARC-AGI-1/2 [12, 13] and BARC [29] (existing datasets), together with three synthesized symbolic-reasoning tasks introduced by GURU: Zebra Puzzle (following Lin et al. [32]), Ordering Puzzle, and Graph Puzzle (following Saparov and He [48], Saparov et al. [49]).
- *Simulation.* Code I/O on PyEdu [27], where the model predicts program inputs from outputs or outputs from inputs without executing code.
- *Table.* HiTab [10] and MultiHierTT [63], both linearized into markdown format. When multiple tables accompany a single query, they are concatenated with line breaks.
- *Stem.* We depart from GURU's default WebInstruct-Verified [36] source and instead use NVIDIA's OpenScienceReasoning-2[3], which provides higher-quality reasoning traces and is formatted as multiple-

[3] https://huggingface.co/datasets/nvidia/OpenScienceReasoning-2

Table 3: Training data sources by domain. Sizes are GURU's post-curation per-source counts (i.e., *before* our subsampling). Codegen and logic each pool multiple sub-sources; STEM uses our replacement source.

Domain	Source	Pre-subsample size
Math	OR1 / DAPO / DeepScaler [18, 60, 35]	54.4k
Codegen	LeetCode [57]	1.3k
	LiveCodeBench [21]	0.4k
	PrimeIntellect [39]	7.5k
	TACO-Verified [28]	8.8k
Logic	ARC-AGI-1 [12]	0.1k
	ARC-AGI-2 [13]	0.2k
	BARC [29]	1.6k
	Graph Puzzle [49]	1.2k
	Ordering Puzzle	1.9k
	Zebra Puzzle [32]	1.3k
Simulation	Code I/O (PyEdu) [27]	3.7k
Table	HiTab [10]	4.3k
	MultiHierTT [63]	1.5k
STEM	OpenScienceReasoning-2	1.4M

Table 4: Per-domain training caps after subsampling. The balanced mixture is used for the main results (Table 1); the imbalanced mixture is used for the stress test (Table 2).

Mixture	Math	Codegen	Logic	Simulation	Table	STEM
Balanced (main)	1,000	1,000	1,000	1,000	1,000	1,000
Imbalanced	1,500	1,000	1,000	500	500	1,500

choice questions with ground-truth answers. The latter point lets us replace GURU's 1.5B model-based verifier [36] with simple rule-based answer matching, eliminating verifier-model noise from the stem reward signal.

Reward design. Following GURU, all domains use binary verifiable rewards (1 if correct, 0 otherwise). With our stem replacement, the entire training suite uses only two verification modes: *(i)* rule-based matching after extracting answers from `\boxed{}` or `<answer>` tags, used for math, logic, simulation, table, and stem (the last via multiple-choice answer extraction); and *(ii)* execution-based verification, where generated programs are run against test cases under a 30-second timeout and 10 GB memory limit, with reward 1 granted only if all test cases pass (codegen). Notably, our pipeline avoids GURU's third reward mode—model-based verification with a 1.5B verifier for stem—which removes a learned-component noise source from RL training.

Subsampling. The raw mixture is dominated by math and stem by two orders of magnitude (Table 3), which would obscure the contribution of any curriculum if used directly. We therefore subsample the training set in two stages, governed by a fixed seed (`subsample_seed=42`) so every method trains on exactly the same data. First, each source file is uniformly subsampled at ratio 0.2. Second, the per-source results are pooled by domain and capped at a per-domain budget; within a domain, the cap is applied evenly across sub-sources (e.g., codegen draws roughly evenly from its four sources). The two training mixtures we use differ only in this per-domain cap, summarized in Table 4.

Validation set (online evaluation). During training we periodically evaluate the policy on a validation set drawn from the same six domains, used *only* for monitoring training dynamics. We do not perform validation-based checkpoint selection or hyperparameter tuning—all numbers reported in Table 1 come from each run's *final* checkpoint (see Appendix B.3). This validation set partially overlaps with the held-out

Table 5: Evaluation run multipliers per benchmark. Each benchmark is evaluated this many times per checkpoint, and the per-checkpoint result is the mean across runs. AIME applies an additional 8× in-set repetition before the run multiplier (32 effective samples per problem).

