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
https://github.com/YangYongJin/transfer-aware-curriculum

Abstract

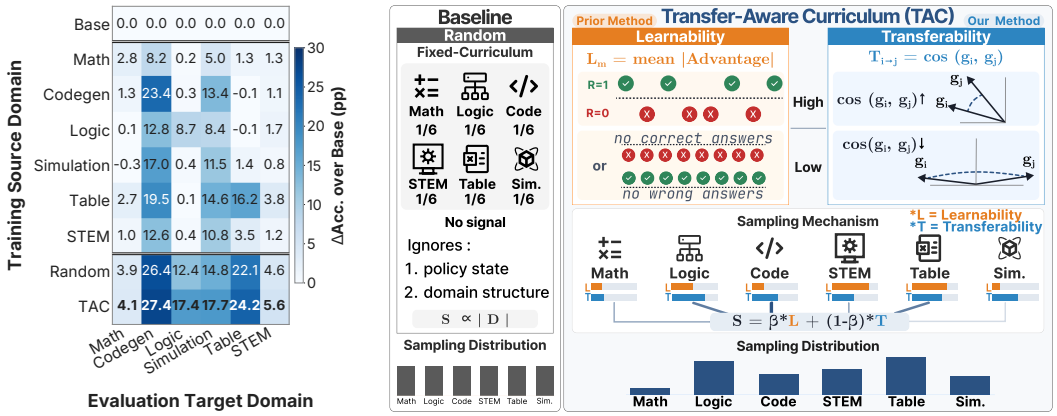
Reinforcement learning with verifiable rewards (RLVR) has been extended from single-domain training to multi-domain reasoning suites spanning mathematics, programming, and science. However, the training curriculum (how often each domain is sampled) is typically fixed or hand-tuned, even though reasoning 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` reuses 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` keeps its lead on an imbalanced training mixture where learnability-only curricula over-commit to dominant domains. Our results suggest that cross-domain transferability is a useful 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

40th Conference on Neural Information Processing Systems (NeurIPS 2026).



(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 widely 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.

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 scoped, so that gains on it fail to transfer. Conversely, a less obviously learnable domain may 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 and miss much of the achievable cross-domain improvement.

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 over the six domains (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 D.3). Ablations isolate the transferability term, and `TAC` keeps its lead 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 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. Figure 1b and Algorithm 1 summarize the full procedure, and Section 3.3 gives the details.

3.1 Local Learnability Signal

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

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

serves as an on-policy proxy for learnability [8, 55]. With binary rewards, it is large when the rollout group mixes successes and failures. It collapses to zero when all K rollouts share the same reward, either because D_m is saturated or because it is beyond the policy's current 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.

3.2 Gradient-Based Transferability

Beyond learnability, a good curriculum should favor domains whose gradient updates are 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 keeps only the update direction, so a single step with an unusually large gradient cannot dominate the per-domain state below. We apply no response-length rescaling to $\mathbf{v}_t$, since response length varies systematically across domains and would bias the accumulated direction. P is generated once with a fixed seed and reused throughout training (Appendix C.1).

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 $\gamma \in [0, 1)$. Unselected domains retain their previous states.

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. The raw transfer of a domain changes at every comparison even when it was not sampled, because the other domains' gradient states have moved. To damp step-level variance, we smooth each raw cosine with a per-domain EMA at these comparison steps,

$$\tilde{\rho}_m^{(t)} \;=\; \delta\, \tilde{\rho}_m^{(t-1)} + (1-\delta)\, \rho_m^{(t)}, \qquad m \in \mathcal{A}_t,\;\; t \bmod K_c = 0, \tag{11}$$

with smoothing coefficient $\delta \in [0, 1)$. Between comparisons, $\tilde{\rho}_m$ 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 D.4), 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 $s_{\min}$. This pins the worst-transferring domain near 0 and the best near 1. The floor prevents one-step spikes when the cross-domain spread briefly collapses (Appendix C.1).

