B-MOPD: DATA AND GRADIENT–GAP BALANCING FOR MULTI-TEACHER ON-POLICY DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Multi-teacher on-policy distillation trains one student from several specialized teachers on responses generated by the student itself. A natural mixture of training data can give domains unequal exposure, while equal loss weights can still produce unequal learning signals. We introduce B-MOPD, a two-stage allocation method that first balances input-token exposure across domains, then adjusts teacher loss weights using gradient strength and the current teacher–student distributional gap. Gradient normalization compensates for unequal signal magnitudes; inverse-gap weighting gives greater relative emphasis to teachers closer to the student. A causal controller estimates both quantities from completed training updates, smooths the estimates, and bounds the weight ratios. Across mathematics, code, and knowledge, B-MOPD achieves macro accuracies of 46.84% and 72.70% for Qwen2.5-0.5B-Instruct and Qwen2.5-1.5B-Instruct on a 753-question diagnostic evaluation. Natural-data uniform baselines obtain 42.79% and 71.50%, respectively. Component comparisons favor the combined gradient–inverse rule over strength-only and gradient–direct weighting in both configurations. We further examine teacher-relative recovery and per-domain trade-offs to characterize the resulting allocation.

1 INTRODUCTION

On-policy distillation trains a student on its own generated trajectories while obtaining learning signals from a teacher (Agarwal et al., 2024). With multiple domain teachers, the student must integrate specialized capabilities rather than imitate a single distribution. A central difficulty is that equal treatment has several incompatible meanings. Equal numbers of prompts do not imply equal numbers of input tokens. Equal input-token exposure does not imply equal gradient magnitudes. Equal loss weights need not imply equal progress toward teachers with different distributions.

We introduce B-MOPD to address allocation in two stages. Starting from natural domain proportions and uniform loss weights, we first approximately balance cumulative raw input tokens across domains. This preprocessing step is part of the complete method. We then correct residual differences in training signals using two quantities: the norm of each domain's unweighted mean gradient and the current conditional reverse KL on student-generated prefixes. Our gradient–inverse controller, G–INV, compensates for planned sampling exposure and gradient magnitude, then gives greater relative weight to smaller gaps. Here, a gap measures disagreement between teacher and student token distributions, not a difference in task accuracy. The complete B-MOPD recipe is input-token balance followed by this controller.

The central distinction is between *how often a teacher is consulted* and *how strongly its signal enters an update*. Prompt length and gradient magnitude need not vary together. Separating the schedule from the controller makes both choices explicit.

We evaluate Qwen2.5-0.5B-Instruct with three 1.5B teachers and Qwen2.5-1.5B-Instruct with three 7B teachers. The natural-data uniform baseline is the starting point for the complete recipe. At a fixed 128-update horizon, B-MOPD improves its three-domain macro accuracy by 4.05 and 1.20 percentage points. We separately examine input balance, the gradient and gap factors, teacher-relative recovery, and realized controller behavior. Our contributions are:

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.

- A two-stage MOPD allocation rule that balances input exposure and then accounts for gradient strength and teacher–student gap.
- Component comparisons across two student–teacher scales, reporting domain-level accuracy as well as the equally weighted domain average.
- An analysis of bounded online allocation that distinguishes refreshed estimates from actual changes in applied teacher weights.

2 Related Work

On-policy and multi-teacher distillation. Generalized knowledge distillation trains on student-generated outputs to address the distributional mismatch encountered by the student (Agarwal et al., 2024). Concurrent work studies token balancing and gap-aware prioritization for multi-teacher distillation (Gao et al., 2026). We study unweighted domain-gradient norms together with current conditional KL, comparing inverse and direct gap directions under a fixed data schedule. We do not claim a head-to-head comparison with that concurrent system.

Gradient-based task balancing. GradNorm adapts multitask losses using gradient magnitudes and relative training rates (Chen et al., 2018). Our controller instead uses on-policy trajectories, all-parameter domain gradients, current teacher KL, and a closed-form clipped rule rather than GradNorm's auxiliary weight objective. Gradient norms are not interpreted as functional importance or actual Adam step sizes.

Gradient surgery projects conflicting task gradients (Yu et al., 2020). Our rule rescales domain contributions without changing their directions: it addresses scalar allocation, not gradient-conflict removal.

Allocation versus teacher ensembling. Our setting routes each prompt to its domain specialist. The allocation problem is therefore how to combine learning signals across examples, rather than how to merge several teachers' token probabilities for one example. This distinction connects multi-teacher distillation to task balancing while preserving the on-policy source of the training trajectories. Data scheduling and loss weighting operate at different stages of this routed process.

3 Gradient–Gap Allocation

B-MOPD first sets a domain schedule from prompt lengths, then weights the resulting learning signals. The current student answers each prompt, and its domain teacher evaluates that answer. The controller requires neither teacher-generated targets nor reward-based response selection.

3.1 Learning signal and input schedule

Let p_θ be the student and q_d the teacher for domain $d \in \{1, 2, 3\}$. For a prompt x in domain d, the student samples one response y. Reverse KL is an established distillation divergence (Gu et al., 2024); our allocation rule uses its conditional value on visited prefixes. At prefix $s_t = (x, y_{<t})$, define

$$D(s_t) = \sum_a p_\theta(a \mid s_t) \log \frac{p_\theta(a \mid s_t)}{q_d(a \mid s_t)}. \tag{1}$$

Probabilities are normalized over the verified common token-ID vocabulary, excluding padded output-head entries. We average this full-vocabulary statistic over response positions within each sequence for controller estimation. The optimizer instead uses the sampled-action surrogate

$$\widehat{\ell}_d(x, y) = \frac{1}{|y|} \sum_t \mathrm{sg}[\log p_\theta(y_t \mid s_t) - \log q_d(y_t \mid s_t)] \log p_\theta(y_t \mid s_t). \tag{2}$$

Equation 2 is a local surrogate, not a KL value or an exact trajectory gradient. At a fixed prefix, the unnormalized token term has the expected conditional-KL gradient. Random prefixes and response-length normalization prevent extending that identity directly to the full sequence objective.