Benchmark	Runs per checkpoint
MATH-500, MBPP, HiTab, SuperGPQA	1
HumanEval, GPQA-Diamond, MultiHierTT, CruxEval	4
CodeI/O, ARC-AGI-1, Zebra Puzzle, FinQA	4
AIME	32

evaluation benchmarks reported in Table 1; because no model-selection or tuning decision is made on the basis of validation accuracy, this overlap does not affect the held-out numbers. Per-domain validation subsets:

- *Math.* MATH-500 (500 samples), AIME (240 samples, 8× repetition), AMC (332 samples, 4× repetition).
- *Codegen.* HumanEval (164 samples), MBPP (200 samples), LiveCodeBench (279 samples).
- *Logic.* Ordering puzzles (100 samples), Zebra puzzles (200 samples), ARC-AGI-1 (200 samples).
- *Simulation.* CodeI/O (200 samples).
- *Table.* HiTab (200 samples), MultiHierTT (200 samples).
- *Stem.* SuperGPQA (200 samples).

Evaluation set. The held-out evaluation benchmarks reported in the main paper (Table 1) cover the same 14 datasets across all six domains. To reduce evaluation noise, each benchmark is run multiple times per checkpoint, with the per-benchmark run multipliers shown in Table 5. The reported numbers in Table 1 are means over three training seeds with each entry additionally averaged over the corresponding number of evaluation runs (so most cells reflect $3 \times \{1, 4, 32\}$ samples per benchmark). All reported numbers come from each run's *final* checkpoint; we do not select checkpoints by validation accuracy.

B.2 Training Details

We implement `TAC` on top of the verl framework using the DAPO recipe (`recipe.dapo.main_dapo`), which provides a GRPO-style policy update with asymmetric clipping and dynamic batching. All baselines (Random, Math-to-Others, SEC) share the same DAPO configuration; only the data-sampling component differs. We split the configuration into method-related settings, which control the `TAC` curriculum, and RL-related settings, which control the underlying GRPO/DAPO update.

Method-related settings (`TAC`). The bandit operates on the six domains extracted from each example's `data_source` field. Q-values are initialized centered log-proportional to domain size (Eq. 14, κ=0.5), and five rounds (W=5) of round-robin warmup per epoch precede bandit control. When a domain's index pool is exhausted within an epoch, we reset that pool and continue sampling from it, so high-Q domains can be over-sampled within an epoch while no data is permanently withheld across epochs. The transferability signal is a Rademacher JL projection [43] of the per-domain gradient computed on the last N=4 transformer layers, with a fixed projection seed shared across all runs. The full hyperparameter list is given in Table 6.

Two-phase update bookkeeping. Because transferability is recomputed for every domain at each comparison step, `TAC` updates the unsampled arms there as well as the sampled one (§3.3). Two details govern how the learnability term enters those updates. First, what is stored per arm is the *normalized* learnability $\hat{L}_m$ (Eq. 7) from the last step the arm was pulled, not the raw L_m. When an unsampled arm is updated, this cached $\hat{L}_m$ is reused as-is and is *not* re-passed through the current normalizer. Re-normalizing a stale raw value through the live running mean and standard deviation, which keep drifting as other arms are sampled, would let an arm's effective learnability change without any new rollout for it; caching the already-normalized value pins each arm's learnability to the last time it was actually measured. Second, the normalizer statistics $\mu_L^{(t)}, \sigma_L^{(t)}$ are updated only on the sampled arm's fresh $L_{m_t}^{(t)}$, never during the unsampled-arm updates, so the

Table 6: `TAC` method-related hyperparameters.