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 are on deliberately different scales. $T_m^{(t)}$ is a bounded ranking in $[0, 1]$, while $\hat{L}_{m_t}^{(t)}$ is an unbounded z-score, and we set β empirically (Table 6).

Two-phase value update. A standard bandit refreshes only the arm it pulls. TAC recomputes transferability for *every* domain at each comparison step, so it also has fresh feedback for the arms it did not pull. The sampled arm m_t is updated every step via Eq. (4). At each comparison step, every *unsampled* arm is also updated by the same rule, using its fresh $T_m^{(t)}$ and the learnability cached from its last pull. This lets transferability reallocate sampling mass even to domains the bandit is not revisiting (Appendix C.1).

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

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 $\mu_m^{(0)} \propto |D_m|^{\kappa/\tau}$. In the balanced setting this reduces exactly to uniform initialization. At the start of each epoch, the sampler runs W rounds of round-robin warmup over all M domains before applying Eq. (5). The bandit and normalizers update during warmup, so every domain has several gradient-state updates before transferability drives sampling (Appendix C.1).

Algorithm 1 summarizes the full procedure. When a domain's pool is exhausted within an epoch, we reshuffle it and keep sampling, so no data is withheld across epochs. Appendix C gives the remaining implementation details.

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]. Averaging over $j \neq m_t$ gives the aggregate cross-domain improvement summarized by $\rho_{m_t}^{(t)}$. 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 across domains and training stages. Appendix C.1 discusses why measured transfer can still be asymmetric. We compute it in a low-dimensional sketch where random projections preserve inner products in expectation [23, 43], and use the EMAs $\mathbf{h}_m^{(t)}$ in place of instantaneous gradients to reduce variance. The signal is therefore *adaptive* because it tracks the current gradient geometry as π_θ evolves. It is also *relative* because Eq. (12) rewards a domain for transferring better than the other domains, not for having large gradients. Eq. (13) thus asks whether a step on D_m both carries local learning signal and points 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 would 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). Appendices C.2, C.4, and B give the training data, evaluation pipeline, and per-domain prompts.

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. We additionally report scaling results on `Qwen3-{0.6, 4}B-Base` in Appendix D.5.

Baselines. We compare `TAC` against four baselines under identical data and compute budgets: *(i)* **Base**, the model without RL training; *(ii)* **Random**, the standard multi-domain baseline with no online adaptation, which draws each single-domain batch from a domain sampled with fixed probability proportional to its pool size (uniform when pools are balanced); *(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 against adaptive scheduling, and SEC isolates the contribution of transferability. Additionally, Appendix D.1 compares `TAC` with four further adaptive curricula.

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. We use $\gamma = 0.8$ for the gradient EMAs, recompute cosines every $K_c = 2$ steps, and smooth them with $\delta = 0.8$. The learnability and transferability terms are mixed with $\beta = 0.2$. Table 6 lists the remaining hyperparameters, and Appendices C.3 and D.3 give compute details and complexity analysis. We observe that the transferability computation adds 1.06 s to a 115 s training step (0.9%, Appendix D.3).

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.

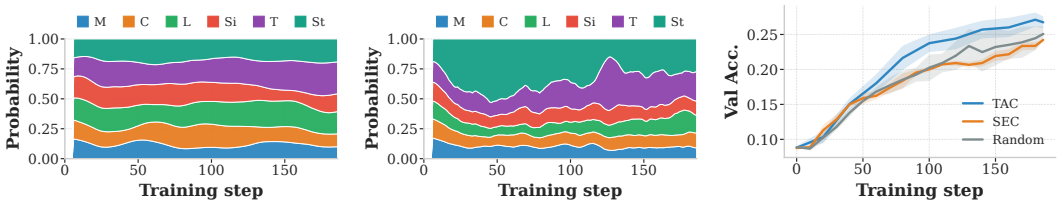
The two adaptive baselines behave inconsistently. M2O improves over Random on both backbones ($+0.2$ on Qwen, $+0.5$ on Llama), confirming that prior domain knowledge yields a useful schedule, but its static two-stage structure cannot react to mid-training shifts in cross-domain transfer. SEC is far 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). A learnability-only bandit anchors on whichever domain gives the strongest early advantage signal (see Curriculum Dynamics below), which pays off on one backbone but not the other. `TAC`'s transferability term avoids this failure mode.