Stage 1: balance input exposure. Each domain has a fixed prompt queue. Let C_d be the cumulative number of raw input tokens assigned to domain d, initially zero. The balanced schedule (B) repeatedly selects a domain with the smallest C_d, takes its next prompt, and adds that prompt's token count to C_d. Ties use a fixed domain order and queues wrap when exhausted. Raw input tokens are the tokenization of the user prompt without special tokens; they exclude the system message and chat template. This greedy schedule approximately equalizes input exposure without padding or dropping a prompt. It does not equalize generated tokens or prompt counts.

The natural schedule (N) uses the same queues but selects the domain furthest below its cumulative target prompt count, proportional to the eligible pool sizes. N–U uses this schedule with all teacher coefficients equal to one and is our baseline for the complete method. Both schedules are frozen before training. Each prompt occurrence produces one fresh student response; 3,072 occurrences therefore do not imply 3,072 distinct questions or multiple candidate responses selected by a reward.

Why balance inputs rather than count examples? Input tokens provide a common exposure unit when prompt lengths differ across domains. Equal generated-token allocation would depend on the changing student policy and its termination behavior. Our input schedule avoids that coupling: it is constructed before training and shared across weighting conditions. Stage 2 accounts for the strength of the resulting signals.

This ordering also separates two sources of imbalance. A domain with a large prompt pool can dominate a natural mixture even when each of its examples is short. Conversely, assigning equal numbers of examples can overrepresent the input material from domains with longer prompts. The token budget addresses the latter unit of exposure without requiring equal prompt counts. It does not claim that a token carries the same amount of useful information in every domain: that remaining variation motivates the gradient measurement in Stage 2. Section 5.3 tests the data step directly by holding Uniform weighting fixed, rather than attributing the full method's gain to preprocessing alone.

3.2 Stage 2: separate exposure, strength, and gap

For update k, let $B_{d,k}$ contain the $n_{d,k}$ generated sequences assigned to domain d. For each present domain ($n_{d,k} > 0$), we compute its unweighted mean gradient

$$g_{d,k} = \nabla_\theta \left(\frac{1}{n_{d,k}} \sum_{(x,y)\in B_{d,k}} \widehat{\ell}_d(x,y) \right), \qquad G_{d,k} = \|g_{d,k}\|_2. \tag{3}$$

The norm is taken before teacher weighting. Absent domains contribute zero to the combined gradient and supply no measurement to the controller window. Domain gradients are combined over present domains using the actual batch fractions,

$$g_k = \sum_{d:n_{d,k}>0} \frac{n_{d,k}}{\sum_j n_{j,k}} w_{d,k}\, g_{d,k}. \tag{4}$$

Let π_d denote the planned prompt share over the frozen schedule. The controller compensates for this planned exposure, not an assumption that every batch contains equal domain counts.

Measuring strength before applying weights. The domain mean is formed before taking its norm: $\|n_{d,k}^{-1} \sum_i \nabla \widehat{\ell}_i\|_2$ is generally different from the mean of the individual gradient norms. The former measures the signal that the domain actually supplies to the combination step, including cancellation among its examples. Measuring this unweighted signal also prevents a previously large teacher coefficient from mechanically inflating the next strength estimate. Each norm spans the trainable student parameters; teacher parameters remain fixed. The response-token mean in Eq. 2 gives each sequence its own normalization before domain averaging, rather than letting a longer response automatically receive more weight through a concatenated token sum.

Every eight completed updates, we aggregate the preceding window. G_d is the mean of the available unweighted route-gradient norms; D_d is the prompt-weighted mean of sequence-average full conditional KL. These are different averaging operations. A window is admitted only when every domain has at least four gradient measurements, at least eight prompt occurrences, and positive statistics

above the implementation thresholds. Otherwise all coefficients are held. Admitted statistics are smoothed with an exponential moving average, using 0.8 for the previous estimate and 0.2 for the new estimate. The first admitted window initializes the estimate.

For exponent α, the ideal unnormalized allocation is

$$v_d = \frac{\bar{D}_d^\alpha}{\pi_d \bar{G}_d}. \tag{5}$$

We evaluate strength-only ($\alpha = 0$), strength+Direct ($\alpha = 1$), and strength+Inverse ($\alpha = -1$). The last is G–INV. Gradient-engine Uniform sets all coefficients to one within the same numerical engine.

Why include the gap after strength normalization? Gradient magnitude describes the size of a local learning signal; the gap describes teacher–student disagreement at visited prefixes. A larger gap need not imply a proportionally useful direction. Inverse weighting favors closer teachers after exposure and strength correction; Direct tests the opposite preference using the same statistics and update rule.

What is being balanced? The controller targets a proxy for each domain's weighted gradient magnitude, not equal accuracy or equal compute. Ignoring clipping and estimation lag, the planned contribution satisfies $\pi_d v_d G_d \propto \bar{D}_d^\alpha$. Strength-only equalizes this proxy; Inverse tilts it toward smaller distributional gaps, and Direct toward larger gaps. This separates exposure correction from a curriculum choice. It is not an optimizer-step guarantee: directions can cancel, current shares differ from planned shares, and Adam changes the update geometry. Likewise, input balance controls prompt exposure, not the policy-dependent cost or learning signal of generated tokens.

