# NOT ALL TOKENS SHOULD UPDATE THE SAME PARAMETERS: PREFIX-CONDITIONED GRADIENT ROUTING FOR ON-POLICY DISTILLATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

On-policy distillation (OPD) trains a student on its own rollouts with token-level feedback from a stronger teacher. Because this feedback is issued on student-generated prefixes, it can also push the student away from behavior it already performs correctly, causing regressions on problems it previously solved. Existing remedies reweight tokens, but a scalar weight shrinks every parameter component of a token gradient together, so suppressing the part of an update that conflicts with previously successful behavior also discards the part that agrees with it. We propose **Prefix-Conditioned Gradient Routing (PCGR)**, which makes this choice per parameter block. Using the gradient of the student's own verified past successes as a reference, PCGR lets the accumulated teacher–student discrepancy of the prefix decide how much reference conflict an update may keep, and block-wise reference geometry decide where the rest is removed. The resulting convex problem has a closed-form gate rule that, at equal retained reference conflict, never distorts the OPD gradient more than the corresponding group-level scalar gate. In standard OPD, 71.2% of token gradients mix reference-compatible and reference-conflicting blocks. Across four verifiable task families with Qwen3-1.7B and Qwen3-4B students, PCGR improves the macro score over OPD by 2.5 and 2.0 points and nearly halves the regression rate on previously solved problems (13.9% to 7.7%) without lowering the rate of new solutions.

## 1 INTRODUCTION

On-policy distillation (OPD) trains a student language model on its own sampled responses, with a stronger teacher providing token-level feedback on every generated prefix (Agarwal et al., 2024; Gu et al., 2024). Unlike offline distillation on teacher-written text (Hinton et al., 2015; Kim & Rush, 2016), OPD supervises the states the student actually visits at inference time, which lets the teacher correct the student's own mistakes, and it has become a standard component of LLM post-training (Yang et al., 2025; Lu & Lab, 2025).

**OPD can erase competence the student already has.** The teacher's feedback on a student prefix mixes genuine corrections with signals that push the student away from behavior that already works—for example, a solution strategy that the student executes reliably but that the teacher would not have chosen. Because every token update moves all parameters, such signals can overwrite existing competence, and they become more frequent as teacher–student discrepancy accumulates along the prefix. The result is *regression*: problems the student solved before training become unsolved. Standard OPD loses 8.0% of the mathematics problems that the initial Qwen3-1.7B student solved, even while raising its mathematics score by 3.1 points. This is a within-task form of the stability–plasticity dilemma of continual learning (Kirkpatrick et al., 2017; Lopez-Paz & Ranzato, 2017).

**Token reweighting cannot fully fix it.** A natural remedy is to down-weight tokens whose prefixes have drifted farthest from the teacher (Xie et al., 2026). A token weight, however, multiplies the entire token gradient by one scalar. Consider a gradient spanning two parameter blocks,



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

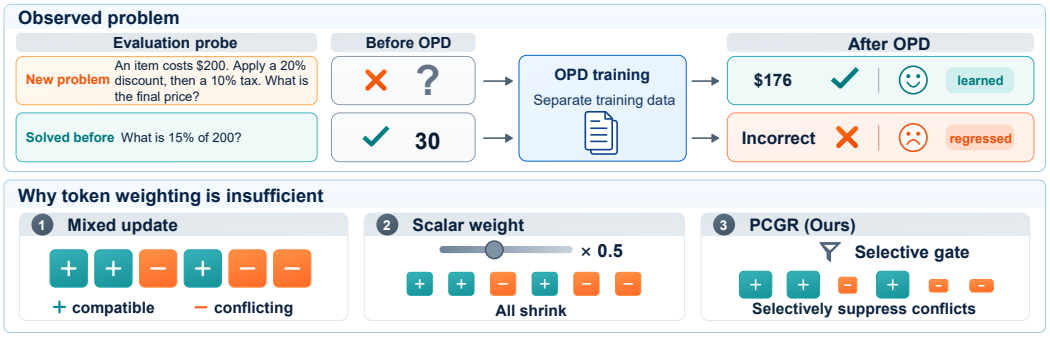


Figure 1: **Motivation.** Top: OPD solves new problems but can regress on problems the student already solved. Bottom: a scalar token weight shrinks compatible and conflicting parameter components alike; PCGR attenuates only the conflicting ones.

where the first block's component improves a trusted reference objective by one unit and the second worsens it by one unit. Any weight $w \in [0, 1]$ keeps $w$ units of conflict together with $w$ units of compatible progress, and eliminating the conflict ($w = 0$) also eliminates the progress; the blockwise gate $(1, 0)$ removes the conflict and keeps the progress. Whenever reference-compatible and reference-conflicting components coexist within one update, scalar weighting thus trades stability against learning (Figure 1). This is the common case: in standard OPD, 71.2% of token gradients contain both compatible and conflicting parameter blocks (Section 5.1).

**Our approach.** We propose **Prefix-Conditioned Gradient Routing (PCGR)**, which replaces the single scalar knob with one knob per parameter block (Figure 2). PCGR separates two decisions that token weighting makes jointly: *prefix discrepancy decides how much conflict an update may keep, and reference geometry decides where the rest is removed.* The reference is the gradient of the student's likelihood on its own earlier responses that passed an outcome verifier, a local proxy for competence that should not regress; it routes the OPD gradient but is never added to it. PCGR groups tokens by the accumulated discrepancy of their prefixes and, for each group, finds the blockwise gates closest to the identity that retain only a budgeted fraction of the conflict with the reference, with a budget that tightens as discrepancy grows. High-discrepancy tokens are therefore not discarded: their compatible components pass through unchanged, and only their conflicting components are constrained more tightly. Because PCGR only rescales blocks of the OPD gradient, it adds no loss term, leaves inference unchanged, and composes with token weighting: using IW-OPD weights inside PCGR gives the best results wherever both are evaluated.

**Results.** The routing problem is convex and has a closed-form soft-gate solution in one dual variable. At the same retained conflict as the best group-level scalar gate, the gates never distort the OPD gradient more, keep all the compatible reference progress that scalar gating discards, and can reduce distortion to a fraction $\kappa \in (0, 1]$ of the scalar cost; in standard OPD, the median of this conflict-concentration coefficient is 0.32. In training, across four verifiable task families and two student scales, PCGR improves the macro score over OPD by 2.5 points and over the strongest token-weighting and reference-aware baselines by 1.87 and 1.21 points, lowers the family-average regression rate from 13.9% to 7.7% while keeping the gain rate of OPD, and stays ahead at equal GPU-hours. The advantage persists at 4B, under mixed-task training, and under AdamW.

Our contributions are threefold. **(i) Problem:** we identify within-update blockwise conflict as a source of regression in OPD that token weighting cannot remove without discarding reference-compatible teacher signal, and give a measurable criterion, the conflict-concentration coefficient, for when parameter-level allocation helps. **(ii) Method and analysis:** we introduce PCGR, a prefix-conditioned blockwise projection with a closed-form gate rule, prove that it dominates the corresponding group-level scalar gate at matched reference conflict, and show that it improves first-order reference alignment over OPD by exactly the conflict it removes. **(iii) Evidence:** we show that blockwise conflict is pervasive in OPD and grows with prefix discrepancy, and that PCGR outperforms token-space, parameter-space, and reference-aware baselines at two scales, reduces regressions without slowing gains, and keeps its advantage under mixed-task training, AdamW, and equal compute.

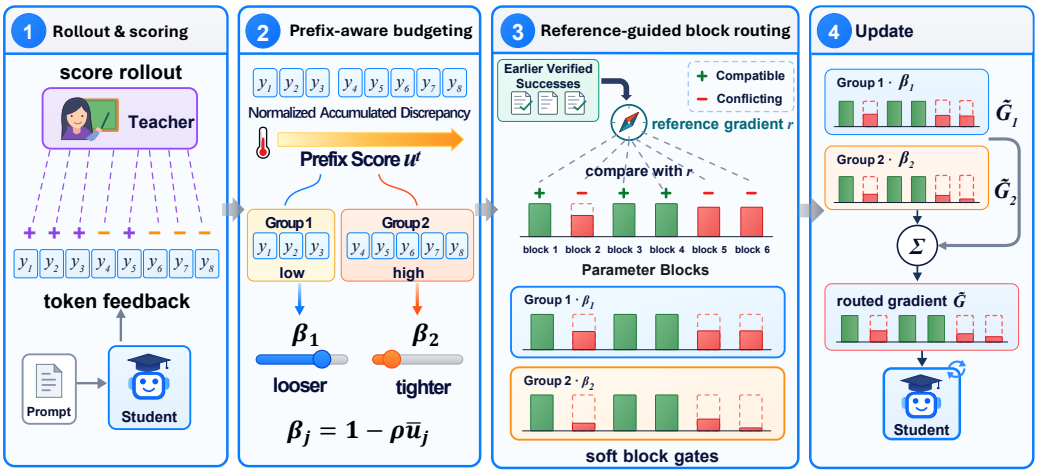


Figure 2: **Overview of PCGR.** (1) The teacher scores student rollouts token by token. (2) Tokens are grouped by prefix discrepancy $u_{i,t}$, and each group $j$ receives a conflict budget $\beta_j = 1 - \rho\bar{u}_j$ that tightens with discrepancy. (3) Each group gradient is compared blockwise with the gradient $r$ of earlier verified successes, and soft gates attenuate only conflicting blocks until the budget is met. (4) The routed group gradients are summed and applied.

## 2 Related Work

**On-policy distillation.** Knowledge distillation trains a student to match a teacher (Hinton et al., 2015; Kim & Rush, 2016). On-policy variants supervise the prefixes the student generates itself, as imitation learning queries the expert on learner-visited states (Ross et al., 2011): GKD uses general divergences on student samples (Agarwal et al., 2024), MiniLLM optimizes reverse KL with policy-gradient techniques (Gu et al., 2024), and DistiLLM combines a skewed divergence with adaptive use of student outputs (Ko et al., 2024). These works shape the objective and the sampling distribution; PCGR shapes how the resulting gradient is allocated across parameters.

**Token-level allocation.** Selective language modeling trains only on tokens with high excess loss (Lin et al., 2024), and IW-OPD weights OPD tokens by prefix discrepancy and position (Xie et al., 2026). These methods choose one scalar per token, whereas PCGR chooses one gate per token group and parameter block; the two combine naturally, so we use IW-OPD both as a baseline and inside PCGR.

**Gradient conflict and forgetting.** Gradient surgery resolves conflicts between task gradients by projection or reweighting (Yu et al., 2020; Wang et al., 2021; Liu et al., 2021), and continual-learning methods constrain or project updates using an episodic memory (Lopez-Paz & Ranzato, 2017; Chaudhry et al., 2019; Farajtabar et al., 2019). Forgetting also arises in LLM post-training (Luo et al., 2025), although on-policy training forgets less than supervised fine-tuning (Shenfeld et al., 2025). The regressions we target arise within a single task, and our memory is the student's own verified outputs collected online. Unlike reference-gradient methods, PCGR conditions the constraint on the prefix, enforces an adjustable budget instead of eliminating conflict, and rescales OPD blocks instead of adding reference directions. Appendix A discusses further related work.

## 3 Method

PCGR changes only how the OPD gradient is applied; the loss, the rollouts, and the teacher are unchanged. Figure 2 summarizes one training step.

### 3.1 On-policy gradients and prefix discrepancy

At iteration $k$, the current student $\pi_{\text{old}} = \pi_{\theta^k}$ samples $y_i \sim \pi_{\text{old}}(\cdot \mid x_i)$ for training prompts $x_i$, and $h_{i,t} = (x_i, y_{i,<t})$ denotes the prefix before token $y_{i,t}$. With a fixed teacher $\pi_T$ that shares the

student's tokenizer, sampled token-local reverse-KL training uses

$$A_{i,t} = \log \frac{\pi_T(y_{i,t} \mid h_{i,t})}{\pi_{\text{old}}(y_{i,t} \mid h_{i,t})}, \qquad \ell_{i,t}(\theta) = -\,\text{sg}[A_{i,t}] \log \pi_\theta(y_{i,t} \mid h_{i,t}), \qquad g_{i,t} = \nabla_\theta \ell_{i,t}(\theta^k), \tag{1}$$

where the stop-gradient $\text{sg}[\cdot]$ treats the sampled coefficient as fixed (Xie et al., 2026). We take one optimization step per freshly generated batch. To measure how far a prefix has drifted from the teacher, we use the accumulated discrepancy

$$D_{i,t} = \sum_{s<t} |A_{i,s}|, \qquad u_{i,t} = \frac{D_{i,t}}{D_{i,t} + \tau_d} \in [0, 1), \tag{2}$$

where the scale $\tau_d$ of task $d$ is fixed before training to the median positive $D_{i,t}$ on initial-student rollouts (Appendix C). We use $u_{i,t}$ as a routing feature rather than a correctness estimate: it tells PCGR how cautious to be, while a separate signal defines what to protect.

### 3.2 Trusted reference

Each task keeps a FIFO buffer of at most $C$ earlier student responses that passed the task's outcome verifier, with at most one response per prompt. At step $k$, we sample up to $M$ of them into $\mathcal{A}_k$, excluding prompts in the current batch, and define

$$R_k(\theta) = -\frac{1}{|\mathcal{A}_k|} \sum_{(x,z)\in\mathcal{A}_k} \frac{1}{|z|} \sum_s \log \pi_\theta(z_s \mid x, z_{<s}), \qquad r = \nabla_\theta R_k(\theta^k). \tag{3}$$

The reference gradient $r$ only guides routing and is never added to the update. A verifier certifies only the final outcome, so $r$ is a proxy for competence rather than a certificate of every step; it is stable across independent reference batches (Section 5.1), and replacing it by an unverified reference clearly hurts (Section 5.4). Current successes are admitted only after the update, and if no eligible reference exists, routing is bypassed.