The gains are broad rather than concentrated. The largest per-benchmark gains appear in logic, math, and table reasoning: Zebra rises by $+9.8$ over Random on Qwen, while MATH rises by $+5.0$ and FinQA by $+7.5$ on Llama. Zebra also rises by $+8.0$ over SEC on Qwen, and MultiHierTT rises by $+3.5$ on Llama and $+3.2$ on Qwen over Random. Gains also extend to stem, with GPQA rising by $+1.9$ over Random on Qwen. On `Llama3.2-3B`, `TAC` leaves none of the 14 benchmarks below the pre-RL model, while Random and M2O leave 4 each and SEC leaves 5. A curriculum that ignores transfer can improve some domains at the cost of others, and the macro average partly hides this. The only benchmarks where `TAC` is not first on either backbone are CodeI/O and CruxEval-O in

[1] https://huggingface.co/datasets/nvidia/OpenScienceReasoning-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. *All* averages the benchmarks within each domain and then averages the six domain scores.

		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.)		$17.8_{\pm 0.1}$	$31.8_{\pm 0.5}$	$32.0_{\pm 0.2}$	$32.1_{\pm 0.6}$	$\mathbf{33.9}_{\pm 0.2}$	$22.2_{\pm 0.2}$	$29.2_{\pm 0.2}$	$29.7_{\pm 0.3}$	$28.5_{\pm 1.5}$	$\mathbf{31.3}_{\pm 0.6}$



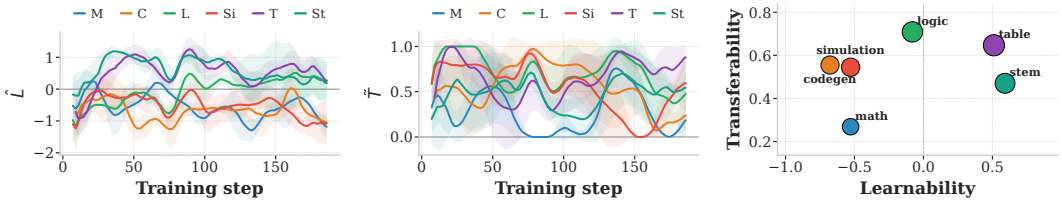
(a) Sampling under TAC. (b) Sampling under SEC ($\beta = 1$). (c) Macro validation accuracy.

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.

the simulation domain. There, learnability and transferability are both relatively flat (Figure 3) and the curriculum has limited leverage. Although `Llama3.2-3B` starts much weaker than `Qwen3-1.7B` (34.4 vs. 52.8 on MATH), TAC improves over Random by the same $+2.1$ macro points on both backbones, which we attribute to the transferability signal being computed from each model's own training gradients. The gains hold across model scale, with TAC improving over Random by $+3.5$ and $+2.2$ macro points on `Qwen3-0.6B` and `Qwen3-4B` (Appendix D.5). None of the four further curricula in Appendix D.1 improves over Random by more than 0.3 points, while TAC exceeds the strongest of them by $+1.8$ and $+2.0$ points. Appendix E shows qualitative examples of model rollouts.