3.3 CLIPPING, CAUSALITY, AND INTERPRETATION

The applied rule operates in log space. With $z_d = \log v_d$, $c = (\max_d z_d + \min_d z_d)/2$, and $h = \log(4)/2$, set

$$r_d = \exp(\mathrm{clip}(z_d - c, -h, h)), \qquad w_d = \frac{r_d}{\sum_j \pi_j r_j}. \tag{6}$$

Thus the largest-to-smallest coefficient ratio is at most four and the planned-share-weighted mean coefficient is one. The controller uses only completed training updates; a new coefficient vector applies on the next update. The first eight updates use uniform coefficients. No evaluation answers or scores enter the controller.

Inverse favors smaller gaps; Direct favors larger gaps. Either can be reasonable under different teacher quality and student capacity constraints. Clipping is a substantive part of the method, not merely an implementation safeguard: it can make substantially different estimated ratios yield identical applied weights. We therefore record both estimated statistics and actual coefficient changes.

Training-time information flow. The input schedule is constructed once from prompt lengths and reused across weighting rules. During training, the controller aggregates gradient and KL measurements associated with the generated trajectories; its decisions take effect only after a window closes. It does not evaluate intermediate checkpoints to choose teacher weights. The schedule fixes exposure, while the past-window estimates determine the next coefficients. Consequently, applying the recipe to a new teacher set requires training prompts and teacher probabilities, but not correctness labels for the student's generated answers. Evaluation-time accuracy remains separate from the quantities that drive allocation.

Two-stage gradient–gap allocation
Before training: construct the raw-input-token-balanced prompt schedule; compute its domain prompt shares π_d; initialize $w_d = 1$.
For update $k = 1, \ldots, 128$**:**

1. Sample one response per scheduled prompt from the current student.
2. Compute unweighted mean domain gradients $g_{d,k}$ and sequence-average full conditional KL. Record $G_{d,k} = \|g_{d,k}\|_2$ before weighting.
3. Combine gradients using the current w_d and actual batch fractions (Eq. 4); clip the combined gradient and take one AdamW update.
4. Every eight completed updates, check per-domain coverage and positive-statistic gates. If admitted, update the EMA estimates and compute weights using Eqs. 5–6. Otherwise retain the weights. Apply any new vector only at update $k + 1$.

Evaluation labels and scores are never inputs to these steps.

Figure 1: The complete method. Data allocation is fixed before optimization; only loss coefficients are controlled online. Setting $\alpha = -1$ gives G–INV.

4 EXPERIMENTAL PROTOCOL

Models. We use the Qwen2.5 family and its specialist variants (Qwen Team, 2024; Yang et al., 2024; Hui et al., 2024); Table 1 gives the two configurations. The general instruction teacher supervises ARC knowledge questions, not instruction-following evaluation. Initial students are pretrained instruction-tuned models with no additional distillation in this study, not randomly initialized models. Both configurations route a training example to the teacher for its domain, rather than ensembling all teachers at every token.

Table 1: Student and domain-routed teacher models. Each column is a separate MOPD configuration. All model names denote the public instruction-tuned checkpoints.

Role	0.5B-student configuration	1.5B-student configuration
Student	Qwen2.5-0.5B-Instruct	Qwen2.5-1.5B-Instruct
Math teacher	Qwen2.5-Math-1.5B-Instruct	Qwen2.5-Math-7B-Instruct
Code teacher	Qwen2.5-Coder-1.5B-Instruct	Qwen2.5-Coder-7B-Instruct
Knowledge teacher	Qwen2.5-1.5B-Instruct	Qwen2.5-7B-Instruct

Training. The primary gradient comparison runs 128 optimizer updates with 24 prompt occurrences per update: 3,072 fresh responses per arm. The training sampling seed is 20260925, with matched per-occurrence random seeds within each configuration; diverged policies need not generate identical tokens from these seeds. Sampling uses temperature 1, top-p 1, and unrestricted top-k. Math respects its teacher's native 4,096-token context, subtracting the rendered prompt and a one-token reserve; Code permits 16,384 response tokens and ARC 4,096. These are caps, not fixed response lengths. We use FP32 master weights with BF16 forward/backward computation and AdamW (Appendix B). Comparisons match update and prompt-occurrence budgets, not generated tokens or compute; Appendix C reports the realized token counts. Each configuration has a same-engine uniform control because their gradient accumulation implementations differ.

Evaluation and scope. We compare frozen 128-update checkpoints using a common SGLang 0.5.9 backend, greedy decoding, identical rendered student prompts, stop rules, and per-domain generation budgets. The fixed evaluation panel contains 753 questions: 256 sampled from the GSM8K test split (Cobbe et al., 2021), 241 supported questions from the MBPP sanitized test split (Austin et al., 2021), and 256 sampled from the ARC-Challenge validation split (Clark et al., 2018). Question selection uses deterministic hash ordering of the eligible pools, after exact-identity and normalized-prompt exclusions against training data. ARC measures science knowledge reasoning,

not instruction following. IDs and normalized prompts are excluded against training pools. Code is a structurally supported subset: 16 unsupported items are excluded, and public test assertions remain in its input prompts. The same subset and sandbox scorer are used for every model. We use the same fixed benchmark subsets for all students, teachers, and methods; the reported scores refer to these subsets rather than the complete benchmarks.

The primary metric is the number of answers that are both correct and naturally terminated, without reaching the generation cap. We separately track answer correctness, strict Math formatting, cap hits, and repetition. The macro average is

$$A_{\text{macro}} = \frac{1}{3}\left(\frac{c_M}{256} + \frac{c_C}{241} + \frac{c_A}{256}\right). \tag{7}$$

Each domain therefore contributes one third of the reported score, despite its different question count. Math scoring checks the numeric final answer separately from compliance with the requested answer format; the literal prompt specification is given in Appendix B. Reported accuracy is answer-based, not strict-format accuracy. Earlier diagnostic subsets and different-backend evaluations are not mixed into the reported tables.