### 3.3 Grouped gradients

We partition the parameters into $B$ blocks, one per parameter tensor by default ($B = 310$ for Qwen3-1.7B and 398 for Qwen3-4B). To avoid per-token parameter gradients, we split response tokens into $K$ fixed-width bins $\mathcal{G}_1, \ldots, \mathcal{G}_K$ of $u_{i,t}$ ($K = 2$ by default) and define, for each group $j$,

$$G_j = \frac{1}{N} \sum_{(i,t)\in\mathcal{G}_j} w_{i,t}\, g_{i,t}, \qquad N = \sum_i T_i, \qquad \bar{u}_j = \frac{\sum_{(i,t)\in\mathcal{G}_j} w_{i,t} u_{i,t}}{\sum_{(i,t)\in\mathcal{G}_j} w_{i,t}}, \tag{4}$$

where $T_i = |y_i|$. The base method uses $w_{i,t} = 1$; PCGR+IW uses detached IW-OPD token weights (Xie et al., 2026). The common denominator $N$ preserves the weighting of the batch objective. Routing thus acts at the group–block granularity, where token contributions within a group can partially cancel; Appendix F varies $K$ and the block granularity.

### 3.4 Blockwise routing under a conflict budget

Let $P \succ 0$ be the diagonal preconditioner of the current optimization step. For group $j$ and block $b$, define

$$E_{j,b} = G_{j,b}^\top P_b G_{j,b}, \qquad a_{j,b} = r_b^\top P_b G_{j,b}, \qquad H_{j,b} = [-a_{j,b}]_+. \tag{5}$$

An update $-\eta P_b G_{j,b}$ changes the reference loss by $-\eta a_{j,b}$ to first order, so $a_{j,b} > 0$ marks a block that is compatible with the reference, $H_{j,b}$ measures the conflict of a block, and $E_{j,b}$ is its energy in the $P$ metric.

**How much conflict to keep.** Each group receives a conflict budget that tightens with its mean discrepancy,

$$\beta_j = 1 - \rho \bar{u}_j, \qquad \rho \in [0, 1]. \tag{6}$$

We treat prefix discrepancy as an empirical indicator of reference-conflict risk: on a high-discrepancy prefix, teacher and student already disagree on much of the trajectory, and the conflict

fraction of OPD updates rises with $u_{i,t}$ beyond what token position explains (Section 5.1). The budget restricts only conflicting mass: compatible components of high-discrepancy tokens pass unchanged, so corrections that agree with verified behavior are kept. Any nonincreasing schedule can replace Eq. (6); Section 5.4 compares it with constant and position-based ones.

**Where to remove it.** Given $\beta_j$, PCGR finds the gates closest to the identity that meet the budget:

$$\begin{aligned} \min_{m_j \in [0,1]^B} \quad & \frac{1}{2} \sum_b E_{j,b}(1 - m_{j,b})^2 \\ \text{subject to} \quad & \sum_b m_{j,b} H_{j,b} \le \beta_j \sum_b H_{j,b}. \end{aligned} \tag{7}$$

Since the objective equals $\frac{1}{2}\|m_j \odot G_j - G_j\|_P^2$, Eq. (7) projects $G_j$, in the optimizer metric, onto the budget-feasible gatings. Compatible blocks consume no budget, so the problem allocates the allowed conflict across blocks instead of trading conflicting blocks against compatible ones. For $E_{j,b} > 0$, the KKT conditions give the soft gates

$$m^\star_{j,b} = \left[1 - \lambda_j \frac{H_{j,b}}{E_{j,b}}\right]_{[0,1]}, \tag{8}$$

where the scalar dual $\lambda_j \ge 0$ is found by bisection so that the budget is met (Lemma 1, Appendix B); all gates equal one if $\sum_b H_{j,b} = 0$ or $\beta_j = 1$, and zero-energy blocks receive gate one. The rule is easy to read: blocks are attenuated in order of their conflict per unit of OPD energy, $H_{j,b}/E_{j,b}$, so a block that adds much conflict but little teacher signal is cut first, and compatible blocks ($H_{j,b} = 0$) are never touched. The detached gates are then applied as

$$\widetilde{G}_b = \sum_j m^\star_{j,b} G_{j,b}, \qquad \theta^{k+1} = \theta^k - \eta_k P \widetilde{G}. \tag{9}$$

Setting $\rho = 0$ recovers the underlying OPD update.

### 3.5 What blockwise routing adds over scalar gating

We analyze one group and drop the index $j$. Let $\mathcal{S} = \{b : E_b > 0\}$, $H = \sum_b H_b$, $Q = \sum_b [a_b]_+$, and $\mathrm{Dist}(m) = \frac{1}{2}\sum_b E_b(1 - m_b)^2$. A scalar gate $m = \alpha \mathbf{1}$, which is what any group-level token weight amounts to, is feasible in Eq. (7) if and only if $\alpha \le \beta$, so $m^{\mathrm{sc}} = \beta \mathbf{1}$ is the best feasible scalar gate; per-token weights such as IW-OPD act at a finer granularity and are compared empirically in Section 5. We define the *conflict-concentration coefficient*

$$\kappa = \frac{\left(\sum_{b \in \mathcal{S}} H_b\right)^2}{\left(\sum_{b \in \mathcal{S}} E_b\right)\left(\sum_{b \in \mathcal{S}} H_b^2 / E_b\right)} \in (0, 1]. \tag{10}$$

By the Cauchy–Schwarz inequality, $\kappa = 1$ if and only if $H_b/E_b$ is constant on $\mathcal{S}$, that is, when conflict is spread across blocks in proportion to gradient energy; $\kappa$ is small when conflict is concentrated in a few blocks, and every block that carries energy but no conflict lowers it. Proofs are in Appendix B.

**Proposition 1** (Blockwise routing versus scalar gating)**.** *Suppose $H > 0$ and $0 < \beta < 1$. Then:*

*(i) Equal conflict: $\sum_b m^\star_b H_b = \sum_b m^{\mathrm{sc}}_b H_b = \beta H$.*

*(ii) Lower distortion: $\kappa\, \mathrm{Dist}(m^{\mathrm{sc}}) \le \mathrm{Dist}(m^\star) \le \mathrm{Dist}(m^{\mathrm{sc}}) = \frac{1}{2}(1 - \beta)^2 \sum_b E_b$. The first inequality holds with equality whenever no gate saturates at zero, and the second is strict if and only if $\kappa < 1$.*

*(iii) Retained compatible progress: $r^\top P(m^\star \odot G) = Q - \beta H = r^\top P(m^{\mathrm{sc}} \odot G) + (1 - \beta)Q$.*

At matched conflict, PCGR is exactly as conservative as the best scalar gate, but its distortion can be as low as a fraction $\kappa$ of the scalar cost, and it keeps the $(1 - \beta)Q$ units of first-order reference progress that scalar gating discards. In the two-block example of Section 1, $E = (1, 1)$, $H = (0, 1)$, and $Q = 1$, so $\kappa = 1/2$: PCGR halves the distortion for every $\beta$ and keeps the compatible block intact. Because $\kappa$ is computed from statistics PCGR already uses, it tests the premise of the method at no extra cost: if OPD updates had $\kappa \approx 1$, blockwise routing would reduce to token weighting. Section 5.1 shows that they do not. The gates are also invariant to rescaling $r \to cr$ with $c > 0$, so they depend on the direction of the reference, not on its norm.

**Proposition 2** (Local change of the reference objective)**.** *For the aggregated update of Eq.* (9) *and the unrouted OPD gradient* $G = \sum_j G_j$*,*

$$r^\top P\widetilde{G} = \sum_j (Q_j - \beta_j H_j) =: S_k, \qquad r^\top P(\widetilde{G} - G) = \sum_j (1-\beta_j) H_j \geq 0.$$

*If the reference loss* $R_k$ *is* $L$*-smooth along the step and* $S_k > 0$*, every step size* $0 < \eta_k < 2S_k/(L\|P\widetilde{G}\|_2^2)$ *decreases* $R_k$*.*

The second identity is the property PCGR is designed for: with the same reference and preconditioner, routing improves first-order reference alignment over the unrouted OPD step by exactly the conflict it removes.

### 3.6 Optimization and implementation

To match the analysis exactly, our main experiments use the diagonal RMS preconditioner $P^k = \mathrm{diag}((\sqrt{v^k} + \epsilon)^{-1})$ with $v^{k+1} = \nu v^k + (1-\nu)(\widetilde{G} \odot \widetilde{G})$, without momentum or weight decay. For AdamW, PCGR computes routing statistics with the previous second-moment state and supplies the routed gradient to the optimizer (Section 5.5). Each group gradient $G_j$ needs its own backward pass, so a step costs $K$ rollout backward passes plus one reference pass and, with groups processed sequentially, three parameter-sized buffers ($r$, $G_j$, $\widetilde{G}$) regardless of $K$. Because routing is nonlinear in $G_j$, each $G_j$ is fully accumulated over micro-batches and data-parallel ranks before routing. Routing is confined to training, so inference is unchanged. Algorithm 1 in Appendix C gives the full procedure.

## 4 Experimental Setup

**Models and tasks.** Qwen3-1.7B (main) and Qwen3-4B students are distilled from a frozen Qwen3-30B-A3B teacher (Yang et al., 2025), starting from the released post-trained checkpoints in thinking mode with full-parameter updates. The core suite has four verifiable task families, each with its own outcome verifier: mathematics (MATH-500, AIME24/25), code (HumanEval+, MBPP+), instruction following (IFEval, IFBench out-of-distribution), and single-turn tool calling (BFCL) (He et al., 2025; Cui et al., 2025; Liu et al., 2023; Pyatkin et al., 2026; Liu et al., 2024; Patil et al., 2025). Appendix J lists the training sources, splits, and decontamination, and Appendix G reports an extended suite with SQL and procedural reasoning.

**Training.** Each run performs 400 updates of 64 prompts with one response per prompt, sampled at temperature one with a 16,384-token response cap. PCGR uses $K = 2$, $\rho = 0.5$, a buffer of $C = 256$ verified responses per task, and at most $M = 8$ references per step. All main runs use the RMS optimizer of Section 3.6, so the analysis of Section 3.5 applies exactly; Section 5.5 repeats the key comparison with AdamW. Every method receives the same tuning budget, including a per-method learning-rate search (Appendix J). Single-task training is the main comparison; mixed-task runs sample the four families uniformly with task-specific buffers (1,600 updates). Main, scale, mixed-task, and ablation runs use three seeds.

**Baselines.** All reference-aware methods share PCGR's verifier access, buffer, and preconditioner and differ only in how $r$ is used (details in Appendix D).

- *Token space.* **OPD**; **OPD (norm-matched)**, which rescales each OPD step to the norm of the PCGR step on the same batch and tests whether PCGR is merely a smaller step; and **IW-OPD** (Xie et al., 2026).
- *Parameter space.* **CPI-FT** and **EPI** (Wang et al., 2025; Lin et al., 2026), importance-mask methods adapted to OPD that concentrate the update on, or shield, high-importance parameters regardless of the reference.
- *Anchor.* **OPD+KL(init)** adds a KL penalty to the frozen initial student and tests whether limiting drift suffices.
- *Reference aware.* **OPD+**$\alpha R$ adds $\alpha r$ to the update. **Uniform group scaling (UGS)**, the scalar counterpart of PCGR, sets $m_{j,b} = \beta_j$ with identical budgets. **A-GEM-style projec-**

Table 1: Blockwise conflict in standard OPD (Qwen3-1.7B). Mixed: token gradients with both compatible and conflicting blocks. Conflict: share of $P$-energy in conflicting blocks. $\kappa$: median of Eq. (10). Ratio: median $\mathrm{Dist}(m^\star)/\mathrm{Dist}(m^{\mathrm{sc}})$. Zero: gates saturated at zero. Effect of $u$: position-controlled regression coefficient with 95% interval.

| Task | Mixed (%) | Conflict (%) | $\kappa$ (token) | $\kappa$ (group) | Ratio | Zero (%) | Effect of $u$ |
|---|---|---|---|---|---|---|---|
| Mathematics | 72.4 | 42.0 | 0.31 | 0.45 | 0.58 | 9.3 | 0.20 [0.13, 0.27] |
| Code | 66.8 | 48.0 | 0.37 | 0.53 | 0.63 | 10.6 | 0.16 [0.08, 0.24] |
| Instructions | 75.9 | 38.5 | 0.28 | 0.41 | 0.55 | 7.8 | 0.23 [0.17, 0.29] |
| Tool calling | 69.5 | 45.0 | 0.35 | 0.48 | 0.61 | 10.2 | 0.18 [0.11, 0.25] |

**tion** (Chaudhry et al., 2019) projects each group gradient with $r^\top PG_j < 0$ off $r$, and **blockwise projection** does so per block with $a_{j,b} < 0$, in the spirit of PCGrad and GradVac (Yu et al., 2020; Wang et al., 2021).

**Evaluation.** We sample at temperature 0.6, top-$p = 0.95$, and top-$k = 20$ with a 16,384-token cap, using 32 samples per AIME problem and 8 elsewhere, and report single-sample success. The macro score averages benchmarks within each family and then across families. We report the mean and standard deviation over seeds, with paired cluster-bootstrap intervals over problems. To measure regressions directly, let $p_q^{(0)}$ and $p_q$ be the per-problem success of the initial and the trained student, $\mathcal{J}_{\mathrm{old}} = \{q : p_q^{(0)} \geq \frac{1}{2}\}$ the initially solved problems, and $\mathcal{J}_{\mathrm{new}}$ its complement. The regression and gain rates are

$$\mathrm{RR} = \frac{|\{q \in \mathcal{J}_{\mathrm{old}} : p_q < \frac{1}{2}\}|}{|\mathcal{J}_{\mathrm{old}}|}, \qquad \mathrm{GR} = \frac{|\{q \in \mathcal{J}_{\mathrm{new}} : p_q \geq \frac{1}{2}\}|}{|\mathcal{J}_{\mathrm{new}}|}. \tag{11}$$

A method that merely becomes more conservative lowers both; PCGR aims to lower RR without lowering GR. Re-evaluating the initial student with fresh sampling seeds gives the noise floor, and the threshold-free changes $\Delta_{\mathrm{old}}$ and $\Delta_{\mathrm{new}}$ are the mean changes $p_q - p_q^{(0)}$ over $\mathcal{J}_{\mathrm{old}}$ and $\mathcal{J}_{\mathrm{new}}$.

