SPECTRUM: PROXIMAL SPECTRAL MODULATION FOR LOOPED SELF-DISTILLATION

Anonymous authors
Paper under double-blind review

ABSTRACT

A model that learns from its own outputs inherits more than their correctness: it inherits which solutions it produces. We formulate Looped Self-Distillation, a self-evolution framework in which a model repeatedly generates and learns from its own raw outputs, under a fixed information budget, without ongoing external assessment or test-based selection of the generated samples. We identify a consequential separation: correctness can improve while the breadth of correct implementations contracts. We introduce SPECTRUM, which re-estimates loss-sensitive key/value geometry from a fixed reference anchor at each round and converts it into full-rank proximal spectral modulation. All generated completions train a single student, whose subsequent inference requires no intervention. After five rounds of experiments on MBPP, SPECTRUM retains 89.9% of the initial model's 64-sample correct AST richness, compared with 66.4% for Vanilla self-distillation and 65.5% for a subspace-projection control. The advantage persists at matched correct-sample counts. Without further training or recalibration, the resulting student also achieves higher matched-correct richness than Vanilla SD on HumanEval+ and APPS Intro, demonstrating transfer of the diversity benefit. These findings establish correct-solution retention as a complementary objective of recursive self-improvement and show that generation-time intervention can improve the solution repertoire retained by subsequent students.

1 INTRODUCTION

Iterative self-distillation has improved rationales and code-data synthesizers (Rao et al., 2023; Zhang et al., 2025). Existing approaches use correctness selection, reward-conditioned revision, evolving teachers, or decoding-based distribution shaping (Zelikman et al., 2022; Singh et al., 2024; He et al., 2026; Sang et al., 2026; Zhang et al., 2026). We ask: *what does a model retain when it repeatedly learns from its own outputs without new external assessments of those outputs?*

Each student supplies its next training distribution. Higher correctness need not make it a better source of diverse correct solutions (Figure 1). Recursive self-improvement therefore calls for measuring whether alternative successful behaviors survive and transfer beyond the training distribution.

We formulate *Looped Self-Distillation under a fixed information budget*. The evolving model generates a corpus, learns from every raw completion, and becomes the next generator. No new annotations, verifier rewards, judge scores, or quality-based sample selection enter training. An optional reference anchor supplies repeated calibration but is never expanded with newly judged samples.

Alternative proofs, plans, and programs make solution breadth relevant across reasoning tasks. We study code because executable tests and program structure make it measurable (Chen et al., 2021; Liu et al., 2023), and multiple implementations support program search (Li et al., 2022). Even the entire pass@k curve cannot identify the distribution within correct outputs: redistributing their probability while holding total correctness fixed leaves pass@k unchanged (Proposition 1).

Our experiments expose a separation between correctness and breadth. Across five rounds, Vanilla SD's pass@1 rises while correct implementation richness contracts (Figure 3; Appendix D). In the final 64-sample evaluation, pass@1 increases from 37.94% to 41.45%, while correct AST richness falls from 12.802 to 8.506 classes per task. This connects recursive synthetic-data degradation

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.