Comparisons and selection. The main table compares the complete recipe to natural-data Uniform and the two alternative strength-aware weighting rules. Appendix A retains the matched gradient-engine Uniform controls for attribution within the weighting stage. All use the same 128-update horizon. The N/B$\times$U/I/D series uses a static reference engine. Its positive initial calibration signal s_d is the domain mean of sampled student–teacher log-probability ratios on initial-student responses; static I/D use $v_d^{\text{static}} = s_d^{\alpha}$, followed by the same log-ratio cap and planned-share normalization in Eq. 6. Unlike D_d, this is a sampled signed signal, admitted by an initial positivity gate, and then frozen.

Table 6 separates the comparisons. Crossing from the static to the gradient engine assesses complete recipes, not a single changed factor. Within the gradient series, Uniform retains the same gradient computation and combination path. The study reports fixed final checkpoints; different-horizon follow-ups are not mixed into these tables. Teachers used for headroom recovery are evaluated on exactly the same diagnostic questions and inference protocol. The diagnostic panel was inspected during method development, so reported rankings describe this evaluation setting rather than an untouched confirmatory test.

Comparison hierarchy. Natural-data Uniform tests the complete recipe from the original allocation policy. Balanced-data Uniform tests the input schedule within the static engine. Strength-only and strength+Direct test the gap factor within the balanced gradient engine. Thus the gain over strength-only evaluates inverse weighting conditional on the measured gradient signal, whereas the gain over natural-data Uniform evaluates the entire two-stage procedure. For each comparison, the checkpoint horizon and evaluation identities remain fixed. The domain counts expose which capabilities contribute to the aggregate change, rather than allowing the macro average to substitute for per-domain assessment.

5 Results

5.1 Overall performance

Table 2 separates data allocation from loss weighting and reports domain counts alongside the macro average.

The complete B-MOPD recipe exceeds natural-data Uniform by 4.05 macro percentage points for Qwen2.5-0.5B-Instruct and 1.20 points for Qwen2.5-1.5B-Instruct. The corresponding Math/Code/ARC count changes are $+7, +2, +22$ and $+1, +4, +4$: all three domains improve relative to the end-to-end baseline. These recipe comparisons include the data schedule, gradient computation, and weighting rule.

Domain-level interpretation. The largest domain-accuracy improvement over natural-data Uniform occurs in ARC for the 0.5B student. Its Code score also increases over the distilled baseline,

Table 2: Final 128-update results on 753 diagnostic questions: naturally terminated correct counts and equal-domain macro accuracy. **Bold** marks the best score among the displayed MOPD conditions per student, excluding the initial student. Natural-data Uniform is the end-to-end baseline; matched-engine controls appear in Appendix A.

Data allocation	Loss weighting	Math/256	Code/241	ARC/256	Macro (%)
Student: Qwen2.5-0.5B-Instruct; teachers: 1.5B					
No MOPD	Initial student	76	114	69	34.65
Natural	Uniform (baseline)	115	106	101	42.79
Input-token balanced	Strength only	120	**108**	108	44.63
Input-token balanced	Strength + Direct	110	**108**	109	43.45
Input-token balanced	B-MOPD (ours)	**122**	**108**	**123**	**46.84**
Student: Qwen2.5-1.5B-Instruct; teachers: 7B					
No MOPD	Initial student	90	139	154	51.00
Natural	Uniform (baseline)	194	146	200	71.50
Input-token balanced	Strength only	193	139	**205**	71.05
Input-token balanced	Strength + Direct	194	144	200	71.22
Input-token balanced	B-MOPD (ours)	**195**	**150**	204	**72.70**

but remains below the initial student's 114 correct answers. For the 1.5B student, all three domains exceed both the initial student and natural-data Uniform. A decline in Code therefore depends on the student and reference point. Model capacity and competing optimization signals are possible explanations, but these experiments do not isolate their causes.

The gap direction matters after strength correction. Adding the inverse-gap factor to strength-only raises macro accuracy by 2.21 points for the 0.5B student and 1.65 points for the 1.5B student. Reversing the exponent to Direct yields 43.45% and 71.22%. These comparisons use the same balanced schedule and numerical engine. They support inverse rather than direct gap weighting when combined with the measured gradient strength in these runs. The separate matched-engine Uniform comparison is retained in Appendix A; static Inverse is examined below as a reference with a different estimator and update policy.

5.2 Teacher-relative recovery

We additionally evaluate each teacher on its own domain using exactly the student's diagnostic questions, rendered prompts, response budgets, greedy decoding, and scoring contract. These are the original teacher models, not separately distilled students. Their domain-routed macro score is denoted A_T. Following the teacher-relative headroom formulation of Gao et al. (2026), with initial student score A_0, define

$$R = \frac{A_{\text{method}} - A_0}{A_T - A_0}. \tag{8}$$

The denominator is positive in both configurations. We compute domain macro accuracies first and then apply Eq. 8; this is not the mean of per-domain recovery ratios, nor raw student accuracy divided by teacher accuracy.

Macro accuracy describes the deployed student's capability under the equal-domain objective. Recovery instead expresses that improvement relative to the headroom offered by the routed teachers. For a fixed initialization and teacher set, its denominator is constant and positive, so recovery preserves the accuracy ranking. Across configurations, both the initial student and the teacher reference

change; the recovery values must therefore be interpreted relative to each configuration's own teachers. Listing the teacher identities and their domain counts makes this normalization explicit.

Table 3: Teacher-relative headroom recovery on the same diagnostic panel. Teacher counts use the teacher specialized for each domain; this routed reference is not one deployable model. All scores use natural termination plus answer correctness.