## 5 Results

We ask four questions. **RQ1:** do OPD updates contain blockwise conflict that scalar weighting cannot remove, and does it grow with prefix discrepancy? **RQ2:** does PCGR improve over token-space, parameter-space, and reference-aware baselines? **RQ3:** does it reduce regressions without slowing gains? **RQ4:** which components matter, and do the gains survive changes of scale, task mix, optimizer, and compute?

### 5.1 RQ1: Blockwise conflict is pervasive and grows with discrepancy

We first test the premise of PCGR on frozen snapshots of the OPD run. For single tokens and for per-response $K = 2$ groups, we compute the blockwise effects $c_{t,b} = r_b^\top P_b g_{t,b}$ and the statistics of Section 3.5 (protocol in Appendix E). Table 1 shows three findings. *Mixed updates are the norm:* 66.8–75.9% of token gradients have both compatible and conflicting nontrivial blocks, and conflicting blocks carry 38.5–48.0% of the gradient energy. *Scalar gating is far from optimal:* the pooled median $\kappa$ is 0.32 for tokens, 0.46 for per-response groups, and 0.56 for the full-batch groups used in training, and the realized distortion of PCGR is 0.55–0.63 times that of the best scalar gate. *Conflict grows with discrepancy:* in a regression of the conflict fraction on $u_{i,t}$ that controls for position, response length, and difficulty, the coefficient is positive on every family (pooled 0.19, 95% interval [0.15, 0.23]), and the trend holds within every position stratum (Appendix Figure 4).

The reference that defines these statistics is stable: an independent reference batch reproduces the sign of $a_{j,b}$ 86% of the time and ranks the gates with Spearman correlation 0.79. One-step interventions on shared batches confirm the consequence for learning (Appendix Table 3). At the same retained conflict, PCGR makes 23% more reverse-KL progress than UGS; at equal step norm, it keeps 97% of OPD's KL progress while cutting the NLL increase on held-out verified responses by 37%.

Table 2: Main results for Qwen3-1.7B, single-task training (one student per family), mean ± std over three seeds. Worst Δ: minimum change over benchmarks relative to the initial student. Best in **bold**; the top four per column are shaded from dark to light orange.

| | Mathematics | | | Code | | Instructions | | Tool calling | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| Method | MATH-500 | AIME24 | AIME25 | HumanEval+ | MBPP+ | IFEval | IFBench | BFCL-AST | BFCL-Exec | Macro | Worst Δ |
| Initial student | 92.0 | 47.5 | 37.5 | 55.0 | 58.0 | 72.5 | 35.0 | 55.0 | 49.0 | 55.31 | – |
| Teacher (Qwen3-30B-A3B) | 96.4 | 78.9 | 69.3 | 84.6 | 76.8 | 82.1 | 55.3 | 72.3 | 66.7 | 75.11 | – |
| *Token space* | | | | | | | | | | | |
| OPD | 91.2 ± 0.4 | 53.5 ± 1.9 | 41.5 ± 2.1 | 60.0 ± 1.1 | 58.0 ± 0.8 | 74.0 ± 0.6 | 38.5 ± 1.0 | 59.0 ± 0.8 | 50.0 ± 0.9 | 57.95 ± 0.42 | −0.8 ± 0.4 |
| OPD (norm-matched) | 91.4 ± 0.5 | 53.1 ± 2.0 | 42.5 ± 1.8 | 60.0 ± 1.2 | 58.3 ± 0.9 | 74.0 ± 0.5 | 38.8 ± 1.1 | 59.0 ± 0.7 | 50.4 ± 1.0 | 58.15 ± 0.39 | −0.6 ± 0.5 |
| IW-OPD | 91.5 ± 0.4 | 54.0 ± 1.8 | 42.4 ± 2.0 | 60.4 ± 1.0 | 58.7 ± 0.7 | 74.5 ± 0.6 | 39.5 ± 0.9 | 59.5 ± 0.8 | 50.8 ± 0.8 | 58.58 ± 0.37 | −0.5 ± 0.4 |
| *Parameter space* | | | | | | | | | | | |
| CPI-FT | 91.6 ± 0.5 | 53.3 ± 2.1 | 42.5 ± 1.9 | 60.3 ± 1.2 | 58.5 ± 0.9 | 74.3 ± 0.7 | 39.1 ± 1.0 | 59.3 ± 0.9 | 50.6 ± 1.0 | 58.38 ± 0.44 | −0.4 ± 0.5 |
| EPI | 91.6 ± 0.4 | 53.6 ± 1.9 | 42.4 ± 2.2 | 60.4 ± 1.1 | 58.6 ± 0.8 | 74.4 ± 0.6 | 39.2 ± 1.1 | 59.4 ± 0.8 | 50.7 ± 0.9 | 58.47 ± 0.40 | −0.4 ± 0.4 |
| *Anchor* | | | | | | | | | | | |
| OPD+KL(init) | 91.9 ± 0.5 | 53.4 ± 1.8 | 42.4 ± 1.9 | 60.3 ± 1.0 | 58.6 ± 0.8 | 74.4 ± 0.5 | 39.5 ± 1.0 | 59.4 ± 0.8 | 50.8 ± 0.9 | 58.52 ± 0.39 | −0.1 ± 0.4 |
| *Reference aware* | | | | | | | | | | | |
| OPD+$\alpha R$ | 91.7 ± 0.5 | 53.5 ± 1.7 | 43.1 ± 1.9 | 60.6 ± 1.0 | 58.8 ± 0.8 | 74.7 ± 0.5 | 39.7 ± 0.9 | 59.7 ± 0.7 | 51.1 ± 0.9 | 58.77 ± 0.38 | −0.3 ± 0.4 |
| UGS | 91.9 ± 0.4 | 54.1 ± 1.8 | 42.7 ± 2.0 | 60.8 ± 1.1 | 59.0 ± 0.7 | 74.7 ± 0.6 | 39.7 ± 1.0 | 59.8 ± 0.8 | 51.1 ± 0.8 | 58.86 ± 0.36 | −0.1 ± 0.3 |
| A-GEM-style | 91.7 ± 0.5 | 53.6 ± 2.0 | 43.0 ± 1.8 | 60.7 ± 1.2 | 58.9 ± 0.9 | 74.6 ± 0.6 | 39.6 ± 1.1 | 59.6 ± 0.9 | 51.0 ± 1.0 | 58.74 ± 0.41 | −0.3 ± 0.5 |
| Blockwise projection | 92.0 ± 0.4 | 54.7 ± 1.7 | 43.1 ± 1.9 | 61.0 ± 1.0 | 59.4 ± 0.7 | 75.2 ± 0.5 | 40.2 ± 0.9 | 60.0 ± 0.7 | 51.6 ± 0.8 | 59.24 ± 0.35 | 0.0 ± 0.3 |
| *Ours* | | | | | | | | | | | |
| PCGR | 92.7 ± 0.3 | 55.9 ± 1.6 | **45.1** ± 1.8 | 62.0 ± 0.9 | 60.5 ± 0.6 | 76.5 ± 0.5 | 41.5 ± 0.8 | 61.0 ± 0.7 | 53.0 ± 0.8 | 60.45 ± 0.33 | +0.7 ± 0.3 |
| PCGR+IW | **92.9** ± 0.4 | **56.9** ± 1.7 | **45.1** ± 1.9 | **62.3** ± 1.0 | **60.9** ± 0.7 | **76.8** ± 0.4 | **42.1** ± 0.9 | **61.4** ± 0.6 | **53.5** ± 0.7 | **60.87** ± 0.34 | +**0.9** ± 0.3 |

## 5.2 RQ2: Main results

PCGR achieves the best macro score among single-component methods (Table 2): it improves over OPD by 2.50 points (paired 95% interval [1.8, 3.2]), over IW-OPD by 1.87, and over the strongest baseline, blockwise projection, by 1.21, and it is the only single-component method that improves every benchmark over the initial student. The controls identify the source of the gain. *It is not a smaller step:* norm-matched OPD recovers only 8% of it. *Blockwise allocation matters:* UGS, with the same reference and budgets but a scalar gate, trails PCGR by 1.59 points, so the distortion advantage of Proposition 1 carries over to training. *It is not extra supervision:* OPD+$\alpha R$, which trains on the verified responses directly, gains less. *It is not an anchor:* OPD+KL(init) reaches only 58.52. Reference-free parameter masks (CPI-FT, EPI) give smaller gains, and the strongest baseline is itself a blockwise method, consistent with conflict being blockwise. PCGR+IW is best overall, so token- and parameter-level allocation are complementary. Relative to the teacher, PCGR closes 26% of the student–teacher macro gap, twice the 13% closed by OPD.

## 5.3 RQ3: Fewer regressions without slower gains

Average scores can hide the trade-off that motivated PCGR, so we examine which problems are lost and which are gained (Figure 3; per-family numbers in Appendix Table 7). The regressions of OPD are real: resampling the initial student gives a family-average regression rate of about 3%, whereas OPD regresses on 8.0–17.4% of initially solved problems. They are not a step-size effect: norm-matched OPD still regresses on 13.0%, versus 13.9% for OPD. Scalar methods trade stability for learning: IW-OPD, UGS, and OPD+KL(init) lower RR but also GR, and sweeping a constant UGS budget on mathematics lowers RR from 6.8% to 4.6% only by lowering GR from 20.0% to 15.0% (Figure 3a). PCGR leaves this frontier: it lowers RR to 5.0–9.4% while its GR matches or exceeds that of OPD on every family, making PCGR and PCGR+IW the only methods that improve on OPD in both rates (Figure 3b). It is also the only single-component method with a nonnegative $\Delta_{\text{old}}$ while nearly doubling the $\Delta_{\text{new}}$ of OPD (10.3 versus 5.3 points), and raising $\rho$ to 1 lowers its mathematics RR to 4.5% at the GR of OPD. Blockwise projection removes all first-order conflict yet regresses more than PCGR; a likely explanation is that it adds reference directions estimated from only $M = 8$ responses, whereas PCGR only rescales OPD components. PCGR is also neutral on the families it was not trained on ($-0.05$ points, versus $-0.98$ for OPD; Appendix H).

## 5.4 RQ4: Which components matter?

We change one component at a time and measure the average over MATH-500, AIME, IFEval, and IFBench (Figure 3c; variants defined in Appendix D, full results in Table 8). *Prefix information* (blue): removing it entirely (b) costs 1.35 points, and a constant budget with discrepancy grouping (a) recovers only part of that loss, so discrepancy helps through both grouping and budgeting;

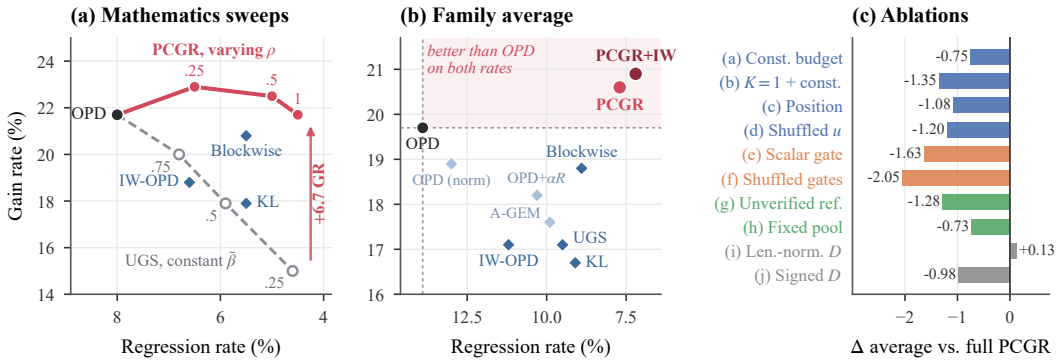


Figure 3: **Regressions, gains, and ablations** (Qwen3-1.7B). In (a) and (b) the regression axis is reversed, so up and right is better. (a) On mathematics, lowering a scalar budget trades gains for fewer regressions, whereas raising PCGR's strength $\rho$ lowers regressions at the gain rate of OPD; at a matched regression rate, PCGR gains 6.7 points more. (b) PCGR and PCGR+IW are the only methods in the shaded region, which beats OPD on both rates (light: norm-matched OPD, OPD+$\alpha R$, A-GEM). Resampling the initial student gives RR 2.9%. (c) Change in the mathematics and instruction average relative to full PCGR (65.30, seed std $\approx$0.5), colored by prefix information, location, reference, and statistic. **Full results** in Appendix Tables 7 and 8.

position-based routing (c) and shuffled discrepancy (d) fall in between, so the benefit is not a position effect. *Location* (orange): scalar gating with the same budgets (e) costs 1.63 points and shuffled block gates (f) cost 2.05, so where conflict is removed matters, not only how much. *Reference* (green): an unverified reference (g) removes about half of PCGR's gain over OPD, whereas a fixed pool of initial successes (h) costs less, so verification matters more than freshness. *Statistic* (gray): length-normalized discrepancy (i) matches the default, and the signed sum (j) is worse.