Component	Hyperparameter	Value
Bandit	Selection rule	Boltzmann sampling over UCB-augmented Q (Eq. 5)
	Softmax temperature τ	0.85
	EMA learning rate α	0.3
	UCB exploration coefficient c	0.2
	Warmup rounds (round-robin) W	5 per epoch
	Initial Q distribution	centered log-proportional to domain size (Eq. 14, κ=0.5)
	Exhaustion strategy	reset
Composite signal	Mixing coefficient β	0.2
	Learnability normalization	EMA z-score (Eq. 7), decay 0.8, warmup 12 steps
	Comparison frequency K_c	2 steps
	Value update	two-phase (sampled arm every step; all arms at refresh steps)
Transferability	Type	projected-gradient cosine (Eqs. 10, 12)
	Parameter subset	last N=4 transformer layers
	Projection source	final PPO mini-batch of each step
	Projection method	Rademacher JL [43]
	Projection dimension r	4096
	Per-domain gradient EMA γ	0.8
	Temporal smoothing δ	0.8
	Cross-domain normalization	min–max (Eq. 12), scale-EMA δ_s=0.9
	Scale floor $s_{\min}$	0.01

running statistics track only genuinely observed advantages. Arms never yet pulled have no cached $\hat{L}_m$ and are skipped entirely until their first selection.

RL-related settings (DAPO/GRPO). We use a DAPO-style [60] update with grouped advantages, asymmetric clipping (ϵ_{low}=0.2, ϵ_{high}=0.4), no KL regularization, and dynamic per-GPU token budgets. The base models are `Qwen3-1.7B-Base` [59] and `Llama3.2-3B-Instruct` [15]; both are trained with FSDP [62] and full parameter/optimizer offloading. Generation uses vLLM [25] in synchronous mode. Validation evaluation is run before the first training step and every 20 update steps thereafter. The detailed configuration is given in Table 7.

Compute. All training runs use a single node with 4× NVIDIA H100 80 GB GPUs, 16 CPU cores, and 512 GB of system memory. With FSDP, full parameter/optimizer offloading, and a 12K-token context window, two epochs of GRPO on the balanced mixture take approximately 7 hours of wall-clock time per seed (≈28 GPU-hours). `TAC` adds less than 1% wall-clock overhead per training step relative to the other curricula (Figure 6), so the same compute budget is used for every method to ensure a fair comparison.

B.3 Evaluation Details

Pipeline. Held-out evaluation is run offline after training completes, on saved checkpoints. For each checkpoint, we merge the trained model from its sharded format into a HuggingFace-format model, load it into vLLM, generate responses for every benchmark in a single inference session to amortize the engine load cost, and score the outputs against verifiable references on CPU. Generation outputs are cached to disk, allowing the scoring pass to be re-run without regenerating responses if scoring logic changes.

Generation settings. The vLLM engine is initialized once per checkpoint with the maximum prompt and response lengths across all benchmarks, so a single engine serves every benchmark without reloading weights. Sampling uses temperature 1.0 and top-p 1.0, matching the training-time sampling configuration. Engine-wide settings are listed in Table 7.

Scoring and aggregation. Each benchmark uses a verifiable reward function appropriate to its domain: exact-match against ground truth for math (after `\boxed{}` extraction), unit-test execution for codegen, structural answer parsing for logic and table benchmarks, and multiple-choice answer extraction for stem.

Table 7: RL-related (DAPO/GRPO) hyperparameters.

Component	Hyperparameter	Value
Algorithm	Advantage estimator	GRPO (group-relative)
	Clip ratio (low / high)	0.2 / 0.4
	Use KL in reward / loss	False / False
	Loss aggregation	token-mean
Optimizer	Optimizer	AdamW
	Learning rate	1×10^{-6}
	LR schedule / warmup	constant, 10 steps
	Weight decay	0.1
	Gradient clipping	1.0
Batching	Train prompt batch size	64 (single-domain)
	PPO mini-batch size	16
	Rollouts per prompt K	4
	Max tokens / GPU	$2(L_p{+}L_r) = 24{,}576$
Sequence	Max prompt length L_p	4096
	Max response length L_r	8192
	Sequence parallelism	1
	Tensor parallelism (rollout)	1
Sampling	Temperature	1.0
	top-p	1.0
	vLLM GPU memory utilisation	0.7
Trainer	Total epochs	2

Each benchmark produces a single mean accuracy. We aggregate within each domain by taking the unweighted mean of its benchmark accuracies, then report the unweighted mean across the six domains as the overall score in Table 1. All numbers are means over three training seeds; the seeds vary the bandit sampling sequence, while the data subsampling seed is held fixed across methods so every method trains on identical data.