Domain	Teacher model	Correct / total
Teachers for Qwen2.5-0.5B-Instruct		
Math	Qwen2.5-Math-1.5B-Instruct	201 / 256
Code	Qwen2.5-Coder-1.5B-Instruct	159 / 241
ARC	Qwen2.5-1.5B-Instruct	154 / 256
Teachers for Qwen2.5-1.5B-Instruct		
Math	Qwen2.5-Math-7B-Instruct	247 / 256
Code	Qwen2.5-Coder-7B-Instruct	208 / 241
ARC	Qwen2.5-7B-Instruct	232 / 256

	Teacher	Headroom recovered (%)	
Student model	macro (%)	Natural Uniform	B-MOPD
Qwen2.5-0.5B-Instruct	68.22	24.24	36.32
Qwen2.5-1.5B-Instruct	91.14	51.07	54.07

B-MOPD recovers 36.32% and 54.07% of the measured teacher-relative headroom, compared with 24.24% and 51.07% for natural-data Uniform (Table 3). Recovery normalizes improvement by the available accuracy gap; it is not a measurement of all knowledge retained from a teacher. It also differs from an integration-gap reference based on separately trained single-teacher students. The underlying domain scores remain the primary observations.

5.3 Component comparisons: data balance and static weights

Table 4: Data-allocation and static-weight references at 128 updates on the same 753-question panel. The two Uniform rows within a configuration isolate the data schedule. Static Inverse uses an initial frozen estimator and is not a pure removal of the gradient factor from B-MOPD.

Data allocation	Weighting	Math/256	Code/241	ARC/256	Macro (%)
Qwen2.5-0.5B-Instruct					
Natural	Uniform	115	106	101	42.79
Input-token balanced	Uniform	112	107	114	44.23
Input-token balanced	Static Inverse	123	101	109	44.18
Qwen2.5-1.5B-Instruct					
Natural	Uniform	194	146	200	71.50
Input-token balanced	Uniform	196	146	198	71.50
Input-token balanced	Static Inverse	190	147	209	72.29

The natural/balanced Uniform contrast changes only the data policy within the static family. Token balance improves the 0.5B student's macro accuracy by 1.44 points, primarily through ARC. For the 1.5B student, it preserves the macro average while trading two ARC answers for two Math answers. Input balance therefore defines a common exposure budget for signal allocation, rather than guaranteeing a gain on its own. Equal prompt-count balancing is a distinct schedule and is not evaluated in this comparison.

The realized schedule. The eligible pools contain 7,409 Math, 310 Code, and 1,054 ARC prompts. Natural allocation therefore favors Math by construction. Under the balanced schedule, the 3,072

occurrences instead comprise 1,186 Math, 688 Code, and 1,198 ARC prompts. Their cumulative raw-token totals are 97,462, 97,436, and 97,391—within 71 tokens across domains. Code requires fewer occurrences to reach comparable input exposure. Some queues wrap, but each occurrence receives a fresh response from the current student. These counts explain why both the data schedule and the planned prompt share π_d enter the method; uniform prompt thirds would describe a different experiment.

What the data-only comparison establishes. For the 0.5B student, balancing the inputs changes the Math/Code/ARC counts by $-3, +1, +13$ relative to natural-data Uniform. The gain is therefore a redistribution of learning toward the initially underrepresented domains, not a uniform increase in every score. For the 1.5B student, the unchanged macro score makes a complementary point: balancing exposure need not improve the aggregate by itself to provide a controlled starting point for signal weighting. We use the same balanced schedule for strength-only, strength+Direct, and strength+Inverse so their differences cannot be explained by different prompt allocations. The evidence supports this two-stage experimental decomposition; it does not establish input-token balance as universally superior to all other schedules.

Static Inverse as a reference. B-MOPD exceeds balanced-data static Inverse by 2.66 and 0.41 macro points. The comparison asks whether the complete gradient–gap recipe improves on a frozen inverse allocation. It changes gradient normalization, gap estimation, and temporal adaptation together, so it does not isolate a dynamic-versus-static effect. The main-table result identifies a consistent leading choice within the strength-aware family, not dominance over every possible data/weight recipe.

5.4 Bounded allocation in practice

The controller estimates weights online but does not require them to change at every window. In the 1.5B run, inverse weights reach the ratio cap after the first admitted window and remain fixed; the 0.5B run continues to change its allocation. Both behaviors are permitted by the same bounded rule. We interpret the result as a benefit of the resulting allocation, not isolated evidence that continuous weight changes are necessary. Applied trajectories and the corresponding gradient/KL estimates are provided in Appendix E.

Reading the rule as a bounded resource allocation. The three factors in Eq. 5 have different roles. Planned prompt share compensates for how often a route is present; the gradient norm compensates for the magnitude of its unweighted signal; inverse KL expresses a preference for a nearer teacher distribution. Their product is then normalized so that increasing one route's relative emphasis does not arbitrarily increase the planned mean loss coefficient. The ratio cap prevents any route from receiving an unbounded advantage when an estimated norm or gap is small. A stable coefficient vector is consequently a possible outcome of the controller, rather than a requirement to change weights simply because another window has elapsed.

6 Future Directions

The present study covers two student sizes and three domains. Extending the recipe to larger students and more diverse teacher sets is a natural next step. The weight-ratio cap and estimation window are fixed empirical choices; adapting them to estimator reliability, or incorporating optimizer-aware strength measurements, offers further directions for improving allocation.

7 Conclusion

B-MOPD organizes multi-teacher on-policy distillation into input-token balance and gradient–gap loss allocation. Relative to natural-data Uniform, it improves macro accuracy while preserving or improving each domain score in both evaluated student configurations. Component comparisons favor inverse gap after strength correction. The resulting framework makes exposure, signal strength, and teacher proximity explicit parts of MOPD design, using an online-estimated allocation that can change over time or settle within its prescribed bounds.