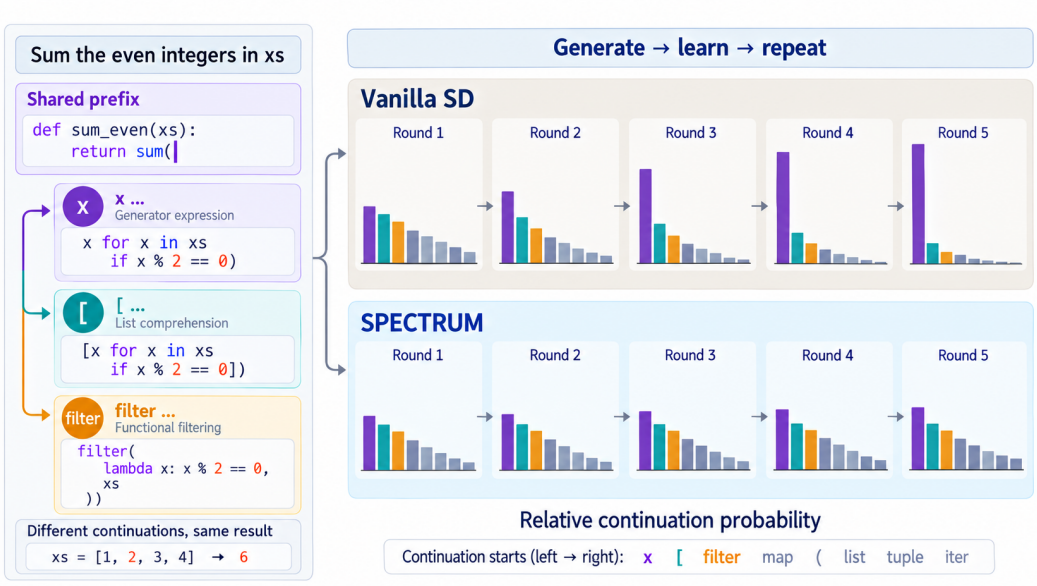


Figure 1: **Each student inherits a distribution of continuations.** A shared code prefix admits different continuation forms. The schematic histograms depict concentration under repeated Vanilla SD and broader retention under SPECTRUM. Heights are conceptual; the experiments measure complete correct programs, rather than the displayed continuation starts (Figure 3).

(Shumailov et al., 2024; Gerstgrasser et al., 2024) and self-distillation diversity loss (Nicolicioiu et al., 2026) to repeated learning from unselected outputs.

We introduce SPECTRUM to improve retention. Building on reference-gradient calibration (Hao et al., 2026), it re-estimates the student's loss-sensitive K/V geometry from the same anchor each round. A full-rank proximal gain attenuates less-sensitive directions while retaining a positive local response in every direction. The modulated model generates the corpus; native weights are restored before one LoRA learns from all completions (Hu et al., 2022). After five rounds, SPECTRUM retains 89.9% of initial 64-sample richness versus Vanilla SD's 66.4%, and improves pass@64 by 2.2 percentage points over Vanilla SD. Pass@1 is 2.39 points above initialization and 1.12 below Vanilla SD. Matched-correct richness gains transfer to HumanEval+ and APPS Intro. Our contributions are:

1. **A framework for Looped Self-Distillation.** Repeated learning from raw self-generated corpora under a fixed information budget, without ongoing external assessment or selection of generated samples.
2. **SPECTRUM: proximal spectral modulation.** Per-round loss-sensitive geometry and a full-rank generation operator transfer a temporary intervention into one native student.
3. **Solution retention across rounds and benchmarks.** Longitudinal, sampling-budget, generation-design, and transfer analyses establish improved retention within correct implementations.

2 PRELIMINARIES

2.1 SELF-DISTILLATION FROM MODEL-GENERATED OUTPUTS

SSD learns from self-generated code through cross-entropy training, using temperature and truncation to shape its synthetic distribution without correctness-based selection (Zhang et al., 2026). We study this sequence-level learning mechanism when the updated student repeatedly becomes its own data source.

2.2 LOOPED SELF-DISTILLATION

We formulate *Looped Self-Distillation* as repeated generation and learning under a fixed information budget. Let θ_t denote the current student, $\mathcal{Q}$ a fixed set of synthesis prompts, and $\mathcal{G}$ a generation constructor. One round consists of

$$\begin{aligned} \widetilde{\pi}_t &= \mathcal{G}(\theta_t; \mathcal{A}), \\ \mathcal{D}_t &= \big((x, Y_{x,s})\big)_{x\in\mathcal{Q},\ s=1,\dots,S}, \qquad Y_{x,s} \sim \widetilde{\pi}_t(\cdot \mid x), \\ \theta_{t+1} &= \mathcal{L}(\theta_t, \mathcal{D}_t), \end{aligned} \tag{1}$$

where $\mathcal{D}_t$ is an indexed corpus retaining duplicate records and $\mathcal{L}$ learns from every generated completion through cross-entropy training. The optional reference information $\mathcal{A}$ is fixed before the loop. Newly generated training samples receive no ongoing external evaluation or supervision: no verifier or judge determines their selection or training weights.

The loop closes when θ_{t+1} becomes the next generator, exposing how an update changes the distribution available for future learning. This specifies a learning mechanism within the broader self-improvement agenda. Our implementation uses one LoRA per round (Hu et al., 2022); the outer data–parameter loop differs from recurrent-depth transformers that repeat computation within inference (Dehghani et al., 2019).