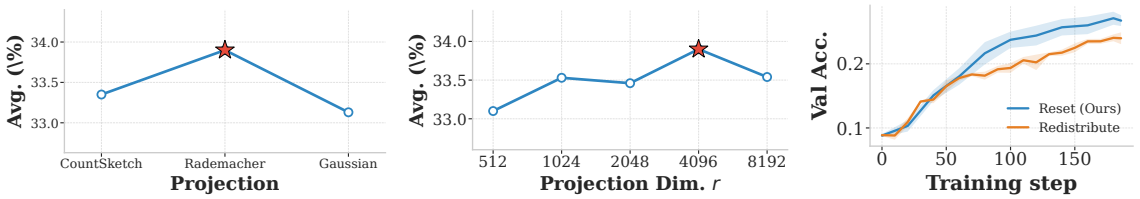
C More Experiments

C.1 More Ablation Study

We complement the main hyperparameter sweeps in Section 4.2 (Figure 4) with three additional ablations targeting design choices in the transferability pipeline. Figure 5 reports macro-averaged accuracy across all 14 evaluation benchmarks; we vary one factor at a time while holding all other hyperparameters at their default values from Table 6.

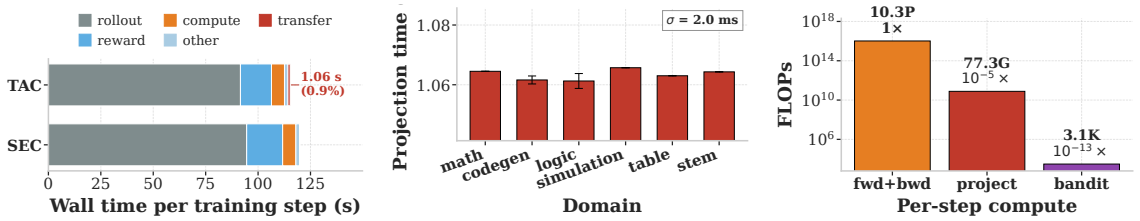
Projection method. We compare the three random projection families used in the gradient-attribution literature: CountSketch [6], Rademacher [1] (`TAC`'s default), and Gaussian. As shown in Figure 5a, the Rademacher projection outperforms the others by roughly 0.5–0.7 points, while CountSketch and Gaussian are within 0.2 points of each other. We attribute Rademacher's edge to its sub-Gaussian tail behavior, which preserves inner products with low variance at the projection dimension (r=4096) used here, consistent with its use as the default in TRAK [43].

Projection dimension. We sweep the projection dimension $r \in \{512, 1024, 2048, 4096, 8192\}$. Performance improves up to r=4096 and then flattens, with r=8192 giving no further gain and the smaller dimensions losing roughly a point of macro accuracy (Figure 5b). The plateau at the top end is consistent with the Johnson-Lindenstrauss bound [23]: once r is large enough to preserve pairwise inner products at the granularity the cosine signal needs, further increases yield diminishing returns. We use r=4096 as the default, at the knee of the curve.



(a) Projection method. **(b)** Projection dimension r. **(c)** Data-exhaustion strategy.

Figure 5: Additional ablations. (a) Macro-averaged accuracy (unweighted mean over the six domain scores) across choices of random projection family (CountSketch, Rademacher, Gaussian). **(b)** Macro accuracy across projection dimensions $r \in \{512, 1024, 2048, 4096, 8192\}$. **(c)** Validation accuracy over training under two data-exhaustion strategies: *reset* (TAC's default) reshuffles a depleted domain in place; *redistribute* re-draws from the remaining domains.



(a) Per-step time breakdown. **(b)** Projection cost by domain. **(c)** Per-step FLOPs.