### 5.5 Robustness: scale, mixed-task training, optimizer, and compute

The gains carry over beyond the main setting (Appendix Figure 5, with per-family results, cost, and learning curves in Appendix I). At 4B, PCGR improves over OPD by 1.98 points and over the strongest baseline by 0.73; under mixed-task training, which adds inter-task conflict, it improves over OPD by 2.40 points. Under AdamW, which normalizes and accumulates the routed gradient, PCGR keeps 82% of its RMS-optimizer gain over OPD, and the realized first-order reference effect of each step correlates with the routed prediction at 0.80. With $K = 2$, a PCGR step takes about $1.20\times$ the time of an OPD step, because rollout generation and teacher scoring dominate; at the GPU-hour budget of OPD, PCGR still leads by 1.68 points, and it reaches OPD's final score with about 29% fewer GPU-hours. The gain over OPD is thus largest in the main setting and smallest at equal compute, where PCGR's extra backward passes are charged against it; even there it leads every baseline.

## 6 Conclusion

Scalar token weighting in OPD must shrink an entire token gradient to suppress any part of it. PCGR separates how much conflict with verified behavior an update may keep, set by prefix discrepancy, from where that conflict is removed, set by blockwise reference geometry, and computes the allocation with a closed-form gate rule that dominates the corresponding group-level scalar gate at matched reference conflict. In standard OPD the conflict-concentration coefficient is far below one, and PCGR turns this headroom into higher scores and fewer regressions without slower gains. More broadly, token-level allocation in distillation can be complemented by allocation over parameters, and $\kappa$ offers a cheap test of when this pays off.

**Limitations.** PCGR requires a verifier or another source of trusted references, and outcome verification does not certify intermediate reasoning. The analysis is first order and exact only for the RMS geometry, and the extra backward passes add about 20% training time; inference is unchanged.

### REPRODUCIBILITY STATEMENT

Section 3 and Algorithm 1 fully specify PCGR, and Appendix B contains complete proofs of all propositions. Section 4 and Appendices C, D, and J give the implementation settings, training hyperparameters, dataset configurations, data splits, verifiers, baselines, and compute, and Appendices E–I report additional experimental results. Due to organizational confidentiality constraints, the source code cannot be released during the review period; we intend to release an implementation upon publication, subject to the applicable code-release policies.

### AI USE STATEMENT

Generative AI tools were used solely to polish the language of the manuscript, such as improving grammar, wording, and readability of author-written text. AI was not used for conceptual development, the mathematical derivations, the methodology, the experiments, the implementation, the data, or the analysis, all of which were carried out by the authors. The authors reviewed all AI-assisted edits to ensure that they did not alter technical content, and take full responsibility for the final manuscript.

## REFERENCES

Rishabh Agarwal, Nino Vieillard, Yongchao Zhou, Piotr Stanczyk, Sabela Ramos Garea, Matthieu Geist, and Olivier Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In *ICLR*, 2024. URL https://openreview.net/forum?id=3zKtaqxLhW.

Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=Hkf2_sC5FX.

Alex Cloud, Jacob Goldman-Wetzler, Evžen Wybitul, Joseph Miller, and Alexander Matt Turner. Gradient routing: Masking gradients to localize computation in neural networks, 2024. URL https://arxiv.org/abs/2410.04332.

Ganqu Cui, Lifan Yuan, Zefan Wang, Hanbin Wang, Yuchen Zhang, Jiacheng Chen, Wendi Li, Bingxiang He, Yuchen Fan, Tianyu Yu, Qixin Xu, Weize Chen, Jiarui Yuan, Huayu Chen, Kaiyan Zhang, Xingtai Lv, Shuo Wang, Yuan Yao, Xu Han, Hao Peng, Yu Cheng, Zhiyuan Liu, Maosong Sun, Bowen Zhou, and Ning Ding. Process reinforcement through implicit rewards. *arXiv preprint arXiv:2502.01456*, 2025. URL https://arxiv.org/abs/2502.01456.

Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning, 2019. URL https://arxiv.org/abs/1910.07104.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=5h0qf7IBZZ.

Zhiwei He, Tian Liang, Jiahao Xu, Qiuzhi Liu, Xingyu Chen, Yue Wang, Linfeng Song, Dian Yu, Zhenwen Liang, Wenxuan Wang, Zhuosheng Zhang, Rui Wang, Zhaopeng Tu, Haitao Mi, and Dong Yu. Deepmath-103k: A large-scale, challenging, decontaminated, and verifiable mathematical dataset for advancing reasoning, 2025. URL https://arxiv.org/abs/2504.11456.

Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network, 2015. URL https://arxiv.org/abs/1503.02531.

Yoon Kim and Alexander M. Rush. Sequence-level knowledge distillation. In Jian Su, Kevin Duh, and Xavier Carreras (eds.), *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1317–1327, Austin, Texas, November 2016. Association for Computational Linguistics. doi: 10.18653/v1/D16-1139. URL https://aclanthology.org/D16-1139/.

James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell. Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526, 2017. doi: 10.1073/pnas.1611835114. URL https://www.pnas.org/doi/abs/10.1073/pnas.1611835114.

Jongwoo Ko, Sungnyun Kim, Tianyi Chen, and Se-Young Yun. Distillm: towards streamlined distillation for large language models. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C.C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Zekai Lin, Chao Xue, Di Liang, Xingsheng Han, Peiyang Liu, Xianjie Wu, Lei Jiang, Yu Lu, Bob Simons, Shuang Liang, and Minlong Peng. Parameter importance is not static: Evolving parameter isolation for supervised fine-tuning. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 32846–32864, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.1516. URL https://aclanthology.org/2026.acl-long.1516/.

Zhenghao Lin, Zhibin Gou, Yeyun Gong, Xiao Liu, Yelong Shen, Ruochen Xu, Chen Lin, Yujiu Yang, Jian Jiao, Nan Duan, and Weizhu Chen. Not all tokens are what you need for pretraining. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. Conflict-averse gradient descent for multi-task learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, NIPS '21, Red Hook, NY, USA, 2021. Curran Associates Inc. ISBN 9781713845393.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by chatgpt really correct? rigorous evaluation of large language models for code generation. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, NIPS '23, Red Hook, NY, USA, 2023. Curran Associates Inc.

Zuxin Liu, Thai Hoang, Jianguo Zhang, Ming Zhu, Tian Lan, Shirley Kokane, Juntao Tan, Weiran Yao, Zhiwei Liu, Yihao Feng, Rithesh Murthy, Liangwei Yang, Silvio Savarese, Juan Carlos Niebles, Huan Wang, Shelby Heinecke, and Caiming Xiong. Apigen: automated pipeline for generating verifiable and diverse function-calling datasets. In *Proceedings of the 38th International Conference on Neural Information Processing Systems*, NIPS '24, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9798331314385.

David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 6470–6479, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Kevin Lu and Thinking Machines Lab. On-policy distillation. *Thinking Machines Lab: Connectionism*, 2025. doi: 10.64434/tml.20251026. https://thinkingmachines.ai/blog/on-policy-distillation.

Yun Luo, Zhen Yang, Fandong Meng, Yafu Li, Jie Zhou, and Yue Zhang. An empirical study of catastrophic forgetting in large language models during continual fine-tuning. *IEEE Transactions on Audio, Speech and Language Processing*, 33:3776–3786, 2025. doi: 10.1109/TASLPRO.2025.3606231.

Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning, 2018. URL https://arxiv.org/abs/1711.05769.

Sagnik Mukherjee, Lifan Yuan, Dilek Hakkani-Tür, and Hao Peng. Reinforcement learning finetunes small subnetworks in large language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=0NdS4xCngO.

Shishir G Patil, Huanzhi Mao, Fanjia Yan, Charlie Cheng-Jie Ji, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (BFCL): From tool use to agentic evaluation of large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Proceedings of the 42nd International Conference on Machine Learning*, volume 267 of *Proceedings of Machine Learning Research*, pp. 48371–48392. PMLR, 13–19 Jul 2025. URL https://proceedings.mlr.press/v267/patil25a.html.

Valentina Pyatkin, Saumya Malik, Victoria Graf, Hamish Ivison, Shengyi Huang, Pradeep Dasigi, Nathan Lambert, and Hannaneh Hajishirzi. Generalizing verifiable instruction following. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=yfYgwjj5F8.

Mengye Ren, Wenyuan Zeng, Bin Yang, and Raquel Urtasun. Learning to reweight examples for robust deep learning, 2019. URL https://arxiv.org/abs/1803.09050.

Stephane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In Geoffrey Gordon, David Dunson, and Miroslav Dudík (eds.), *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pp. 627–635, Fort Lauderdale, FL, USA, 11–13 Apr 2011. PMLR. URL https://proceedings.mlr.press/v15/ross11a.html.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Idan Shenfeld, Jyothish Pari, and Pulkit Agrawal. Rl's razor: Why online reinforcement learning forgets less, 2025. URL https://arxiv.org/abs/2509.04259.

Zafir Stojanovski, Oliver Stanley, Joe Sharratt, Richard Jones, Abdulhakeem Adefioye, Jean Kaddour, and Andreas Köpf. Reasoning gym: Reasoning environments for reinforcement learning with verifiable rewards. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2026. URL https://openreview.net/forum?id=GqYSunGmp7.

Yao Wang, Di Liang, and Minlong Peng. Not all parameters are created equal: Smart isolation boosts fine-tuning performance. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 9871–9885, Suzhou, China, November 2025. Association for Computational Linguistics. ISBN 979-8-89176-332-6. doi: 10.18653/v1/2025.emnlp-main.500. URL https://aclanthology.org/2025.emnlp-main.500/.

Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=F1vEjWK-lH_.

Yan Xie, Sijie Zhu, Tiansheng Wen, Bo Chen, and Yifei Wang. On the position bias of on-policy distillation, 2026. URL https://arxiv.org/abs/2606.22600.

Runxin Xu, Fuli Luo, Zhiyuan Zhang, Chuanqi Tan, Baobao Chang, Songfang Huang, and Fei Huang. Raise a child in large language model: Towards effective and generalizable fine-tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih

(eds.), *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 9514–9528, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.749. URL https://aclanthology.org/2021.emnlp-main.749/.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jing Zhou, Jingren Zhou, Junyang Lin, Kai Dang, Keqin Bao, Kexin Yang, Le Yu, Lianghao Deng, Mei Li, Mingfeng Xue, Mingze Li, Pei Zhang, Peng Wang, Qin Zhu, Rui Men, Ruize Gao, Shixuan Liu, Shuang Luo, Tianhao Li, Tianyi Tang, Wenbiao Yin, Xingzhang Ren, Xinyu Wang, Xinyu Zhang, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yinger Zhang, Yu Wan, Yuqiong Liu, Zekun Wang, Zeyu Cui, Zhenru Zhang, Zhipeng Zhou, and Zihan Qiu. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models, 2023. URL https://arxiv.org/abs/2308.01825.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: self-taught reasoner bootstrapping reasoning with reasoning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2022. Curran Associates Inc. ISBN 9781713871088.

# A More related work

**Learning from verified self-generated responses.** Verified student outputs are a common training signal. STaR and rejection-sampling fine-tuning train on self-generated solutions that reach correct answers (Zelikman et al., 2022; Yuan et al., 2023), and reinforcement learning with verifiable rewards optimizes outcome success directly (Shao et al., 2024). PCGR uses the same kind of data for a different purpose: verified successes are not training targets but define a local reference that constrains where the OPD update may act. An OPD-plus-reference-loss baseline separates the effect of routing from the effect of simply training on verified responses.

**Parameter isolation and masked updates.** Another line of work reduces interference by restricting which parameters are updated. PackNet frees capacity through iterative pruning (Mallya & Lazebnik, 2018), Child-Tuning masks gradients to a sub-network chosen at random or by Fisher information (Xu et al., 2021), EWC penalizes changes to parameters important for earlier tasks (Kirkpatrick et al., 2017), and CPI-FT and EPI isolate important parameters during supervised fine-tuning, with importance fixed or evolving over training (Wang et al., 2025; Lin et al., 2026). Gradient Routing applies data-dependent masks to localize what a network learns (Cloud et al., 2024), and RL fine-tuning of LLMs has been observed to update only a small sub-network (Mukherjee et al., 2025). These masks are determined by parameter importance, data labels, or training dynamics. PCGR's gates are recomputed at every step from the interaction between the current OPD gradient and a trusted reference, and satisfy an explicit, prefix-dependent conflict budget.

**Reference-guided example weighting.** Ren et al. (2019) reweight training examples by aligning their gradients with those of a small clean validation set. PCGR can be viewed as a blockwise, budget-constrained analogue of this idea for OPD: the trusted set plays the role of the clean reference, but the output is a vector of block gates rather than one scalar weight per example.

# B Proofs

Throughout, we fix one group, drop the index $j$, and use the notation of Section 3.5. Blocks with $E_b = 0$ satisfy $G_b = 0$, so they contribute neither to the objective nor to the constraint of Eq. (7);

their gates are set to one by convention. In particular, every conflicting block lies in $\mathcal{S}$. We first establish the structure of the routing solution used in Section 3.4, and then restate and prove the two propositions of the main text.