2.3 CORRECTNESS AND CORRECT-SOLUTION BREADTH

For a task x, let $v_x(y) \in \{0, 1\}$ denote passing its recorded verifier, and let $\phi(y)$ identify a normalized AST class. Let p_θ denote raw model likelihood and π_θ the fixed decoding policy induced by the model. Define $a_\theta(x) = \Pr_{\pi_\theta}[v_x(Y) = 1]$ and, when $a_\theta(x) > 0$,

$$q_{\theta,j}(x) = \Pr_{\pi_\theta}[\phi(Y) = j \mid v_x(Y) = 1], \qquad \Pr[v_x(Y) = 1, \phi(Y) = j] = a_\theta(x) q_{\theta,j}(x). \tag{2}$$

The first factor measures success frequency; the second describes the distribution within successful outputs. Suppress x and assume independent draws from a fixed decoding policy below.

Proposition 1 (Correctness does not identify correct-solution breadth)**.** *For fixed $a > 0$, pass@$k = 1 - (1 - a)^k$ is independent of q. Expected correct-class counts under k total draws and b correct draws are*

$$C_k(a, q) = \sum_j [1 - (1 - a q_j)^k], \qquad D_b(q) = \sum_j [1 - (1 - q_j)^b]. \tag{3}$$

Identical pass@k at every k can coexist with different D_b for any $b \geq 2$.

We call C_k *correct AST richness@k* and D_b *correct-count-matched AST richness@b*. Finite-pool rarefaction estimators (Hurlbert, 1971) and the proof appear in Appendix A. Conditional comparisons use tasks eligible for both methods and report their number. Across rounds, relative richness retention is the ratio of task-mean C_k to its initial-model value. It measures expected class counts at a fixed budget, rather than identity overlap with the initial classes.

3 SPECTRUM: PROXIMAL SPECTRAL MODULATION

SPECTRUM temporarily modulates the current model's K/V representations to generate its next training corpus. Each round recalibrates the geometry from the fixed anchor, generates under a full-rank spectral operator, restores native weights, and trains one student on all raw completions (Figure 2).

3.1 LOSS-SENSITIVE KEY/VALUE GEOMETRY

Let $\mathcal{A} = \{(x_i, y_i^\star)\}$ be the reference anchor fixed before the loop. For the current student θ_t, let $\ell_i(\theta_t)$ denote the mean negative log likelihood of reference completion $y_i^\star$. Prompt and padding targets are masked. Following reference-loss-gradient calibration of K/V directions (Hao et al., 2026), collect the gradient vector $g_{iu}^{(\ell)} \in \mathbb{R}^{d_\ell}$ with respect to the native linear output of each selected K/V module

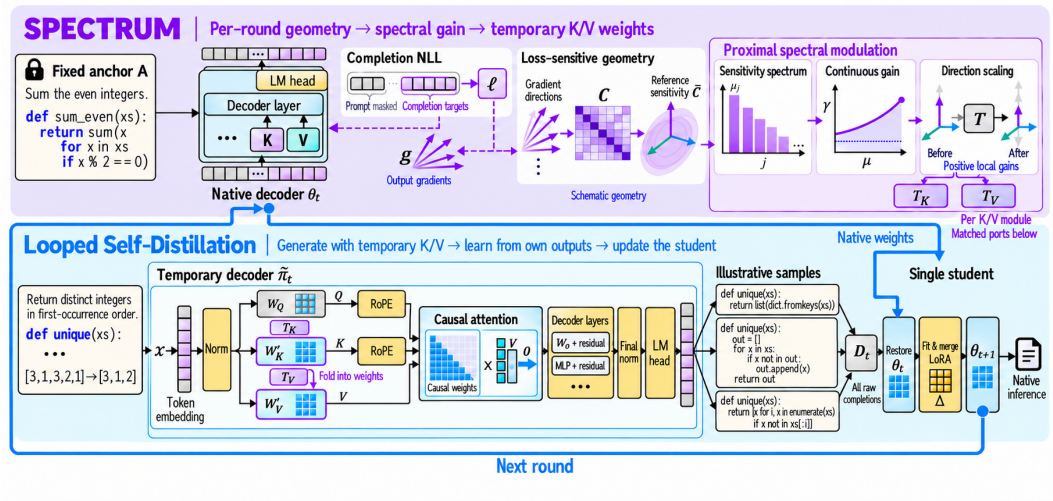


Figure 2: **SPECTRUM within Looped Self-Distillation.** Fixed-anchor completion gradients determine a per-round K/V operator, folded into temporary generation weights. All raw completions train one LoRA student after native weights are restored. The merged student starts the next round and supports native inference. Dashed arrows denote calibration gradients; the bottom loop carries updated parameters. Geometry and code examples are schematic.