Figure 6: Wall-clock overhead of TAC's transferability signal (`Qwen3-1.7B-Base`, 4×H100, batch 64 with $K=4$ rollouts). **(a)** Per-step decomposition into *rollout* (vLLM generation), *reward*, *compute* (forward+backward+optimizer), *other* (old-logprob+advantage), and *transfer* (gradient projection + bandit Q-update). TAC adds 1.06 s on top of a 115 s step (0.9%); SEC has no *transfer* slice. **(b)** Projection cost is essentially constant across the six training domains ($\sigma = 2$ ms on a 1.06 s mean), confirming that overhead depends only on parameter count and projection dimension, not on batch composition. **(c)** Analytic FLOP cost: projection is structurally $\sim 10^{-5}\times$ the forward/backward compute and the bandit update $\sim 10^{-13}\times$, well below the optimizer-step noise floor.

Data-exhaustion strategy. When a domain's index pool is exhausted within an epoch, the curriculum must decide what to do at the next time the bandit selects that domain. We compare two strategies: *reset* (TAC's default), which reshuffles the depleted domain's data and continues sampling from it; and *redistribute*, which re-draws m_t from the remaining domains. Figure 5c shows that *reset* converges to higher validation accuracy throughout training, with the gap widening as more domains exhaust. *Redistribute* forces the sampler to pick among lower-Q domains late in an epoch simply because their data happens to remain, dragging down the effective curriculum quality. Resetting instead lets high-Q domains continue to be over-sampled within an epoch, while still guaranteeing that no data is permanently withheld across epochs.

C.2 Complexity Analysis

A central design goal of TAC is that the transferability signal should add negligible compute on top of standard GRPO. We verify this empirically by profiling each step on `Qwen3-1.7B-Base`, with a per-step batch of 64 prompts and $K=4$ rollouts on 4×H100 GPUs. The profiled run uses $r=1024$; at our default $r=4096$ the projection cost grows by $4\times$ but remains far below the noise floor of the optimizer step, so the conclusions below are unchanged.

TAC's transferability machinery (gradient projection plus the bandit Q-update) adds only 1.06 s to a 115 s step—a 0.9% overhead (Figure 6a). The dominant per-step costs remain rollout generation (∼70 s) and the forward/backward/optimizer pass (∼35 s); SEC, which uses no transferability signal, runs at the same wall-clock as TAC minus this 1.06 s, confirming the bandit machinery itself is free and the entire cost lies in the projection.

The overhead is also stable across domains. Figure 6b reports projection wall-time per training step grouped by the domain of the sampled minibatch. The standard deviation across domains is 2 ms on a 1.06 s mean (0.2%), confirming that the projection cost depends only on parameter count and projection dimension—