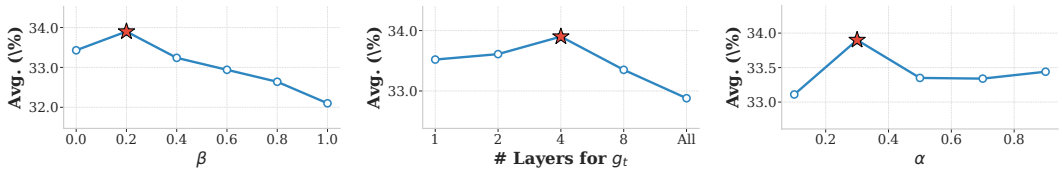
Curriculum Dynamics: TAC Adapts, SEC Anchors. Both TAC and SEC begin near proportional initialization and behave similarly for ~ 30 steps. Around step 30–50, *stem* learnability spikes (Figure 3a) and SEC's bandit collapses onto it. By step ~ 60, stem occupies about half of SEC's sampling mass (Figure 2b) and stays the dominant domain thereafter. TAC up-weights stem only transiently (Figure 2a). It then releases stem as *table* overtakes every other domain in transferability (Figure 3b), and table reaches its largest sampling share by the end of training. The most learnable domain (stem) and the most transferable domain (table) are therefore not the same. The validation curves (Figure 2c) follow the same timeline. The three methods are indistinguishable until step ~ 40, after which TAC pulls ahead while SEC tracks Random closely, indicating that the bandit machinery alone does not improve over uniform sampling.

Learnability and Transferability Favor Different Domains. Figures 3a and 3b track $\hat{L}$ and T within the same TAC run. Early in training ($t \lesssim 50$), learnability spikes for stem while T is uniformly



(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.



(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.

high and not yet discriminative, so SEC and TAC make similar decisions. Afterwards T separates the domains clearly. *Table* climbs to the highest transferability while *codegen* and *math* drift to the bottom, and stem's $\hat{L}$ falls as it is trained down. Table's $\hat{L}$ holds over the same period, so TAC's sampling moves away from SEC's toward table. Late in training, stem's $\hat{L}$ keeps falling while logic's climbs, so the two orderings stay misaligned rather than converging. Averaged over the run (Figure 3c), stem has high $\hat{L}$ but only moderate T, and *logic* has low $\hat{L}$ but high T. Table is high on both axes and sampled most, while math is low on both and sampled least. A learnability-only curriculum therefore locks onto stem and under-weights table and logic, which TAC corrects through T.

This ranking agrees with the offline transfer matrix in Figure 1a, where *table* as a single source yields the largest off-diagonal gains and *math* the smallest. The two need not match exactly, since the matrix measures end-to-end accuracy transfer from training each domain alone, while TAC's signal is an online first-order estimate that evolves with the policy. The per-pair cosines in Appendix D.4 show the same structure. *Table* aligns positively with simulation, stem, and logic, while *math* and *codegen* align weakly or negatively with every other domain. We attribute this isolation to pretraining, since base models such as `Qwen3` are already saturated with math and code. TAC samples both domains less than uniform sampling does, yet it still improves over Random on 7 of 8 math and codegen benchmarks across the two backbones (Table 1).

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). We set $\beta = 0.2$ once before running any sweep and keep it unchanged across all backbones and data mixtures. 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 that learnability still adds signal. α peaks at 0.3

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**

(Figure 4c). Smaller values update too slowly for the fast-changing gradient geometry of RL, and larger values amplify step-level noise. N has the smallest effect (Figure 4b), indicating that a few trailing layers already carry most of the cross-domain alignment signal. Additional ablations on the projection family, projection dimension r, and the data-exhaustion strategy are in Appendix D.2.

Additional Study: Imbalanced Data Budget. Training pools are rarely balanced in practice. We therefore re-run training on the imbalanced data budget (§4.1) and evaluate `Qwen3-1.7B` on the same 14 benchmarks (Table 2). `TAC` again leads on macro-averaged accuracy (32.7, $+1.9$ over Random and $+1.5$ over SEC) and ranks first on 11 of 14 benchmarks. The Q-initialization gives math and stem about $1.9\times$ the initial mass of simulation and table, and a learnability-only signal cannot correct this once those large domains saturate. `TAC`'s transferability term releases weight from them, and the largest gains go to under-represented domains (CodeI/O $+2.1$ over Random, FinQA $+4.0$ over SEC). `TAC`'s gains therefore hold under data-budget skew and are not an artifact of the balanced setting.