**Lemma 1** (Structure of the routing solution). *Suppose $H > 0$ and $0 < \beta < 1$. Problem (7) has a unique solution on $\mathcal{S}$, given by Eq. (8) with a dual variable $\lambda^\star \in (0, \lambda_{\max})$, where $\lambda_{\max} = \max_{b:H_b>0} E_b/H_b$. The budget constraint is active, $\sum_b m_b^\star H_b = \beta H$, and blocks with $H_b = 0$ receive $m_b^\star = 1$. The retained conflict $\phi(\lambda) = \sum_b H_b[1 - \lambda H_b/E_b]_{[0,1]}$ is continuous and nonincreasing with $\phi(0) = H$ and $\phi(\lambda_{\max}) = 0$, so bisection on $\phi(\lambda) = \beta H$ over $[0, \lambda_{\max}]$ recovers $\lambda^\star$. If $\beta = 0$, all conflicting gates are zero and all other gates are one.*

*Proof.* On $\mathcal{S}$ the objective is strictly convex, and the feasible set is a nonempty compact convex set because $m = 0$ is feasible. Hence a unique minimizer exists. The point $m_b = \beta/2$ for all $b \in \mathcal{S}$ lies in the interior of the box and satisfies $\sum_b m_b H_b = \frac{\beta}{2} H < \beta H$, so Slater's condition holds and the KKT conditions are necessary and sufficient. For a multiplier $\lambda \geq 0$, the Lagrangian separates across blocks, and minimizing $\frac{1}{2}E_b(1 - m_b)^2 + \lambda H_b m_b$ over $m_b \in [0, 1]$ gives $m_b = [1 - \lambda H_b/E_b]_{[0,1]}$, which is Eq. (8). Blocks with $H_b = 0$ therefore receive $m_b = 1$ for every $\lambda$. If the budget constraint were inactive, complementary slackness would give $\lambda = 0$ and $m = \mathbf{1}$, whose retained conflict is $H > \beta H$, a contradiction. Hence the constraint is active and $\lambda^\star > 0$. Each term of $\phi$ is continuous and nonincreasing in $\lambda$, so $\phi$ is continuous and nonincreasing, with $\phi(0) = H$. At $\lambda_{\max}$, every block with $H_b > 0$ satisfies $1 - \lambda_{\max} H_b/E_b \leq 0$ and receives gate zero, so $\phi(\lambda_{\max}) = 0$. Because $0 < \beta H < H$, the intermediate value theorem yields $\lambda^\star \in (0, \lambda_{\max})$ with $\phi(\lambda^\star) = \beta H$, and bisection on $[0, \lambda_{\max}]$ converges to it. □

**Proposition 1** (Blockwise routing versus scalar gating). *Suppose $H > 0$ and $0 < \beta < 1$. Then:*

*(i) Equal conflict: $\sum_b m_b^\star H_b = \sum_b m_b^{\mathrm{sc}} H_b = \beta H$.*

*(ii) Lower distortion: $\kappa \operatorname{Dist}(m^{\mathrm{sc}}) \leq \operatorname{Dist}(m^\star) \leq \operatorname{Dist}(m^{\mathrm{sc}}) = \frac{1}{2}(1 - \beta)^2 \sum_b E_b$. The first inequality holds with equality whenever no gate saturates at zero, and the second is strict if and only if $\kappa < 1$.*

*(iii) Retained compatible progress: $r^\top P(m^\star \odot G) = Q - \beta H = r^\top P(m^{\mathrm{sc}} \odot G) + (1 - \beta)Q$.*

*Proof.* (i) The first equality is the active constraint of Lemma 1, and $\sum_b \beta H_b = \beta H$.
(ii), upper bound. The scalar gate $m^{\mathrm{sc}}$ is feasible and $m^\star$ is optimal, so $\operatorname{Dist}(m^\star) \leq \operatorname{Dist}(m^{\mathrm{sc}}) = \frac{1}{2}(1 - \beta)^2 \sum_b E_b$. By uniqueness, equality holds if and only if $m_b^\star = \beta$ for all $b \in \mathcal{S}$. Since $\beta \in (0, 1)$ lies in the interior of the box, Eq. (8) then requires $1 - \lambda^\star H_b/E_b = \beta$, that is, $H_b/E_b = (1 - \beta)/\lambda^\star$ for all $b \in \mathcal{S}$. Conversely, if $H_b/E_b = c$ for all $b \in \mathcal{S}$, then $c > 0$ because $H > 0$, and $\lambda = (1 - \beta)/c$ produces $m_b = \beta$ for all $b \in \mathcal{S}$, which satisfies the KKT conditions with an active constraint; hence $m^\star = m^{\mathrm{sc}}$. Thus equality holds if and only if $H_b/E_b$ is constant on $\mathcal{S}$. By the Cauchy–Schwarz inequality,

$$\Big(\sum_{b\in\mathcal{S}} H_b\Big)^2 = \Big(\sum_{b\in\mathcal{S}} \frac{H_b}{\sqrt{E_b}}\sqrt{E_b}\Big)^2 \leq \Big(\sum_{b\in\mathcal{S}} \frac{H_b^2}{E_b}\Big)\Big(\sum_{b\in\mathcal{S}} E_b\Big),$$

with equality if and only if $H_b/\sqrt{E_b} \propto \sqrt{E_b}$, that is, if and only if $H_b/E_b$ is constant on $\mathcal{S}$. Hence $\kappa \in (0, 1]$, and the upper bound is strict if and only if $\kappa < 1$.
(ii), lower bound. Removing the box constraints gives the relaxation $\min_{m\in\mathbb{R}^{\mathcal{S}}} \operatorname{Dist}(m)$ subject to $\sum_b m_b H_b \leq \beta H$. Its KKT conditions give $m_b = 1 - \lambda H_b/E_b$, and the active constraint gives $\lambda = (1 - \beta)H/\sum_{b\in\mathcal{S}}(H_b^2/E_b)$. The relaxed optimal value is

$$\frac{1}{2}\lambda^2 \sum_{b\in\mathcal{S}} \frac{H_b^2}{E_b} = \frac{1}{2}(1 - \beta)^2 \frac{H^2}{\sum_{b\in\mathcal{S}} H_b^2/E_b} = \kappa \operatorname{Dist}(m^{\mathrm{sc}}).$$

A relaxation cannot have a larger optimal value, so $\operatorname{Dist}(m^\star) \geq \kappa \operatorname{Dist}(m^{\mathrm{sc}})$. If no gate of $m^\star$ saturates at zero, then $m_b^\star = 1 - \lambda^\star H_b/E_b$ for all $b \in \mathcal{S}$ satisfies the KKT conditions of the convex relaxation, so $m^\star$ is also its minimizer and equality holds.
(iii) Since $r^\top P(m \odot G) = \sum_b m_b a_b$, split the blocks by the sign of $a_b$. Blocks with $a_b > 0$ have $H_b = 0$, so $m_b^\star = 1$, whereas blocks with $a_b \leq 0$ have $a_b = -H_b$. Hence $\sum_b m_b^\star a_b = Q - \sum_b m_b^\star H_b = Q - \beta H$ by (i). For the scalar gate, $\sum_b \beta a_b = \beta(Q - H)$. The difference is $(1 - \beta)Q$. □

**Scope of Proposition 1.** The proposition compares gates on the same group and measures conflict by the negative blockwise effects $H_b$, the quantity PCGR controls. Matching the net first-order reference effect or the first-order OPD progress $\sum_b m_b E_b$ instead would define a different scalar competitor. Per-token scalar weights such as IW-OPD act at a finer granularity than one group and are compared empirically. Replacing the quadratic objective of Eq. (7) by the linear progress $\sum_b m_b E_b$ would give a fractional-knapsack solution whose hard gates switch discontinuously with the statistics; the quadratic objective yields soft gates that vary continuously. Finally, the gates are invariant to $r \to cr$ for $c > 0$ because every $H_b$ and the budget scale by the same factor.

**Proposition 2** (Local change of the reference objective)**.** *For the aggregated update of Eq.* (9) *and the unrouted OPD gradient $G = \sum_j G_j$,*

$$r^\top P\widetilde{G} = \sum_j (Q_j - \beta_j H_j) =: S_k, \qquad r^\top P(\widetilde{G} - G) = \sum_j (1-\beta_j) H_j \geq 0.$$

*If the reference loss $R_k$ is L-smooth along the step and $S_k > 0$, every step size $0 < \eta_k < 2S_k/(L\|P\widetilde{G}\|_2^2)$ decreases $R_k$.*

*Proof.* Because $P$ is block diagonal, $r^\top P\widetilde{G} = \sum_j \sum_b m_{j,b} a_{j,b}$. We show $\sum_b m_{j,b} a_{j,b} = Q_j - \beta_j H_j$ for every group. If $H_j = 0$ or $\beta_j = 1$, all gates equal one and the sum is $Q_j - H_j = Q_j - \beta_j H_j$ (with $H_j = 0$ in the first case). If $H_j > 0$ and $\beta_j = 0$, conflicting gates are zero and compatible gates are one, so the sum is $Q_j$. If $H_j > 0$ and $0 < \beta_j < 1$, the identity is Proposition 1(iii). Summing over groups gives the first identity. For the unrouted gradient, all gates equal one, so the same sign split gives $r^\top PG = \sum_j (Q_j - H_j)$; subtracting yields the second identity. $L$-smoothness along the step $\Delta = \eta_k P\widetilde{G}$ gives $R_k(\theta^k - \Delta) \leq R_k(\theta^k) - r^\top \Delta + \frac{L}{2}\|\Delta\|_2^2 = R_k(\theta^k) - \eta_k S_k + \frac{L\eta_k^2}{2}\|P\widetilde{G}\|_2^2$, which is below $R_k(\theta^k)$ for $0 < \eta_k < 2S_k/(L\|P\widetilde{G}\|_2^2)$. □

## C Implementation details

Algorithm 1 gives one PCGR iteration. The OPD loss of Eq. (1) is a token-local semi-gradient: rollout occupancy is held fixed, so it is not the full gradient of a trajectory-level KL objective. The discrepancy scale $\tau_d$ is the median positive $D_{i,t}$ over initial-student rollouts on 128 training prompts of task $d$, selected with a fixed seed ($\tau_d = 1$ if no positive value occurs); its cost is included in the training budget. Because $u_{i,t}$ is not normalized by the terminal discrepancy of each response, low- and high-discrepancy responses keep different scores. The dual variable is found with $n_{\text{bis}} = 20$ bisection steps. Because a group's tokens are scattered across the batch, every $G_j$ requires its own forward and backward pass; all groups in a step share the frozen preconditioner $P^k$. With AdamW, the gates act on the optimizer's input rather than on its step: adaptive normalization can partly undo a soft attenuation, and momentum can move a block whose current gate is zero. We therefore log the realized step $\Delta\theta_k = \theta^{k+1} - \theta^k$ and compare its reference effect with the routed prediction (Appendix E). Length-capped responses keep their observed-token losses but never enter the reference buffer.

## D Baseline and ablation details

All reference-aware baselines share PCGR's verifier access, buffer rules ($C = 256$, $M \leq 8$), and the frozen RMS preconditioner $P^k$; they differ only in how the reference gradient $r$ enters the update.

**OPD (norm-matched).** On each batch we compute the candidate PCGR update $P\widetilde{G}$, take its parameter-step norm, and rescale the unrouted update $PG$ to that norm; zero-norm steps are left unscaled and the candidate-routing cost is charged to this control.

**IW-OPD.** We use the discrepancy-based token weights of Xie et al. (2026), $w_{i,t} = 1 + \gamma\big(1 - D_{i,t}/D_{i,T_i}\big)$ with $\gamma$ searched in $\{0.25, 0.5, 1\}$ (0.5 selected) and the weight clipped to $[1, 1.5]$, where $D_{i,t}$ is the accumulated discrepancy of Eq. (2) and $D_{i,T_i}$ its terminal value; the weight is detached and set to one when $D_{i,T_i} = 0$. Relative to uniform OPD this holds the earliest tokens near the 1.5 ceiling and returns the highest-discrepancy tokens to the baseline weight of one.

**Algorithm 1** One PCGR training iteration for task $d$

**Input:** student $\theta^k$, teacher $\pi_T$, preconditioner $P^k$, reference buffer $\mathcal{B}_d$, number of bins $K$, strength $\rho$, scale $\tau_d$, step size $\eta_k$

**① Rollout and prefix scoring**

Sample prompts $\{x_i\}$ and rollouts $y_i \sim \pi_{\theta^k}(\cdot \mid x_i)$; score them with $\pi_T$ to obtain $A_{i,t}$
Compute $u_{i,t}$ by Eq. (2) and assign tokens to bins $\mathcal{G}_1, \ldots, \mathcal{G}_K$

**② Trusted reference**

$\mathcal{E}_k \leftarrow \{(x, z) \in \mathcal{B}_d : x \text{ not in the current batch}\}$ ▷ *eligible references*
**if** $\mathcal{E}_k = \emptyset$ **then**
    $\widetilde{G} \leftarrow \nabla_\theta \frac{1}{N} \sum_{i,t} w_{i,t} \ell_{i,t}(\theta^k)$ ▷ *routing bypassed*
**else**
    Sample $\mathcal{A}_k \subseteq \mathcal{E}_k$ with $|\mathcal{A}_k| = \min(M, |\mathcal{E}_k|)$; $r \leftarrow \nabla_\theta R_k(\theta^k)$

**③ Prefix-aware budget and blockwise routing**

    $\widetilde{G} \leftarrow 0$
    **for** each group $j$ with $\sum_{(i,t)\in\mathcal{G}_j} w_{i,t} > 0$ **do**
        $G_j \leftarrow \nabla_\theta \frac{1}{N} \sum_{(i,t)\in\mathcal{G}_j} w_{i,t} \ell_{i,t}(\theta^k)$ ▷ *one backward sweep*
        Compute $E_{j,b}, a_{j,b}, H_{j,b}$ by Eq. (5) for all $b$; $\beta_j \leftarrow 1 - \rho \bar{u}_j$
        $\lambda_j \leftarrow \textsc{Dual}(E_{j,\cdot}, H_{j,\cdot}, \beta_j)$; $m_{j,b} \leftarrow [1 - \lambda_j H_{j,b}/E_{j,b}]_{[0,1]}$, with $m_{j,b} \leftarrow 1$ if $E_{j,b} = 0$
        $\widetilde{G}_b \leftarrow \widetilde{G}_b + m_{j,b} G_{j,b}$ for all $b$; release $G_j$