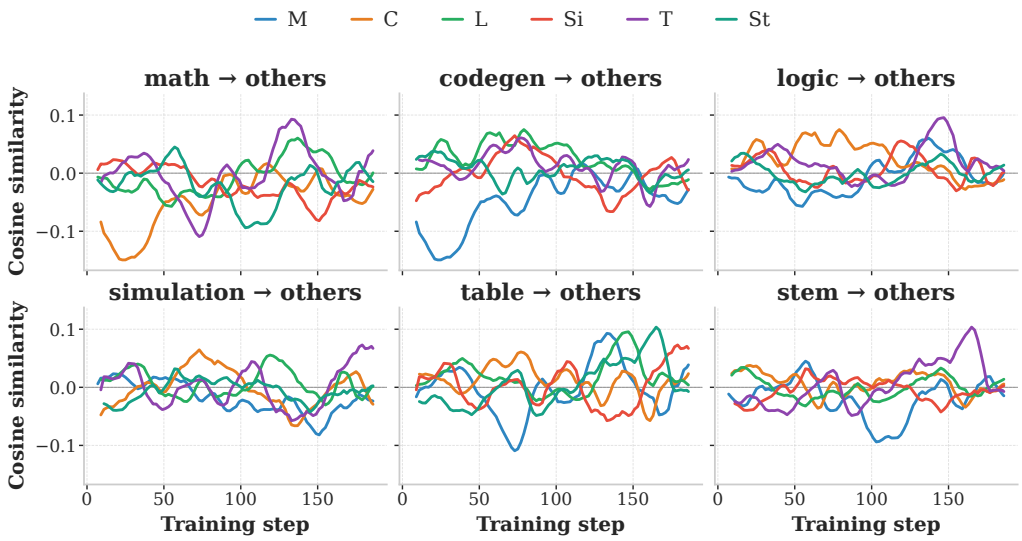


Figure 7: Pairwise gradient cosine similarity over training, per source domain. Each panel plots the cosine similarity between the projected-gradient EMA $\mathbf{h}_m^{(t)}$ of one *source* domain m (panel title) and the EMAs of the five other *target* domains (line color), averaged across three TAC seeds and smoothed with a 10-step running mean. The aggregate transferability signal $T_m^{(t)}$ consumed by the bandit (Eq. 12) is a cross-domain–normalized transform of each row's mean cosine (Eq. 10). Absolute magnitudes are small ($|\cos| \lesssim 0.15$)—domain pairs are only weakly aligned in the projected subspace—but the *relative* structure is stable and coherent: *table* acts as a hub, ending positively aligned with *simulation* and *stem* (its two strongest bonds) and with *logic*, which is why it carries the highest aggregate T in Figure 3b; *math* and *codegen*, by contrast, sit at negative cosine with nearly every domain—and most negatively with *each other*—explaining their lowest aggregate T.

not on batch composition or response length. The ℓ_2 normalization in Eq. 8 is a single vector-norm and contributes no measurable cost.

Figure 6c gives the analytic picture. The full forward/backward step costs roughly 10^{16} FLOPs (≈10.3 PFLOPs); the projection $P^\top g_t$ applied to the last $N{=}4$ transformer layers' gradients adds approximately 7.7×10^{10} FLOPs at the profiled $r{=}1024$ (a $\sim 10^{-5}\times$ overhead), rising to $\sim 3.1 \times 10^{11}$ at our default $r{=}4096$, and the bandit Q-update is $\mathcal{O}(M)$ scalar operations across $M{=}6$ domains ($\sim 3 \times 10^3$ FLOPs, $\sim 10^{-13}\times$). Both are well below the noise floor of the optimizer step itself. The empirical 0.9% wall-clock overhead measured in panel (a) is dominated by GPU–CPU transfer of the projected gradient and Python-side bookkeeping, not by the projection arithmetic itself, suggesting further reductions are possible but would have negligible practical effect.

Together, these measurements confirm TAC's design goal: the transferability signal is effectively free at training time. Wall-clock overhead is sub-1%, stable across domains and batch compositions, and the analytic FLOP cost is 10^{-5} relative to the optimizer step itself. Practitioners can adopt TAC as a drop-in replacement for fixed-mixture or learnability-only sampling without modifying their training budget.

C.3 A Pairwise View of Transferability

Figure 7 decomposes the aggregate $T_m^{(t)}$ signal into its underlying per-pair cosines, with one panel per source domain. Two observations are worth emphasizing. First, the absolute magnitudes are small: nearly all pairs stay within $|\cos| \lesssim 0.15$ across training, confirming that gradient agreement in the projected subspace is weak in absolute terms. The cross-domain normalization of Eq. (12) is therefore essential—raw cosines alone would be uninformative as a curriculum signal, but the *relative* ordering across domains is robust enough to drive sample reallocation. Second, that ordering has a clear geometric structure rather than being noise. *Table* behaves as a hub: its gradients end positively aligned with *simulation* and *stem* (its two largest cosines) and with *logic*, so training on table moves the policy in directions that also help those domains—the

Table 8: Additional results across two model sizes (`Qwen3-0.6B-Base` and `Qwen3-4B-Base`) using the same training data setup and baselines as the main table. Each configuration is run with a single seed.