5 Related Work

Multi-Domain Reasoning for LLMs. Multi-domain training builds on classical multi-task learning, including shared representations [5, 47], gradient-conflict mitigation [61, 33], and data-mixture optimization for language model pretraining [58]. Recent work brings these questions to RL training of LLM reasoners. General Reasoner [36] and GURU [11] build broad multi-domain reasoning suites, including domains with judge-based rewards [64]. Both show that math-only RL is insufficient for general-purpose reasoning. Huan et al. [20] and Li et al. [30] study when reasoning skills transfer across domains. These studies measure transfer after training, while `TAC` estimates it online and uses it to choose the next training domain. Liang et al. [31] encourage gradient alignment across domains during RL, and Ramesh et al. [45] reweight per-task losses for balanced multi-task GRPO. Both modify the optimizer or the loss. `TAC` instead targets the sampling distribution over domains.

Curriculum Learning for RL. Curriculum learning has a long history in RL [38, 53, 54, 52]. Most methods estimate task difficulty or learning progress and then select tasks, goals, or environments to improve sample efficiency. Recent work applies curricula to RL training of LLM reasoners. Pang et al. [40] train on math first as a foundation for other tasks, and easy-to-hard curricula order problems by difficulty [42]. DUMP [55] and SEC [8] build curricula automatically from advantage-based learnability, and VCRL [22] uses reward variance in a similar way. These signals score each domain in isolation, so they cannot tell whether an update on one domain helps the others. Omni-Thinker [26] uses cross-domain information but measures accuracy transfer on held-out probes. `TAC` shares the bandit framework and the learnability term of these methods. Its addition is a transferability signal computed directly from training gradients, which needs no extra rollouts or held-out probes.

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` outperforms proportional sampling, a hand-designed math-to-others curriculum, and a learnability-only bandit on both `Qwen3-1.7B` and `Llama3.2-3B`, and it keeps its lead under an imbalanced training mixture. We hope these results motivate further work on cross-domain transferability as a first-class signal for curriculum design in multi-domain reasoning. We discuss limitations and broader impact in Appendix A.

Acknowledgments and Disclosure of Funding

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 Limitations and Broader Impact

Limitations. We note three limitations of the curriculum design. First, the gradient cosine is a first-order proxy for cross-domain transfer. Our support for it is empirical, through the β ablation and its agreement with the offline transfer matrix (§4.2), and we do not characterize when its relative ordering stops being informative. The cosine is also symmetric in the two domains, while measured transfer need not be (Appendix C.1). Second, `TAC` scores a domain by its mean alignment with all other domains. A domain whose updates help a few domains strongly but most only weakly can therefore be under-sampled. Conversely, a domain that is broadly but shallowly aligned can be over-credited. Aggregations richer than the mean are a natural next step. Third, our evidence is limited to RL with verifiable rewards on six domains and models up to 4B parameters. Model-judged rewards [64] and broader domain mixtures remain untested. The transferability term reads gradients rather than reward statistics, so we expect it to degrade more gracefully under noisy rewards than advantage- or variance-based criteria, but we have not tested this.

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.

B Experiment Prompts

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

B.1 Math

User Prompt (Math)

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

B.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).

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

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:

{{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```.
```

B.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>.
```

B.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.
```

B.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{}.
```

B.6 STEM

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

C Implementation Details

C.1 Method Details

This subsection collects the method details omitted from Section 3.

Learnability normalization. During the normalizer's own warmup (12 steps, Table 6), before the running statistics in Eq. (7) are reliable, we feed the raw $L_{m_t}^{(t)}$ to the curriculum instead of the z-score.

Gradient sketch. The ℓ_2 normalization in Eq. (8) makes the per-domain state 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 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. In practice we project the gradient of the final PPO mini-batch of each step, and the EMA smoothing absorbs the additional variance. The first observation for a domain initializes its gradient state $\mathbf{h}_m$ directly.

Transferability normalization. The raw transfer $\rho_m^{(t)}$ changes at every comparison even for domains that were not sampled in between, because the *other* domains' gradient states have moved. The scale $s^{(0)}$ in Eq. (12) is initialized to the first observed range. The floored EMA scale guards against a specific failure mode. The cross-domain spread occasionally collapses when all domains' smoothed cosines briefly cluster. 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.