**④ Update**

$\theta^{k+1} \leftarrow \theta^k - \eta_k P^k \widetilde{G}$; update $v^{k+1}$ as in Section 3.6
Admit verified, non-truncated responses to $\mathcal{B}_d$ (at most one per prompt, FIFO, capacity $C$)

**function** $\textsc{Dual}(E, H, \beta)$ ▷ *only blocks with $E_b > 0$ enter*
    **if** $\sum_b H_b = 0$ or $\beta = 1$ **then**
        **return** 0
    $\lambda_{\text{lo}} \leftarrow 0$; $\lambda_{\text{hi}} \leftarrow \max_{b: E_b>0,\, H_b>0} E_b/H_b$
    **if** $\beta = 0$ **then**
        **return** $\lambda_{\text{hi}}$ ▷ *all conflicting gates become zero*
    **for** $n = 1, \ldots, n_{\text{bis}}$ **do**
        $\lambda \leftarrow (\lambda_{\text{lo}} + \lambda_{\text{hi}})/2$
        **if** $\sum_{b:E_b>0} H_b[1 - \lambda H_b/E_b]_{[0,1]} > \beta \sum_b H_b$ **then**
            $\lambda_{\text{lo}} \leftarrow \lambda$
        **else**
            $\lambda_{\text{hi}} \leftarrow \lambda$
    **return** $\lambda_{\text{hi}}$ ▷ *feasible side of the bracket*

**CPI-FT / EPI.** Both use a per-parameter importance score. For CPI-FT the importance is the diagonal of the squared OPD gradient accumulated over one 64-prompt update, and the update is concentrated on the top-$s$ core set. For EPI the importance is an exponential moving average $I_k = \mu I_{k-1} + (1-\mu) g_k^2$ with $\mu = 0.9$, normalized within each layer, and the top-$s$ fraction is *shielded*: the applied step is $\theta^{k+1} = \theta^k - \eta_k P\,(1-m) \odot G$, so only the less-important parameters move. We searched $s \in \{0.1, 0.3, 0.5\}$ and a refresh period in $\{\infty, 50\}$ steps, selecting a static $s = 0.3$ core for CPI-FT and $s = 0.3$ shielded and refreshed every 50 steps for EPI. Because OPD has no separate task-tuning phase, these adapt the original methods rather than reproduce them.

**OPD+$\alpha R$.** We add $\alpha r$ with $\alpha$ searched in $\{0.1, 0.3, 1.0\}$; $\alpha = 0.3$ was selected on the E1 development split.

**OPD+KL(init).** We add $\beta_{\text{KL}} \widehat{\text{KL}}$ to the OPD loss, where $\pi_{\text{init}}$ is the frozen initial student and $\widehat{\text{KL}}$ is the token-level $k_3$ estimator $\pi_{\text{init}}/\pi_\theta - \log(\pi_{\text{init}}/\pi_\theta) - 1$ averaged over generated response tokens. The coefficient is searched in $\beta_{\text{KL}} \in \{0.01, 0.05, 0.1\}$ and $\beta_{\text{KL}} = 0.05$ was selected on the E1 development split; this baseline uses no reference gradient.

**UGS.** $m_{j,b} = \beta_j$ for groups with $H_j > 0$ and 1 otherwise, using the same $\beta_j = 1 - \rho \bar{u}_j$ as PCGR.

**A-GEM-style and blockwise projection.** Both use the $P$-metric inner product $r^\top P(\cdot)$. A-GEM replaces a group gradient with $r^\top P G_j < 0$ by $G_j - (r^\top P G_j / r^\top P r)\, r$, and blockwise projection applies the same projection to each block with $a_{j,b} < 0$; unlike PCGR, both remove all conflict,

ignore the prefix, and add reference directions to the update. A zero reference norm requires no projection.

**Ablations.** The variants of Section 5.4 are: *Prefix information*—(a) a constant budget $\beta_j = \bar{\beta}$, matched to the average PCGR budget on the calibration rollouts, with discrepancy grouping kept, which tests the adaptive budget alone; (b) $K = 1$ with the same constant budget, which removes prefix information entirely and leaves reference-aware blockwise routing; (c) grouping and budget both computed from a normalized position score calibrated in the same way, which replaces discrepancy by position; (d) discrepancy shuffled within task and position strata before both grouping and budgeting. *Location*—(e) no blockwise routing, which is UGS; (f) shuffled block gates, which permute the PCGR gates across blocks within each group and keep the gate distribution but not its location. *Reference*—(g) an unverified reference built from earlier student responses regardless of outcome; (h) a fixed reference pool built once from initial-student successes. *Discrepancy statistic*—(i) length-normalized discrepancy $D_{i,t}/(t-1)$; (j) the magnitude of the signed sum, $|\sum_{s<t} A_{i,s}|$, in place of $D_{i,t}$. The constant budget for (a)–(c) is $\bar{\beta} = 1 - \rho\,\bar{u}$ evaluated at the calibration-mean discrepancy $\bar{u}$; the position score for (c) is $\mathrm{pos}/(\mathrm{pos} + \bar{T})$ with $\bar{T}$ the median response length; ablation (d) shuffles $u_{i,t}$ within (task, position-tercile) strata. Selected values for every method are fixed before final evaluation.

## E Mechanism diagnostics protocol

Diagnostics are computed on frozen snapshots of the E1 OPD run at 0%, 25%, 50%, and 100% of training, so they characterize the problem in standard OPD rather than a trajectory already shaped by PCGR. At each snapshot, all interventions share rollouts, teacher scores, references, and preconditioners. At each snapshot we draw 512 held-out prompts per task family (2,048 in total) and score all of their response tokens; the singleton-token statistics (D1, D2) are computed on a class-balanced subsample of 20,000 tokens per stratum, where strata are the $4 \times 3 \times 3$ combinations of task family, position tercile, and discrepancy tercile (about $7.2 \times 10^5$ tokens per snapshot), and the group statistics use both $K = 2$ groups of every sampled response. The one-step interventions (D3) use a separate 256 prompts and 256 verified responses per family, disjoint from the routing reference $\mathcal{A}_k$.

- D1 Within-update heterogeneity. On singleton-token subsets stratified by task, position, and $u_{i,t}$, we compute the blockwise effects $c_{i,t,b} = r_b^\top P_b g_{i,t,b}$. A block is nontrivial if $|c_{i,t,b}|$ exceeds 1% of $\sum_b |c_{i,t,b}|$. We report the fraction of token gradients with both compatible and conflicting nontrivial blocks, the share of gradient energy in conflicting blocks, the coefficient $\kappa$ of Eq. (10), the realized distortion ratio $\mathrm{Dist}(m^\star)/\,\mathrm{Dist}(m^{\mathrm{sc}})$, and the fraction of gates saturated at zero, for singleton tokens and for the $K = 2$ groups.
- D2 Discrepancy and conflict. We regress the conflict fraction $H/(H+Q)$ of singleton tokens on $u_{i,t}$ with task, position, response length, and initial-student difficulty as covariates, and report the coefficient on $u_{i,t}$ with cluster-bootstrap intervals over prompts. Figure 4b shows the raw trend within each position stratum.
- D3 One-step interventions. From the same snapshot and batch, we compare OPD, UGS, A-GEM-style projection, blockwise projection, and PCGR under two protocols. In the budget-matched protocol, UGS and PCGR use the same budgets $\beta_j$ and their native solutions, so retained conflict is equal while step norms may differ. In the norm-matched protocol, every update is rescaled to the same parameter-step norm, and we report the retained conflict after rescaling, because rescaling changes it. Gates in $[0, 1]$ do not guarantee a smaller aggregate norm, since routing can reduce cancellation between groups, so norms are measured rather than assumed. In both protocols we measure (a) the decrease in teacher–student reverse KL on held-out diagnostic prompts, which quantifies OPD progress, and (b) the change in NLL of held-out verified responses that are disjoint from the routing reference $\mathcal{A}_k$, which quantifies damage to verified behavior. Because both quantities are held out, utility is not measured by alignment with the reference used for routing. Results are in Table 3: at equal norm, the projection baselines reduce verified-response damage further than PCGR (to 0.52–0.57) but give up more KL progress (0.90–0.93), because they remove conflict by adding reference directions rather than by keeping compatible components intact.
- D4 Training-time statistics. For both optimizers we compare the realized first-order reference effect $\delta_k^{\mathrm{real}} = r^\top \Delta\theta_k$ with the effect predicted by routing, $\delta_k^{\mathrm{pred}} = -\eta_k r^\top P^k \widetilde{G}$; the two coincide for

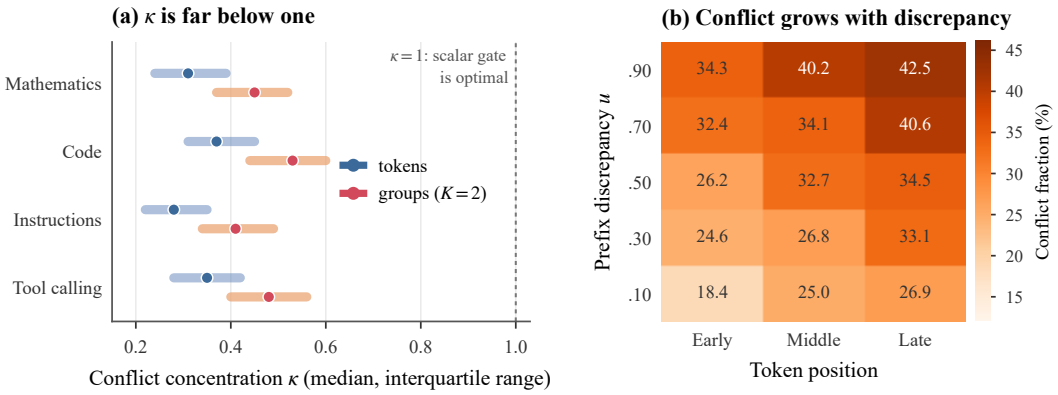


Figure 4: **Structure of blockwise conflict in OPD** (Qwen3-1.7B snapshots). (a) Median and interquartile range of $\kappa$ over samples with $H > 0$; $\kappa = 1$ is the only case in which a scalar gate is as good as blockwise routing. (b) Conflict fraction $H/(H+Q)$ by prefix discrepancy and token position: it rises with $u$ in every position stratum.

Table 3: One-step interventions on OPD snapshots, as ratios to the OPD update on a shared batch. KL progress: reverse-KL decrease on held-out prompts. NLL change: on held-out verified responses (lower is better). Projections have no budget and use their native updates.

| Protocol | Method | Conflict | Step norm | KL progress | NLL change |
|---|---|---|---|---|---|
| Budget-matched | OPD | 1.00 | 1.00 | 1.00 | 1.00 |
| | UGS | .64 | .80 | .74 | .60 |
| | A-GEM-style | .30 | .84 | .78 | .46 |
| | Blockwise projection | .00 | .82 | .80 | .40 |
| | PCGR | .64 | .92 | .91 | .52 |
| Norm-matched | OPD | 1.00 | 1.00 | 1.00 | 1.00 |
| | UGS | .80 | 1.00 | .90 | .78 |
| | A-GEM-style | .36 | 1.00 | .90 | .57 |
| | Blockwise projection | .00 | 1.00 | .93 | .52 |
| | PCGR | .70 | 1.00 | .97 | .63 |

the RMS variant, and their correlation under AdamW is reported in Table 10. To test reference stability, we recompute routing with an independently sampled reference batch and report the sign agreement of $a_{j,b}$, which is 0.86 pooled, and the rank correlation of gates, Spearman 0.79, both far from the 0.5 and 0 expected if $r$ were pure noise.

## F Sensitivity Analysis

The sensitivity grid varies $K \in \{1, 2, 4\}$, $\rho \in \{0, 0.25, 0.5, 1\}$, $M \in \{4, 8, 16\}$, the block granularity, and $\tau_d \in \{0.5, 1, 2\}\times$ median, one factor at a time (Table 4). Figure 3a contrasts the $\rho$ sweep with a scalar frontier: UGS with a constant budget $\bar{\beta} \in \{0.75, 0.5, 0.25\}$ lowers RR from 6.8% to 4.6% only by lowering GR from 20.0% to 15.0%, whereas PCGR with $\rho = 1$ reaches RR 4.5% at the GR of OPD.

## G Extended Task Suite

The extended suite tests whether the gains transfer to task families whose success predicates differ from the core suite. Reasoning Gym evaluates unseen instances of the training generators, harder instances of the same generators (difficulty shift), and generators never seen in training; SQL is evaluated by execution accuracy on BIRD dev. PCGR improves over OPD by 3.2, 3.5, 3.0, and 2.9 points on the four splits and over the strongest reference-aware baseline by 1.5–1.8 points on every split (Table 5), with the same method ordering as on the core suite.