Method	Codegen		Logic		Math		Simulation			STEM		Table			All
	HE	MBPP	ARC-AGI	Zebra	MATH	AIME	CodeI/O	CruxEval-I	CruxEval-O	GPQA	SuperGPQA	MH	FinQA	HiTab	*macro avg.*
Qwen3-0.6B															
Random	**32.9**	**37.0**	0.3	5.6	28.6	1.5	**5.1**	26.9	28.6	27.7	15.6	4.2	0.4	36.5	18.1
SEC	30.5	32.4	0.3	2.6	44.4	0.9	2.8	**33.4**	35.2	**29.5**	15.0	12.0	3.8	37.9	19.9
`TAC`	32.0	35.6	**0.5**	**13.4**	**47.0**	**1.7**	2.6	29.6	**35.6**	27.5	**16.9**	**12.3**	**4.5**	**41.5**	**21.6**
Qwen3-4B															
Random	83.2	**67.4**	**4.5**	36.4	74.4	10.8	4.2	59.6	47.6	41.3	30.5	44.9	37.0	61.4	43.2
SEC	81.1	66.6	4.2	35.6	73.4	**12.9**	5.2	59.5	**62.7**	36.7	29.9	45.4	38.9	65.4	43.8
`TAC`	**83.5**	**67.4**	2.6	**36.9**	**76.2**	12.3	**5.6**	**61.4**	62.1	**42.5**	**32.0**	**47.7**	**39.7**	**70.1**	**45.4**

geometric origin of table's top-ranked T in Figure 3b. *Math* and *codegen*, by contrast, sit at negative cosine with almost every other domain, and their *mutual* cosine is the most negative pair in the matrix; their updates pull the policy in idiosyncratic directions that do not benefit—and often oppose—the rest of the mixture, which is why they carry the lowest aggregate T and `TAC` samples them least. We read this isolation as a pretraining effect: base models, and `Qwen3` in particular, are already saturated on math and code, so RL there sharpens an already-formed skill along directions roughly orthogonal to the shared reasoning gradients the other domains move along.

C.4 Scaling Experiments

Table 8 shows that `TAC`'s gains generalize across model scales within the Qwen3 family. On `Qwen3-0.6B-Base`, `TAC` reaches 21.6 macro accuracy against 18.1 (Random) and 19.9 (SEC), gaps of $+3.5$ and $+1.7$, ranking first on $9/14$ benchmarks. On `Qwen3-4B-Base`, `TAC` reaches 45.4 versus 43.2 (Random) and 43.8 (SEC), gaps of $+2.2$ and $+1.6$, ranking first on $10/14$. Combined with the $+2.1$/$+1.8$ over Random/SEC at 1.7B in Table 1, the per-scale gains over SEC are comparable ($+1.6$ to $+1.8$) and `TAC` improves over Random at every scale ($+2.1$ to $+3.5$): neither vanishing at the smaller backbone where rollouts are noisier, nor washing out at the larger one where the gradient geometry is higher-dimensional. We attribute this scale-invariance to the signal being computed from each model's own training gradients rather than any fixed external probe—as the policy changes shape with scale, T adapts with it.

D More Examples

The following examples compare rollouts from models trained with `TAC` and with the *Random* curriculum baseline. We sample one example each from MATH-500 [19], GPQA-DIAMOND [46], and ZEBRA-PUZZLE [32], drawn from final checkpoints of `Qwen3-1.7B-Base` runs. The correct final answer is highlighted in green and an incorrect one in red. These examples are illustrative rather than systematic: they showcase the kinds of qualitative differences we observe between the two curricula on representative problems from three of the six training domains, with `TAC`-trained models more consistently following through the right chain of reasoning where Random-trained models stall on early misidentifications or fail to invoke the relevant principle (relativistic time dilation in the GPQA case, parity propagation in the Zebra case).

Example 1: Algebra (`MATH`, `Qwen3-1.7B-Base`)
Random vs. `TAC`

Question. What is the smallest positive integer n such that all the roots of $z^4 + z^2 + 1 = 0$ are n^{th} roots of unity?

Ground-truth answer. $\boxed{6}$

Response from Random-trained model. Substituting $w = z^2$ gives $w^2 + w + 1 = 0$, so:

$$w = \frac{-1 \pm i\sqrt{3}}{2}$$

...

The model identifies $\frac{-1+i\sqrt{3}}{2} = e^{i\pi/3}$ (incorrect: the argument is $2\pi/3$).

...

Taking square roots under this misidentification yields:

$$z \in \{e^{\pm i\pi/6},\ e^{\pm i5\pi/6}\}$$

The model concludes these are 12th roots of unity.