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

Warmup. The round-robin order is shuffled in each warmup round. 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. It also ensures 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. After warmup, the UCB bonus in Eq. (5) carries the ongoing exploration burden, occasionally reviving arms whose Q-values dipped early so that their cached learnability is refreshed.

Symmetry of the transfer estimate. The cosine in Eq. (10) is symmetric in the two domains, while measured transfer is not. In Figure 1a, training on *table* raises *codegen* accuracy by 19.5 points, but training on codegen changes table accuracy by -0.1. Part of this gap reflects target headroom, since every source domain improves codegen by at least 8 points. The first-order predictor in Eq. (15) is symmetric only for exact gradients at a single parameter point. In training, each update uses a gradient estimated from a small minibatch, and transfer accumulates over many steps along different trajectories. Target-specific curvature and optimizer dynamics then make the realized transfer directional. These directional effects are hard to estimate reliably from a few minibatch gradients, so we use the symmetric cosine as a stable ordering signal rather than as a per-pair prediction.

C.2 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]).

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

- *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-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). Our pipeline also 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 in Table 1 come from each run's *final* checkpoint (Appendix C.4). This validation set partially overlaps with the held-out evaluation benchmarks reported in Table 1; because no model-selection or tuning

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

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

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 $8\times$ 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

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, $8\times$ repetition), AMC (332 samples, $4\times$ 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.

C.3 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.

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

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

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

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

Table 8: **Additional curriculum baselines.** Macro-averaged accuracy (%) over the six domains, mean over three training seeds. Rows for Random, M2O, SEC, and TAC are from Table 1.

Method	Selection signal	Qwen3-1.7B	Llama3.2-3B
Random	uniform	31.8	29.2
M2O [40]	fixed schedule	32.0	29.7
SEC [8]	advantage	32.1	28.5
VCRL [22]	reward variance	31.7	28.2
DUMP [55]	advantage with UCB	31.8	28.0
Val-Gain (TSCL-style [38])	held-out probe improvement	32.0	29.1
Omni-Thinker [26]	held-out accuracy transfer	32.1	29.3
TAC	gradient alignment	**33.9**	**31.3**

relative to the other curricula (Figure 6), so the same compute budget is used for every method to ensure a fair comparison.

C.4 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.

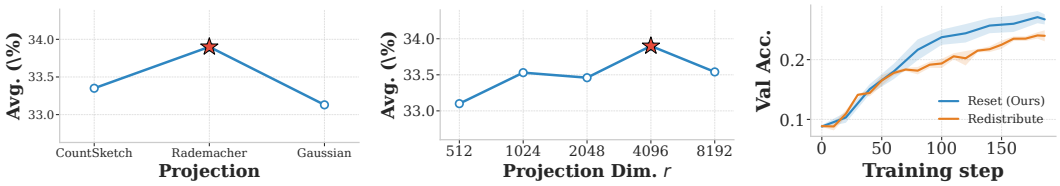
Generation settings. The vLLM engine is initialized once per checkpoint with the maximum prompt and response lengths across all benchmarks, so a single engine serves every benchmark without reloading weights. Sampling uses temperature 1.0 and top-p 1.0, matching the training-time sampling configuration.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

D More Experiments

D.1 Additional Curriculum Baselines

Table 8 compares TAC with four further adaptive curricula under the same data and compute budgets. VCRL [22] ranks domains by reward variance, and DUMP [55] applies a UCB rule to advantage-based learnability. Val-Gain is a TSCL-style bandit [38] whose arm signal is the accuracy improvement on a small held-out probe per domain (20 items, evaluated every 2 steps). Omni-Thinker [26] measures accuracy transfer on held-out probes at initialization and then fixes its schedule. No additional baseline improves over Random by more than 0.3 points on either backbone. The curricula driven by reward statistics (SEC, VCRL, DUMP) rank domains by nearly the same quantity. For binary rewards, the reward variance is $\hat{p}(1-\hat{p})$ and the mean absolute advantage is $2\sqrt{\hat{p}(1-\hat{p})}$, and neither carries information about cross-domain effects. The two methods that consult held-out accuracy see cross-domain outcomes, but Omni-Thinker cannot adapt once its schedule is fixed and Val-Gain relies on small probes whose noise exceeds the per-step changes it needs to detect. TAC improves over the strongest additional baseline (Omni-Thinker) by $+1.8$ points on Qwen3-1.7B and $+2.0$ points on Llama3.2-3B.