Table 4: Sensitivity of PCGR on mathematics (Qwen3-1.7B), one factor at a time from the default (*). Varied rows use two seeds. Time is relative to OPD.

| Factor | Value | MATH-500 | AIME | RR | GR | Time / step |
|---|---|---|---|---|---|---|
| Bins $K$ | 1 | 92.1 | 49.0 | 6.1 | 21.9 | 1.10 |
| | 2* | 92.7 | 50.5 | 5.0 | 22.5 | 1.20 |
| | 4 | 92.9 | 50.8 | 4.8 | 22.7 | 1.41 |
| Strength $\rho$ | 0 (OPD) | 91.2 | 47.5 | 8.0 | 21.7 | 1.00 |
| | 0.25 | 92.1 | 49.1 | 6.5 | 22.9 | 1.20 |
| | 0.5* | 92.7 | 50.5 | 5.0 | 22.5 | 1.20 |
| | 1 | 92.6 | 49.7 | 4.5 | 21.7 | 1.20 |
| Reference batch $M$ | 4 | 92.4 | 50.0 | 5.5 | 22.2 | 1.15 |
| | 8* | 92.7 | 50.5 | 5.0 | 22.5 | 1.20 |
| | 16 | 92.8 | 50.6 | 4.9 | 22.7 | 1.29 |
| Block granularity | Tensor* | 92.7 | 50.5 | 5.0 | 22.5 | 1.20 |
| | Layer | 92.5 | 50.0 | 5.4 | 22.2 | 1.19 |
| | Module type | 92.1 | 49.4 | 5.9 | 21.8 | 1.18 |
| Scale $\tau_d$ | 0.5× median | 92.6 | 50.0 | 4.8 | 22.4 | 1.20 |
| | 1× median* | 92.7 | 50.5 | 5.0 | 22.5 | 1.20 |
| | 2× median | 92.3 | 49.7 | 6.0 | 21.9 | 1.20 |

Table 5: Extended suite with Qwen3-1.7B, single-task training, mean ± std over three seeds.

| | Reasoning Gym | | | SQL |
|---|---|---|---|---|
| Method | Unseen instances | Difficulty shift | Held-out generators | BIRD dev (EX) |
| Initial student | 55.0 | 38.0 | 32.0 | 18.5 |
| OPD | 59.8 ± 0.8 | 42.6 ± 1.0 | 35.4 ± 0.9 | 21.9 ± 0.9 |
| IW-OPD | 60.7 ± 0.7 | 43.4 ± 0.9 | 36.0 ± 0.8 | 22.6 ± 0.8 |
| Best ref. | 61.4 ± 0.7 | 44.3 ± 0.9 | 36.9 ± 0.8 | 23.2 ± 0.8 |
| PCGR | 63.0 ± 0.6 | 46.1 ± 0.8 | 38.4 ± 0.7 | 24.8 ± 0.7 |

## H CROSS-FAMILY RETENTION

RR measures regressions within the trained family. To assess broader forgetting, we evaluate every single-task checkpoint on the three core families it was not trained on, although PCGR's reference contains only responses from the trained family (Table 6). OPD loses 0.98 points on untrained families and norm-matched OPD still loses 0.85, so the forgetting is not a step-size effect. PCGR is neutral ($-0.05$), and for every training family its change lies within one seed standard deviation of zero: constraining within-family conflict also protects the other families.

Table 6: Cross-family retention (Qwen3-1.7B): change in the average score on the three untrained families, relative to the initial student, by training family; mean ± std over three seeds.

| | Trained on | | | | |
|---|---|---|---|---|---|
| Method | Mathematics | Code | Instructions | Tool calling | Average |
| OPD | $-0.8 \pm 0.4$ | $-1.2 \pm 0.5$ | $-0.9 \pm 0.4$ | $-1.0 \pm 0.5$ | $-0.98 \pm 0.30$ |
| OPD (norm-matched) | $-0.7 \pm 0.4$ | $-1.0 \pm 0.5$ | $-0.8 \pm 0.4$ | $-0.9 \pm 0.5$ | $-0.85 \pm 0.30$ |
| IW-OPD | $-0.5 \pm 0.4$ | $-0.8 \pm 0.4$ | $-0.6 \pm 0.4$ | $-0.7 \pm 0.4$ | $-0.65 \pm 0.28$ |
| Best ref. | $-0.3 \pm 0.3$ | $-0.6 \pm 0.4$ | $-0.4 \pm 0.4$ | $-0.5 \pm 0.4$ | $-0.45 \pm 0.26$ |
| PCGR | $+0.1 \pm 0.3$ | $-0.2 \pm 0.4$ | $0.0 \pm 0.3$ | $-0.1 \pm 0.4$ | $-0.05 \pm 0.24$ |

# I ADDITIONAL RESULTS

Table 7 gives the per-family regression and gain rates behind Figure 3, and Table 8 the per-benchmark ablation results behind Figure 3c.

Table 7: Regression rate (RR, ↓) and gain rate (GR, ↑) per family for Qwen3-1.7B, and mean changes on initially solved ($\Delta_{\text{old}}$) and unsolved ($\Delta_{\text{new}}$) problems in points, averaged over families. Re-evaluation: the initial student resampled (noise floor). The top four methods per column (lowest RR, highest GR and $\Delta$) are shaded from dark to light orange.

| | Mathematics | | Code | | Instructions | | Tool calling | | Average | |
|---|---|---|---|---|---|---|---|---|---|---|
| Method | RR | GR | RR | GR | RR | GR | RR | GR | $\Delta_{\text{old}}$ | $\Delta_{\text{new}}$ |
| Re-evaluation | 1.7 ± 0.5 | 10.0 ± 1.5 | 3.2 ± 0.8 | 4.3 ± 0.9 | 3.0 ± 0.7 | 4.4 ± 0.9 | 3.5 ± 0.8 | 3.8 ± 0.8 | 0.0 ± 0.2 | 0.1 ± 0.2 |
| OPD | 8.0 ± 0.9 | 21.7 ± 1.6 | 16.1 ± 1.3 | 18.8 ± 1.5 | 14.2 ± 1.2 | 20.5 ± 1.6 | 17.4 ± 1.4 | 17.9 ± 1.5 | −0.8 ± 0.3 | 5.3 ± 0.6 |
| OPD (norm-matched) | 7.4 ± 0.9 | 20.9 ± 1.6 | 15.0 ± 1.3 | 18.0 ± 1.5 | 13.2 ± 1.2 | 19.6 ± 1.6 | 16.2 ± 1.4 | 17.1 ± 1.5 | −0.7 ± 0.3 | 5.1 ± 0.6 |
| IW-OPD | 6.6 ± 0.8 | 18.8 ± 1.5 | 12.9 ± 1.2 | 16.4 ± 1.4 | 11.5 ± 1.1 | 17.7 ± 1.5 | 13.8 ± 1.3 | 15.6 ± 1.4 | −0.5 ± 0.3 | 6.2 ± 0.6 |
| OPD+KL(init) | 5.5 ± 0.7 | 17.9 ± 1.5 | 10.5 ± 1.1 | 16.4 ± 1.4 | 9.5 ± 1.0 | 17.0 ± 1.4 | 11.0 ± 1.2 | 15.6 ± 1.3 | −0.2 ± 0.3 | 6.0 ± 0.6 |
| OPD+$\alpha R$ | 6.3 ± 0.8 | 20.0 ± 1.5 | 11.8 ± 1.2 | 17.5 ± 1.4 | 10.6 ± 1.1 | 18.4 ± 1.5 | 12.4 ± 1.3 | 16.7 ± 1.4 | −0.4 ± 0.3 | 6.7 ± 0.6 |
| UGS | 5.8 ± 0.7 | 18.3 ± 1.4 | 11.0 ± 1.1 | 16.8 ± 1.4 | 9.8 ± 1.0 | 17.5 ± 1.4 | 11.5 ± 1.2 | 15.8 ± 1.3 | −0.3 ± 0.3 | 6.9 ± 0.6 |
| A-GEM-style | 6.1 ± 0.8 | 19.2 ± 1.5 | 11.4 ± 1.2 | 17.1 ± 1.4 | 10.1 ± 1.1 | 18.0 ± 1.5 | 11.9 ± 1.3 | 16.1 ± 1.4 | −0.4 ± 0.3 | 6.6 ± 0.6 |
| Blockwise projection | 5.5 ± 0.7 | 20.8 ± 1.5 | 10.1 ± 1.1 | 18.1 ± 1.4 | 9.2 ± 1.0 | 19.3 ± 1.5 | 10.8 ± 1.2 | 17.1 ± 1.4 | −0.2 ± 0.3 | 7.6 ± 0.6 |
| PCGR | 5.0 ± 0.7 | 22.5 ± 1.6 | 8.7 ± 1.0 | 19.8 ± 1.4 | 7.6 ± 0.9 | 21.2 ± 1.5 | 9.4 ± 1.1 | 18.8 ± 1.4 | +0.1 ± 0.3 | 10.3 ± 0.7 |
| PCGR+IW | **4.8** ± 0.7 | **22.9** ± 1.6 | **8.1** ± 1.0 | **20.0** ± 1.5 | **7.2** ± 0.9 | **21.6** ± 1.5 | **8.8** ± 1.1 | **19.2** ± 1.4 | **+0.2** ± 0.3 | **11.2** ± 0.7 |

Table 8: Ablations with Qwen3-1.7B on mathematics and instruction following, mean ± std over three seeds. AIME averages AIME24/25; $\Delta$: change in average relative to full PCGR.

| | Variant | MATH-500 | AIME | IFEval | IFBench | Average | $\Delta$ |
|---|---|---|---|---|---|---|---|
| | PCGR (full) | 92.7 ± 0.3 | 50.5 ± 1.7 | 76.5 ± 0.5 | 41.5 ± 0.8 | 65.30 ± 0.49 | – |
| Prefix info | (a) Constant budget, $u$-grouping | 92.3 ± 0.4 | 49.5 ± 1.8 | 75.7 ± 0.6 | 40.7 ± 0.9 | 64.55 ± 0.50 | −0.75 |
| | (b) $K = 1$, constant budget | 92.0 ± 0.5 | 48.7 ± 1.9 | 75.1 ± 0.6 | 40.0 ± 1.0 | 63.95 ± 0.52 | −1.35 |
| | (c) Position grouping and budget | 92.1 ± 0.4 | 49.1 ± 1.8 | 75.4 ± 0.7 | 40.3 ± 0.9 | 64.23 ± 0.50 | −1.08 |
| | (d) Shuffled discrepancy | 92.0 ± 0.5 | 49.0 ± 2.0 | 75.3 ± 0.7 | 40.1 ± 1.0 | 64.10 ± 0.53 | −1.20 |
| Location | (e) No blockwise routing (UGS) | 91.9 ± 0.4 | 48.4 ± 1.8 | 74.7 ± 0.6 | 39.7 ± 0.9 | 63.68 ± 0.48 | −1.63 |
| | (f) Shuffled block gates | 91.5 ± 0.5 | 48.0 ± 1.9 | 74.3 ± 0.7 | 39.2 ± 1.1 | 63.25 ± 0.52 | −2.05 |
| Reference | (g) Unverified reference | 92.0 ± 0.5 | 48.9 ± 1.8 | 75.2 ± 0.6 | 40.0 ± 1.0 | 64.03 ± 0.51 | −1.28 |
| | (h) Fixed reference pool | 92.3 ± 0.4 | 49.6 ± 1.7 | 75.8 ± 0.5 | 40.6 ± 0.9 | 64.58 ± 0.48 | −0.73 |
| Statistic | (i) Length-normalized $D$ | 92.9 ± 0.3 | 50.7 ± 1.7 | 76.6 ± 0.5 | 41.5 ± 0.8 | 65.43 ± 0.47 | +0.13 |
| | (j) Signed-sum $D$ | 92.2 ± 0.4 | 49.2 ± 1.8 | 75.5 ± 0.6 | 40.4 ± 0.9 | 64.33 ± 0.50 | −0.98 |

Figure 5 summarizes the macro gain of PCGR over OPD beyond the main setting, and Tables 9 and 10 give the per-family results for the 4B, mixed-task, and AdamW runs. Table 11 and Figure 6 give the training cost and learning curves. Cost accounting includes rollout generation, teacher scoring, calibration, reference passes, routing, communication, and verifier time. Because on-policy response lengths differ across methods, the same 400 updates yield different rollout-token totals, so the curves are compared on the common covered range. At equal rollout tokens PCGR starts slightly behind and ends highest; at equal GPU-hours it overtakes the baselines between 340 and 510 GPU-hours.

# J ADDITIONAL EXPERIMENTAL DETAILS

**Task resources.** Instruction training uses IF-RLVR resources associated with IFBench while excluding IFBench OOD constraints (Pyatkin et al., 2026). Procedural tasks use a frozen selection of Reasoning Gym generators (Stojanovski et al., 2026), and SQL uses an audited BIRD training split (Li et al., 2023). The remaining resources follow their released data and evaluation protocols.

**Training protocol.** Prompt and response limits are 8,192 and 16,384 tokens; oversized prompts are excluded rather than truncating schemas. Training samples at temperature one without truncation and takes exactly one update per rollout batch. We use BF16 model arithmetic and FP32 routing statistics. Mixed-task runs use homogeneous batches, and within each seed, methods share prompt

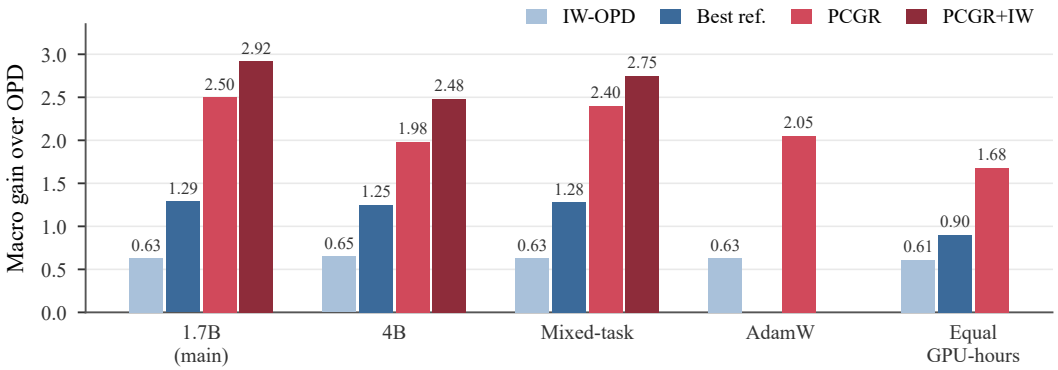


Figure 5: Macro gain over OPD beyond the main setting: Qwen3-1.7B (main), Qwen3-4B, mixed-task training, AdamW, and the GPU-hour budget of OPD. Best ref.: blockwise projection; missing bars were not run.