Final Answer

$$\boxed{12}$$

Response from `TAC`-trained model. Substituting $w = z^2$ gives $w^2 + w + 1 = 0$, so:

$$w = \frac{-1 \pm i\sqrt{3}}{2} = e^{\pm 2\pi i/3}$$

...

The four roots of the original quartic are:

$$z = \pm e^{\pm i\pi/3}$$

...

Since $e^{\pm i\pi/3}$ are primitive 6th roots of unity, and $-e^{\pm i\pi/3}$ are also 6th roots of unity, all four roots are 6th roots of unity.

Final Answer

$$\boxed{6}$$

Example 2: Particle Physics (`GPQA-Diamond`, `Qwen3-1.7B-Base`)
Random vs. `TAC`

Question. In the CERN Bubble Chamber a decay $X^0 \to Y^+ Z^-$ occurs with proper lifetime $\tau_0 = 8 \times 10^{-16}$ s. What minimum spatial resolution is needed to observe at least 30% of the decays, given an energy of 27 GeV and mass $m_{X^0} = 3.41$ GeV?
A) 2.08×10^{-3} m B) 2.08×10^{-6} m C) 2.08×10^{-1} m D) 2.08×10^{-9} m

Ground-truth answer. $\boxed{\text{B}}$

Response from Random-trained model. The model attempts to set up a decay-rate calculation:

$$\lambda = \frac{1}{\tau_0}$$

...

It tries to convert "number of decays" into a spatial resolution via $\Delta x = v \cdot t$, leaving the decay count N as a free parameter.

...

The chain stalls without setting up the relativistic boost. Across four rollouts, the model selects A or D three times.

Final Answer

$$\boxed{\text{D}}$$

Response from `TAC`-trained model. Recognizing this as a time-dilation problem, we compute the Lorentz factor:

$$\gamma = \frac{E}{m} = \frac{27}{3.41} \approx 7.92$$

...

The lab-frame decay length is:

$$d = \gamma \, \tau_0 \, c \approx 7.92 \cdot 8 \times 10^{-16}\,\text{s} \cdot 3 \times 10^{8}\,\text{m/s}$$

$$d \approx 1.9 \times 10^{-6}\,\text{m}$$

...

This matches option B to within rounding.

Final Answer

$$\boxed{\text{B}}$$

Example 3: Constraint Satisfaction (`Zebra-Puzzle`, `Qwen3-1.7B-Base`)
Random vs. `TAC`

Question (abridged). Two positions, each with attributes Beverage ∈ {iced-tea, tea}, Hobby ∈ {traveling, writing}, Movie-Genre ∈ {fantasy, spy}, Music-Genre ∈ {d&b, hip-hop}, Nationality ∈ {australian, russian}, Pet ∈ {ferret, fish}.
Clues: `Beverage:tea` is on the far right; `Hobby:traveling` and `Music-Genre:hip-hop` have the same parity; `Music-Genre:d&b` is left of `Hobby:traveling`; `Movie-Genre:fantasy` is left of `Movie-Genre:spy`; `Music-Genre:hip-hop` and `Nationality:australian` have the same parity; `Beverage:iced-tea` is left or right of `Pet:ferret`.

Ground-truth grid.

```
[[1, iced-tea, writing, fantasy, d&b, russian, fish],
 [2, tea, traveling, spy, hip-hop, australian, ferret]]
```

Response from Random-trained model. The model lists and restates the clues but never propagates the parity constraints.

...

One rollout overwrites the schema with the example header from the prompt ("Nationality, Job"); another invents extra rows.

...

Average cell-level accuracy across four rollouts: ∼36%.

Final Answer

Inconsistent grid (schema mismatch).

Response from TAC-trained model. All four rollouts return the requested JSON in the correct schema:

```
<answer>{"header": ["Position", "Beverage", "Hobby",
"Movie-Genre", "Music-Genre", "Nationality", "Pet"],
"rows": [["1", "iced-tea", "writing", "fantasy", "d&b",
"russian", "fish"], ["2", "tea", "traveling", "spy",
"hip-hop", "australian", "ferret"]]}</answer>
```

Final Answer

Correct grid (89–100% cell-level accuracy across rollouts).