REPRODUCIBILITY STATEMENT

Equations 1–6, Figure 1, and Appendix B specify the algorithm and implementation choices. Frozen manifests, generation contracts, outputs, scores, and checkpoint receipts are retained; completed artifacts are checked for identity, integrity, termination, and arithmetic. Primary gradient runs exclude interrupted weight-only continuations. We claim neither verified exact resumption nor independent reproducibility through a currently public code/model release.

AI USAGE STATEMENT

AI tools were used to aid and polish writing, including grammar and spelling checks; retrieve and discover related work; draft sections of the manuscript; and assist research ideation and execution. The human authors reviewed the manuscript and references and are responsible for the accuracy of the reported work.

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 *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/5be69a584901a26c521c2b51e40a4c20-Abstract-Conference.html.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. GradNorm: Gradient normalization for adaptive loss balancing in deep multitask networks. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 794–803, 2018. URL https://proceedings.mlr.press/v80/chen18a.html.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018. URL https://arxiv.org/abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

Huan-ang Gao, Haohan Chi, Yong Yan, Shiyuan Feng, Hanlin Wu, Zheng Jiang, Bingxiang He, Wei-Ying Ma, Ya-Qin Zhang, and Hao Zhou. Open-MOPD: Diagnosing and fixing capability imbalance in multi-teacher on-policy distillation. *arXiv preprint arXiv:2608.19098*, 2026. URL https://arxiv.org/abs/2608.19098.

Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. MiniLLM: Knowledge distillation of large language models. In *International Conference on Learning Representations*, 2024. URL https://proceedings.iclr.cc/paper_files/paper/2024/hash/8ac015d409635f196f9e3e9dcfb9a94e-Abstract-Conference.html.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, Yunlong Feng, Xingzhang Ren, Xuancheng Ren, Jingren Zhou, and Junyang Lin. Qwen2.5-Coder technical report. *arXiv preprint arXiv:2409.12186*, 2024. URL https://arxiv.org/abs/2409.12186.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024. URL https://arxiv.org/abs/2412.15115.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. Qwen2.5-Math technical report: Toward mathematical expert model via self-improvement. *arXiv preprint arXiv:2409.12122*, 2024. URL https://arxiv.org/abs/2409.12122.

Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. Gradient surgery for multi-task learning. In *Advances in Neural Information Processing Systems*, volume 33, 2020. URL https://proceedings.neurips.cc/paper/2020/hash/3fe78a8acf5fda99de95303940a2420c-Abstract.html.

A Matched Gradient-Engine Controls

Natural-data Uniform is the baseline for the complete two-stage method. A different comparison holds the balanced data schedule and gradient accumulation engine fixed, setting all coefficients to one. These controls separate end-to-end recipe gains from the narrower effect of weighting within each numerical implementation. The gradient–inverse controller improves over this matched control

Table 5: Matched-engine control and gradient–inverse controller. Both use the same input-balanced schedule within a model configuration. Counts are Math/Code/ARC naturally terminated correct answers.

Student / panel	Weighting	Math	Code	ARC	Macro(%)
0.5B / 753	Uniform	116	105	113	44.34
	Strength + Inverse	122	108	123	46.84
1.5B / 753	Uniform	193	154	199	72.34
	Strength + Inverse	195	150	204	72.70

by 2.50 and 0.36 macro points. The 0.5B student improves in all three domains; the 1.5B student trades four Code answers for gains in Math and ARC. The control is matched within a configuration: the two student sizes have different gradient accumulation implementations, detailed below. A pure removal of gradient strength would additionally require the same current-KL estimator and temporal controller; static Inverse is not that isolated ablation.

B Implementation and Audit Details

Table 6: Allocation conditions. B fixes the input-balanced schedule; s_d is the frozen initial sampled signal, while $\bar{G}_d, \bar{D}_d$ use past training windows. All nonuniform coefficients are clipped and normalized. Cross-engine comparisons are references, not isolated controller ablations.

Engine / condition	Data	Unnormalized coefficient	Statistics
Static N–U / B–U	N / B	1	Fixed
Static B–I	B	s_d^{-1}	Initial, frozen
Gradient Uniform	B	1	Fixed
Gradient strength-only	B	$(\pi_d \bar{G}_d)^{-1}$	Past windows
Gradient strength+Direct	B	$\bar{D}_d/(\pi_d \bar{G}_d)$	Past windows
Gradient strength+Inverse	B	$(\pi_d \bar{G}_d \bar{D}_d)^{-1}$	Past windows

Eligible training pools. After the frozen calibration/development identity and normalized-prompt exclusions, the pools contain 7,409 Math, 310 Code, and 1,054 ARC prompts. N uses proportions given by these counts. The Code pool combines 56 retained sanitized training prompts with 254 additional full-MBPP training prompts at the same dataset revision; task IDs from test, validation, and prompt splits are excluded across both configurations. This increases training coverage but does not make full and sanitized records equivalent in quality. Within each domain, a deterministic hash order fixes the queue. The manifest and rendered-prompt records are frozen before allocation; no development score is used to reorder them.

Realized input schedule. The frozen balanced schedule is identical across the four weighting conditions within each configuration, and has the same counts in Small and Medium (Table 7). Counts denote prompt occurrences, not distinct training examples. Each repeated occurrence receives a fresh on-policy rollout. The planned prompt shares π_d in Eq. 5 are these occurrence counts divided by 3,072, not uniform thirds. Generated-response lengths remain policy-dependent and are not equalized.

Table 7: Frozen input exposure for each balanced-data gradient run. Raw token counts exclude the system message, chat template, and generated response.