Table 9: Per-family results at 4B (single-task) and under mixed-task training (1.7B), mean ± std over three seeds.

| Method | Mathematics | Code | Instructions | Tool calling | Macro |
|---|---|---|---|---|---|
| *Qwen3-4B, single-task* | | | | | |
| Initial student | 77.80 | 68.40 | 64.80 | 62.10 | 68.28 |
| Teacher (Qwen3-30B-A3B) | 81.53 | 80.70 | 68.70 | 69.50 | 75.11 |
| OPD | 79.30 ± 1.2 | 70.30 ± 0.5 | 66.10 ± 0.5 | 63.90 ± 0.6 | 69.90 ± 0.36 |
| IW-OPD | 80.40 ± 1.0 | 70.60 ± 0.4 | 67.20 ± 0.5 | 64.00 ± 0.5 | 70.55 ± 0.33 |
| Best ref. | 80.70 ± 1.1 | 71.50 ± 0.5 | 67.50 ± 0.4 | 64.90 ± 0.5 | 71.15 ± 0.34 |
| PCGR | 81.60 ± 0.9 | 71.90 ± 0.4 | 68.60 ± 0.4 | 65.40 ± 0.5 | 71.88 ± 0.31 |
| PCGR+IW | 82.10 ± 1.0 | 72.40 ± 0.5 | 69.00 ± 0.4 | 66.00 ± 0.5 | 72.38 ± 0.32 |
| *Qwen3-1.7B, mixed-task* | | | | | |
| Initial student | 59.00 | 56.50 | 53.75 | 52.00 | 55.31 |
| OPD | 61.23 ± 1.3 | 58.10 ± 0.6 | 55.25 ± 0.5 | 53.60 ± 0.6 | 57.05 ± 0.40 |
| IW-OPD | 61.80 ± 1.1 | 58.65 ± 0.5 | 56.00 ± 0.5 | 54.25 ± 0.6 | 57.68 ± 0.38 |
| Best ref. | 62.43 ± 1.2 | 59.30 ± 0.5 | 56.70 ± 0.5 | 54.90 ± 0.5 | 58.33 ± 0.36 |
| PCGR | 63.53 ± 1.0 | 60.35 ± 0.5 | 58.00 ± 0.4 | 55.92 ± 0.5 | 59.45 ± 0.34 |
| PCGR+IW | 63.95 ± 1.1 | 60.70 ± 0.5 | 58.30 ± 0.4 | 56.25 ± 0.5 | 59.80 ± 0.35 |

and task schedules but generate their own responses. All settings are fixed before final evaluation, and final benchmarks are never used for checkpoint selection. "Best ref." is the strongest reference-aware baseline on the development split (blockwise projection), fixed before the 4B, mixed-task, and extended runs.

**Evaluation protocol.** Completions that hit the token cap without a parseable answer are scored as failures; truncation stays under 2% on most benchmarks and reaches about 7% on AIME, and we log it per method. Structured tasks also report native strict metrics (prompt-level instruction satisfaction, SQL execution accuracy, BFCL AST and execution). Bootstrap clusters are problems, databases for SQL, and shared prompt, generator, or API families for other structured tasks; repeated completions are never treated as independent problems.

**Training data.** Training resources are the sources of Table 13, partitioned approximately 90/5/5 into optimization, development, and diagnostics and grouped by problem family, instruction template, database, or API family; Reasoning Gym data are drawn from a frozen set of generators, so only the manifest of held-out instances is fixed.

**Data and verifier controls.** Training uses prompts and outcome metadata rather than provided reasoning traces as extra SFT targets. We remove unusable examples and screen training resources against final test sets using source identifiers, normalized matching, and near-duplicate review. BIRD dev is reserved for final evaluation. Procedural instance manifests use disjoint seeds and deduplicated outputs; unseen instances, difficulty shifts, and held-out generators form separate tests. Ref-

Table 10: Optimizer dependence (Qwen3-1.7B, family averages, mean ± std over three seeds). Corr.: Pearson correlation over steps between the realized reference effect $r^\top \Delta\theta_k$ and the routed prediction $-\eta_k r^\top P^k \widetilde{G}$.

| Optimizer | Method | Math | Code | Instr. | Tool | Macro | Corr. |
|---|---|---|---|---|---|---|---|
| RMS | OPD | 62.07 ± 1.2 | 59.00 ± 0.5 | 56.25 ± 0.5 | 54.50 ± 0.6 | 57.95 ± 0.42 | – |
| | IW-OPD | 62.63 ± 1.0 | 59.55 ± 0.5 | 57.00 ± 0.5 | 55.15 ± 0.5 | 58.58 ± 0.37 | – |
| | PCGR | 64.57 ± 0.9 | 61.25 ± 0.4 | 59.00 ± 0.5 | 57.00 ± 0.5 | 60.45 ± 0.33 | 1.00 |
| AdamW | OPD | 62.20 ± 1.1 | 59.10 ± 0.5 | 56.35 ± 0.5 | 54.60 ± 0.6 | 58.06 ± 0.41 | – |
| | IW-OPD | 62.75 ± 1.0 | 59.65 ± 0.5 | 57.10 ± 0.5 | 55.25 ± 0.5 | 58.69 ± 0.38 | – |
| | PCGR | 64.20 ± 0.9 | 60.90 ± 0.5 | 58.65 ± 0.5 | 56.70 ± 0.5 | 60.11 ± 0.35 | 0.80 |

Table 11: Training cost for Qwen3-1.7B, averaged over steps and seeds; total GPU-hours aggregate the four families for one seed. $K = 4$ was run on mathematics only (–). At 4B the per-step overhead of PCGR is also about $1.20\times$.

| Method | Sweeps | Time/step (s) | Memory (GB) | Total GPU-h | Macro @ equal GPU-h |
|---|---|---|---|---|---|
| OPD | 1 | 185.9 | 68.6 | 675.8 | 57.95 |
| IW-OPD | 1 | 188.7 | 69.5 | 686.2 | 58.56 |
| A-GEM-style | 3 | 214.3 | 72.4 | 778.9 | 58.42 |
| Blockwise projection | 3 | 218.6 | 73.6 | 789.6 | 58.85 |
| PCGR ($K = 2$) | 3 | 223.4 | 74.1 | 808.4 | 59.63 |
| PCGR ($K = 4$) | 5 | 262.1 | 74.6 | – | – |

erence admission requires complete success under the declared predicate: SQL verification checks execution-result equivalence on the designated database after an annotation audit, and tool calls must match accepted functions and typed arguments, including required order, with execution additionally checked when supported. Merely parsing or executing without exception is insufficient, and ambiguous items are excluded. Fixed parsers extract final outputs without repair, while the OPD loss covers all generated response tokens. No final-test predicate or example is used to admit training references.

**Benchmark versions and sizes.** We pin dataset revisions and report evaluation sizes. Mathematics uses MATH-500 (500) and AIME 2024/2025 (30 each), with a fixed DeepMath revision for training. Code uses HumanEval+ (164) and EvalPlus MBPP+ v0.2.0 (378). Instructions use IFEval (541) and the IFBench out-of-distribution split. Tool calling uses a fixed 500-prompt subset of the BFCL single-turn categories, with AST and executable checks evaluated on the same items.

**Training budget and optimization.** Each single-task run performs 400 optimization updates of 64 prompts with one response each (25,600 rollouts per run); the non-padding generated-token total is logged per method and used for the equal-token curves of Figure 6. The reference buffer holds $C = 256$ verified responses per task and each update samples $M \leq 8$ of them. The controlled RMS implementation uses $\nu = 0.99$ and $\epsilon = 10^{-8}$, with $v^0$ initialized to the second moment $G^0 \odot G^0$ of the unrouted first-batch gradient, because $P^0$ is needed before the first routing step, so the preconditioner tracks gradient scale from the first step. Because the RMS preconditioner normalizes each coordinate by its running root-mean-square, $\eta_k$ is an Adam-scale step size. Learning rates are selected from $\{10^{-6}, 10^{-5}, 10^{-4}\}$ on development data with a constant rate after 5% linear warm-up: $10^{-4}$ for mathematics and code and $10^{-5}$ for instruction following and tool calling; the AdamW runs (E9) use the same rates. The grid was searched independently for every method, and smaller rates were worse for every method including OPD.

**Hardware and cost.** Each run uses 8 H100-80GB GPUs, with the student trained under fully sharded data parallelism in BF16 and the frozen teacher served for scoring. The total GPU-hours of Table 11 aggregate the four core families for a single seed; per-run non-padding token totals and the 4B costs are logged rather than derived, and the $\approx 1.20\times$ PCGR-over-OPD per-step overhead also holds at 4B. The AdamW runs (E9) use moment coefficients $(0.9, 0.95)$, $\epsilon = 10^{-8}$, no weight decay, and gradient clipping at global norm 1.0; both optimizers keep master weights and optimizer states in FP32 while model arithmetic is BF16, so the small routed updates accumulate at full precision.

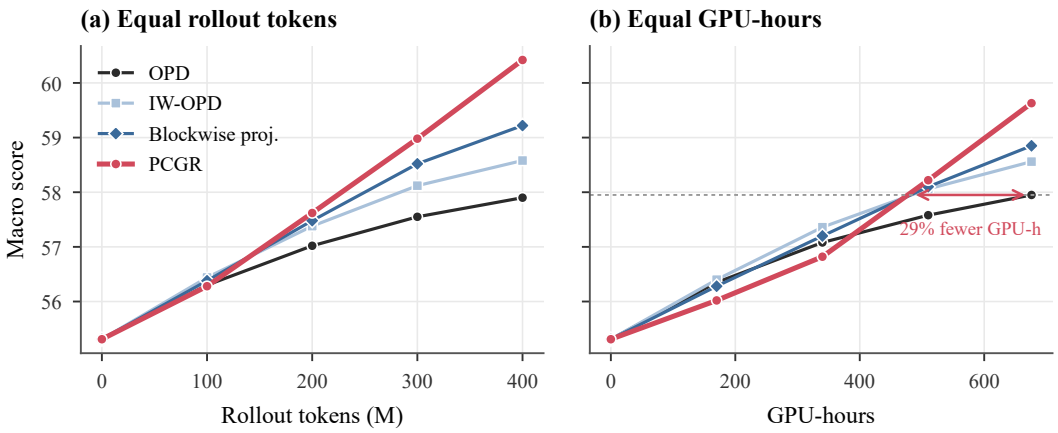


Figure 6: Learning curves for Qwen3-1.7B, mean over three seeds (seed std 0.3–0.5). PCGR reaches the final OPD score with about 29% fewer GPU-hours.

Table 12: Experimental matrix. "Best ref." denotes the strongest reference-aware baseline on the E1 development split (blockwise projection).

| ID | Question | Student | Tasks | Methods | Seeds |
|---|---|---|---|---|---|
| E1 | RQ2, RQ3 | 1.7B | Core, single-task | All methods in Section 4 | 3 |
| E2 | RQ2 | 4B | Core, single-task | OPD, IW-OPD, Best ref., PCGR, PCGR+IW | 3 |
| E3 | RQ2 | 1.7B | Core, mixed-task | Same as E2 | 3 |
| E4 | RQ4 | 1.7B | Math, Instructions | Ablations in Section 5.4 | 3 |
| E5 | RQ1 | 1.7B | Core | Diagnostics D1–D3 on frozen E1 OPD snapshots (no additional training); D4 logged from E1 and E9 PCGR runs | – |
| E6 | RQ4 | 1.7B | Math | Sensitivity grid (Appendix F) | 2 |
| E7 | RQ4 | 1.7B, 4B | Core | Cost accounting from E1 and E2 | – |
| E8 | RQ2 | 1.7B | Extended, single-task | OPD, IW-OPD, Best ref., PCGR | 3 |
| E9 | RQ2 | 1.7B | Core, single-task | OPD, IW-OPD, PCGR with AdamW | 3 |
| E10 | RQ3 | 1.7B | Core | Cross-family evaluation of E1 checkpoints (evaluation only) | – |
| E11 | RQ3 | 1.7B | Math | Scalar frontier: UGS with constant $\bar{\beta} \in \{0.25, 0.5, 0.75\}$ | 3 |

Table 13: Task suite. Only training resources are used to build references; evaluation resources never enter optimization or tuning.

| Task | Training | Evaluation | Success predicate |
|---|---|---|---|
| *Core suite* | | | |
| Mathematics | DeepMath | MATH-500; AIME24/25 | Answer equivalence |
| Code | Eurus-2-RL (code) | HumanEval+; MBPP+ | Pass all tests |
| Instructions | IF-RLVR | IFEval; IFBench OOD | All prompt constraints |
| Tool calling | Verifiable xLAM/APIGen | BFCL single-turn subsets | Function and argument correctness |
| *Extended suite* | | | |
| Logic / algorithms | Reasoning Gym | Unseen instances and shifts | Algorithmic full credit |
| SQL | Audited BIRD train | BIRD dev | Execution-result match |

**Diagnostics and reporting.** The frozen-snapshot diagnostics use the held-out prompts and singleton-token subsample of Appendix E; D3 uses 256 prompts and 256 held-out verified responses per family, and D4 logs every optimization step. Computing the per-block quantities

$c_{i,t,b} = r_b^\top P_b g_{i,t,b}$ and $E_{i,t,b} = g_{i,t,b}^\top P_b g_{i,t,b}$ requires per-token parameter gradients, which we obtain with micro-batched backward passes over the subsampled diagnostic tokens; this offline analysis is run once per snapshot and its cost is separate from the training budget of Table 11. We report $\kappa$ at three granularities—singleton tokens, per-response $K = 2$ groups, and full training-batch $K = 2$ groups; the batch-level group $\kappa$ is the quantity that governs training in Proposition 1 and stays below one on every family (pooled median 0.56). Paired cluster-bootstrap intervals use $10{,}000$ resamples over problem clusters within each seed.