(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.

D.2 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 over the six domains, as in Table 1; 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.

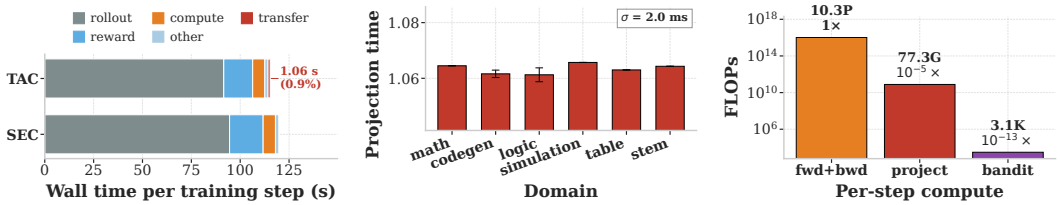
Data-exhaustion strategy. When a domain's index pool is exhausted within an epoch, the curriculum must decide what to do at the next time the bandit selects that domain. We compare two strategies: *reset* (TAC's default), which reshuffles the depleted domain's data and continues sampling from it; and *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.

D.3 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× 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



(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 $1.06\,\mathrm{s}$ on top of a $115\,\mathrm{s}$ step (0.9%); SEC has no *transfer* slice. **(b)** Projection cost is essentially constant across the six training domains ($\sigma = 2\,\mathrm{ms}$ on a $1.06\,\mathrm{s}$ 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.

on a $1.06\,\mathrm{s}$ mean (0.2%), confirming that the projection cost depends only on parameter count and projection dimension, and 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 ($\approx$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 $\sim 3.1\times10^{11}$ at our default $r=4096$, and the bandit Q-update is $\mathcal{O}(M)$ scalar operations across $M=6$ domains ($\sim 3\times10^3$ 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.

D.4 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 necessary. Raw cosines alone would be uninformative as a curriculum signal, but the *relative* ordering across domains is stable 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. This explains 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 the rest of the mixture and often oppose it. This 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.