Domain	Prompt occurrences	Distinct prompts	Raw input tokens
Math	1,186	1,186	97,462
Code	688	310	97,436
ARC	1,198	1,054	97,391

Optimizer and response aggregation. The gradient series uses AdamW with learning rate 5×10^{-6}, eight warmup updates, a cosine schedule with floor 5×10^{-7}, betas $(0.9, 0.95)$, epsilon 10^{-8}, zero weight decay, and global gradient clipping at 1. Losses are response-token means followed by sequence means, not one concatenated token mean. Teacher coefficients are normalized using frozen planned prompt shares. The gradient series is an HF/PyTorch reference engine, not a claim of equivalence to a distributed production training framework.

Precision controls. Both configurations maintain FP32 master parameters and optimizer state while computing model forward/backward operations in BF16. In Small, each example's backward gradient is transferred immediately into FP32 domain accumulation. In Medium, domain accumulation follows the earlier BF16 route before FP32 combination. The Small implementation passes an independently constructed per-example FP32 combination oracle, but does not satisfy the earlier legacy-engine equivalence tolerance; an unsuccessful zero-update gate was retained, not silently relaxed. The Medium uniform route comparison is within a 0.02 relative tolerance and is not exact equality. These checks motivate the own-engine Uniform rows. They do not demonstrate full-FP32 model gradients or identical training trajectories across engines.

Controller coverage. For the gradient controller, a completed eight-update window needs at least four gradient measurements and eight prompt occurrences in every domain. It holds if any mean gradient norm is at most 10^{-8} or any KL mean is at most 10^{-6}. Estimated statistics, admission decisions, emitted coefficients, next-update use, and cap saturation are recorded separately. The gap-only signed-signal controller in Appendix E is a distinct experiment, not the full-KL gradient controller.

Training budget and checkpointing. Each primary condition has 128 updates and 24 prompt occurrences per update. Checkpoints include weights and training state at designated updates; evaluations in the main table use the fixed final update, not the most favorable checkpoint. FP32/Adam/RNG/controller state is saved, but a bitwise or trajectory-level exact-resume claim has not been established.

Prompt and stopping contract. Prompts use the student tokenizer's chat template with a system message and a user message, followed by an assistant-generation prefix. The shared system text is: "Follow the user instructions, including the requested final-answer format. For code return only Python code without Markdown fences." Math requests reasoning followed by `#### <answer>`; Code supplies a task and public assertions and requests raw Python; ARC supplies the question and labeled choices and requests `Final answer: <label>`. The teacher scores the same rendered student trajectory. Evaluation is greedy, with repetition penalty one and the chat end token as the stop token. Per-row Math budgets reserve one context token; Code and ARC use the response caps stated in the main text.

Completed-run inclusion. The primary gradient rows use complete runs, not weight-only continuations with rebuilt optimizer state. Closed model/evaluation artifacts are checked independently of scheduler exit status. Interrupted continuations, canceled redundant runs, and failed preflight tests are retained in the execution record but do not count as independent seed replications or replace the primary comparisons.

C Realized Generation and Termination

Table 8 reports all generated response tokens seen by optimization, aggregated from completed-update records. The corresponding response positions receive teacher probabilities; these counts exclude prompt-prefill work and do not equal teacher-forward FLOPs. Policy-dependent lengths differ despite the shared input schedule, so the results are update/prompt-occurrence matched, not compute matched. No speedup or sample-efficiency claim is made.

Table 9 separates correctness from termination and formatting on the same frozen panel. Repetition flags use the fraction of repeated token 20-grams exceeding one half. Strict Math formatting is distinct from answer correctness and does not enter the macro score. These diagnostics retain all eight primary gradient endpoints without selecting a favorable condition.

Table 8: Realized training response-token counts for the primary gradient runs. Each row has 128 updates and 3,072 prompt occurrences. Cap hits count training responses reaching their domain budget. These totals are not a FLOP or wall-clock estimate.

Config.	Method	Math	Code	ARC	Total	Cap hits
Small	Uniform	409,492	60,799	254,640	724,931	1
Small	Strength	404,756	64,778	256,964	726,498	0
Small	Strength+D	411,428	61,939	260,340	733,707	1
Small	Strength+I	411,008	64,597	257,555	733,160	1
Medium	Uniform	355,818	65,121	86,861	507,800	0
Medium	Strength	369,323	67,003	90,044	526,370	3
Medium	Strength+D	363,994	78,610	89,390	531,994	1
Medium	Strength+I	368,539	74,701	93,155	536,395	2

Table 9: Diagnostic-panel behavior, by domain. Correct ignores termination; End counts natural terminations regardless of correctness; Cap and Repeat count cap hits and flagged repetition. Strict reports correct strict Math formatting only (a dash means not applicable to that column). Mean length counts generated tokens. Math/Code/ARC denominators are 256/241/256. These are diagnostics of the same frozen outputs, not new evaluations.

Config.	Method	Domain	Correct	End	Cap	Repeat	Strict	Mean length
Small	Uniform	Math	116	247	9	6	0	478.2
		Code	105	235	6	3	–	465.3
		ARC	113	256	0	0	–	210.0
Small	Strength	Math	120	250	6	6	0	428.4
		Code	108	236	5	3	–	402.9
		ARC	108	256	0	0	–	203.5
Small	Strength+D	Math	110	252	4	4	0	398.4
		Code	108	239	2	0	–	193.7
		ARC	109	255	1	1	–	211.0
Small	Strength+I	Math	122	254	2	1	0	382.9
		Code	108	237	4	1	–	332.7
		ARC	123	256	0	0	–	209.9
Medium	Uniform	Math	193	255	1	1	0	341.1
		Code	154	240	1	0	–	144.4
		ARC	199	256	0	0	–	51.1
Medium	Strength	Math	193	255	1	1	0	328.9
		Code	139	240	1	1	–	151.9
		ARC	205	256	0	0	–	49.7
Medium	Strength+D	Math	194	254	2	1	0	344.5
		Code	144	241	0	0	–	84.6
		ARC	200	256	0	0	–	52.7
Medium	Strength+I	Math	195	254	2	2	0	341.1
		Code	150	241	0	0	–	83.6
		ARC	204	256	0	0	–	53.4

D Evaluation Integrity and Statistical Scope

The diagnostic panel is selected by a deterministic hash ordering from frozen eligible public-source pools, after exact identity and normalized-prompt exclusions against training pools. Semantic equivalence and pretraining contamination are not established by these exclusions. In Code, the scorer supports a restricted assertion structure and executes model programs in a sandbox with positive/negative fixtures. A code-fence extraction flag is not a general measure of compliance with raw-code instructions.