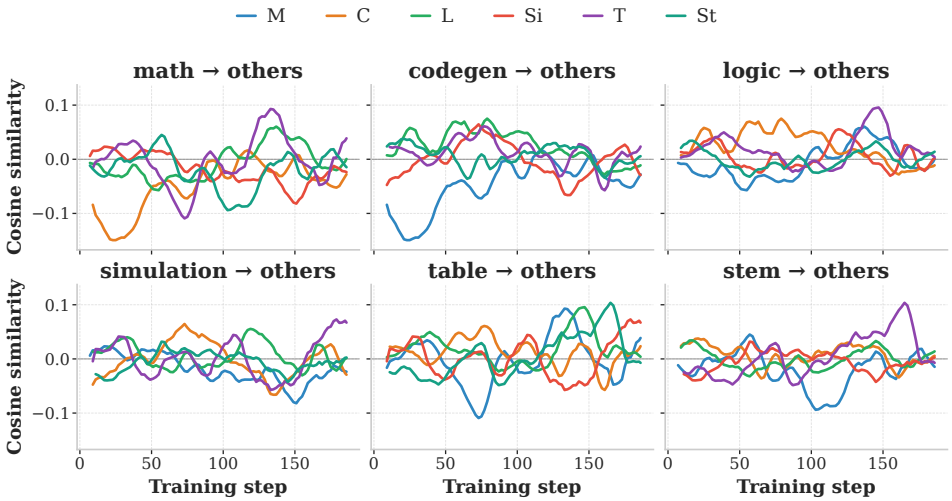


Figure 7: **Pairwise gradient cosine similarity over training, per source domain.** Each panel plots the cosine similarity between the projected-gradient EMA $\mathbf{h}_m^{(t)}$ of one *source* domain m (panel title) and the EMAs of the five other *target* domains (line color), averaged across three 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 ($|\cos| \lesssim 0.15$), so domain pairs are only weakly aligned in the projected subspace. The *relative* structure is still stable. *Table* acts as a hub, ending positively aligned with *simulation*, *stem* (its two largest cosines), and *logic*, which is why it carries the highest aggregate T in Figure 3b. *Math* and *codegen* sit at negative cosine with nearly every domain and most negatively with each other, which explains their lowest aggregate T.

Table 9: 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**

D.5 Scaling Experiments

Table 9 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 with scale, T adapts with it.

E 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:

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

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.

$$\dots$$

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:

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

The lab-frame decay length is:

$$d = \gamma \, \tau_0 \, c \approx 7.92 \cdot 8 \times 10^{-16} \, \text{s} \cdot 3 \times 10^{8} \, \text{m/s}$$
$$d \approx 1.9 \times 10^{-6} \, \text{m}$$
$$\dots$$

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).

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The abstract and introduction (§1) state TAC's three contributions and the empirical claims (1.6–2.8 point macro improvement) that are quantitatively established in §4.2.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: Appendix A discusses three limitations: the gradient cosine is a first-order proxy for transfer, mean aggregation over domain pairs can mis-credit domains, and the evidence is limited to RLVR on six domains with models up to 4B parameters.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [N/A]

Justification: The paper does not include formal theoretical results. The first-order Taylor argument in §3.3 (Interpretation) is a brief justification, not a theorem.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: [Yes]

Justification: §4.1 specifies all training details (datasets, models, baselines, hyperparameters); Algorithm 1 gives the full method; Appendix C.1–C.3 provides method details, per-domain sources, subsampling protocol, and full hyperparameter tables.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in

some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We release the `TAC` implementation, training scripts, and configuration files used in our experiments at `https://github.com/YangYongJin/transfer-aware-curriculum`. All training data sources are public and listed in Appendix C.2.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: §4.1 covers data splits, models, and baselines; Appendix C.3 (Tables 6, 7) provides full method-side and RL-side hyperparameters.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: Table 1 reports mean$\pm$std over three training seeds, with most benchmarks additionally averaged over 4 evaluation runs (Appendix C.2, Table 5). Figures 3 and 4 show shaded $\pm 1\sigma$ regions over seeds.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.

- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: Appendix C.3 (Compute paragraph) reports a single 4×H100 80 GB node, ∼7 h wall-clock per run, and the total budget shared across all baselines.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: The work uses publicly released datasets and pretrained models under their licenses; no human subjects, no scraped data, no deployment.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

Justification: Appendix A (Broader Impact) discusses positive impacts (broader access to multi-domain reasoning training, lower compute barrier) and the absence of new deployment risks beyond those generic to RL fine-tuning of LLMs.

Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

Answer: [N/A]

Justification: The paper releases no new pretrained models, datasets, or other dual-use assets; `TAC` is a curriculum-learning method applied to existing public resources.

Guidelines:

- The answer [N/A] means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [Yes]

Justification: All datasets and models are cited in §4.1 and Appendix C.2 with their original sources. The verl framework, GURU suite, and all benchmark datasets are used under their public licenses.

Guidelines:

- The answer [N/A] means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The only new asset is the `TAC` code, released with the training scripts and configuration files for all experiments at `https://github.com/YangYongJin/transfer-aware-curriculum`. The paper releases no new datasets or models.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: The paper does not involve crowdsourcing or human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [N/A]

Justification: The paper does not involve human subjects.

Guidelines:

- The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

Answer: [N/A]

Justification: LLMs were used only for grammar review of the manuscript and do not affect the core methodology, scientific rigor, or originality of the research.

Guidelines:

- The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.