Macro accuracy assigns equal weight to domains with unequal sample sizes. The reported ordering is descriptive, not a test of statistical significance. Method selection and diagnostic inspection are distinguished from an untouched confirmatory evaluation; final checkpoints and all reported domain scores are retained consistently.

Teacher and student in Table 3 share question identities and inference/scoring contracts. The 1.5B general teacher is identical to the larger initial student; overlapping ARC answers are reused only after exact model-file, prompt, budget, decoding, and score-receipt checks. All six own-domain teacher outputs pass independent closed-artifact SHA, identity, token, termination, and arithmetic checks. All reported student and teacher results use the same fixed 753-question panel. Its source datasets, split identities, deterministic selection, and supported Code subset are specified in the experimental protocol; it is not described as a full-benchmark evaluation.

E Coverage Gates and Realized Adaptation

Figure 2 plots the applied coefficients and the estimates available before each update. In the 1.5B run, all 16 statistic windows are admitted. Strength-only, Direct, and Inverse change their applied vectors 8, 15, and 1 times, respectively. Inverse settles at update 9: Math and Code receive coefficients of approximately 1.4134 and ARC receives 0.3533. Their ratio is four and the planned-share-weighted mean is one. The 0.5B run changes allocation over training. Thus statistical refresh, window admission, and an actual change in coefficients are distinct events.

A separate gap-only controller experiment illustrates why admission and action must be distinguished. Under the natural schedule, every eight-update window contains only 6–7 Code prompts, below that controller's eight-prompt threshold. All windows retain the preceding coefficients, despite positive signed-signal estimates. Under the balanced schedule, the analogous controllers change their coefficients 15 times. This diagnostic uses a signed sampled signal rather than the full conditional KL of the main gradient controller and is not an additional main-table ablation. Together with the capped Medium G–Inv trajectory, it shows why counting statistical refreshes alone can overstate adaptation.

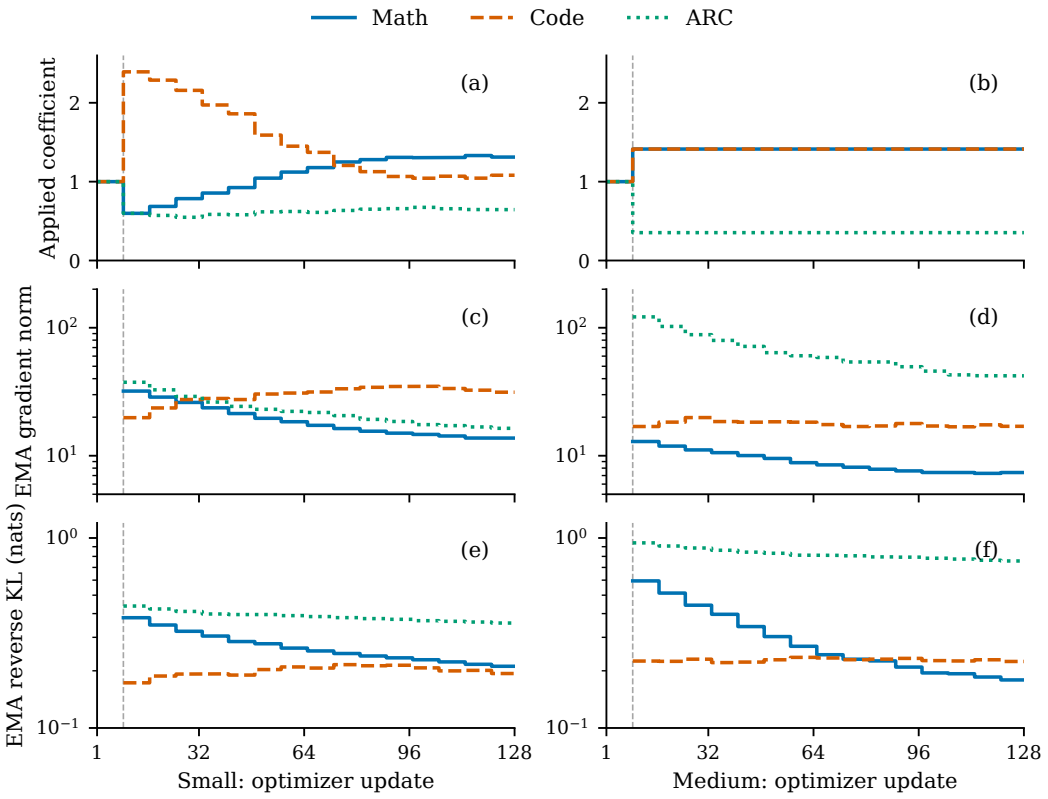

Figure 2: G–INV controller traces. Left: Qwen2.5-0.5B-Instruct; right: Qwen2.5-1.5B-Instruct. Panels show applied coefficients (a,b), all-parameter Euclidean gradient-norm estimates (c,d), and KL estimates (e,f), available before each update. The vertical guide marks update 9. The 1.5B student's Math/Code coefficients overlap at the ratio-capped plateau. Numerical engines differ across columns, so absolute gradient scales are not a pure model-size comparison.