ℓ at nonpadding position u. Each module is calibrated separately. Its uncentered gradient second moment and normalized geometry are

$$C_{t,\ell} = \frac{1}{M_\ell} \sum_{i,u,\text{nonpad}} g_{iu}^{(\ell)} g_{iu}^{(\ell)\top}, \qquad \overline{C}_{t,\ell} = \frac{C_{t,\ell}}{\lambda_{\max}(C_{t,\ell})}. \tag{4}$$

Here M_ℓ counts all included positions, including zero gradients. Nonzero calibration signal gives $0 \preceq \overline{C}_{t,\ell} \preceq \mathbf{I}$. Prompt positions are included because they influence completion loss despite their masked prediction targets.

For a unit direction v, $v^\top C_{t,\ell} v$ is the mean squared directional derivative $(g_{iu}^{(\ell)\top} v)^2$. A finite anchor observes only part of the model's behavior, so low measured sensitivity need not imply uselessness for other valid implementations. We use this statistic for continuous attenuation and re-estimate it under each updated student.

3.2 Full-rank proximal spectral modulation

Suppressing round and module indices, consider a row activation $h \in \mathbb{R}^{1\times d}$. We choose the quadratic penalty $z(\mathbf{I} - \overline{C})z^\top$ and define

$$\begin{aligned} z^\star &= \arg\min_{z\in\mathbb{R}^{1\times d}} \left\{ \frac{1}{2}\|z-h\|_2^2 + \frac{\tau}{2} z(\mathbf{I}-\overline{C})z^\top \right\} = hT_\tau, \\ T_\tau &= [\mathbf{I} + \tau(\mathbf{I}-\overline{C})]^{-1}, \qquad \tau \geq 0. \end{aligned} \tag{5}$$

This quadratic proximal map (Parikh & Boyd, 2014) limits departure from the native activation while penalizing less-sensitive directions more strongly at fixed norm.

Writing $\overline{C} = U\,\mathrm{diag}(\mu_j)U^\top$ gives the closed-form response

$$T_\tau = U\,\mathrm{diag}\left(\frac{1}{1+\tau(1-\mu_j)}\right)U^\top. \tag{6}$$

The highest-sensitivity directions retain unit gain; all others retain positive gains at finite τ. Strength varies continuously without a prescribed rank.

Proposition 2 (Local distinguishability and calibration stability). *Let* $0 \preceq \overline{C} \preceq \mathbf{I}$ *and* $0 \le \tau < \infty$. *Equation* (5) *has a unique minimizer, and the gains of* T_τ *lie in* $[1/(1+\tau), 1]$. *For any row-vector difference* δ,

$$\frac{\|\delta\|_2}{1+\tau} \le \|\delta T_\tau\|_2 \le \|\delta\|_2, \qquad \|T_\tau - \mathbf{I}\|_2 \le \frac{\tau}{1+\tau}. \tag{7}$$

For $0 \preceq A, B \preceq \mathbf{I}$, *the corresponding operators satisfy* $\|T_\tau(A) - T_\tau(B)\|_2 \le \tau \|A - B\|_2$.

The operator also increases or preserves relative alignment with the measured sensitivity geometry. Define $s_{\overline{C}}(h) = h\overline{C}h^\top / \|h\|_2^2$ for $h \neq 0$. Monotonicity of the gains in μ_j yields

$$s_{\overline{C}}(hT_\tau) \ge s_{\overline{C}}(h). \tag{8}$$

These local properties establish bounded, sensitivity-aligned modulation and calibration stability (Appendix B). Retention after attention, decoding, and student learning is measured experimentally.

3.3 Distilling the modulated generator

For a native linear module $h = xW^\top + b^\top$, the row operator can be folded into its parameters:

$$W' = T_\tau^\top W, \qquad b' = T_\tau^\top b, \qquad xW'^\top + b'^\top = hT_\tau. \tag{9}$$

Folding acts at native K/V outputs before rotary transformations and key normalization, requiring no generation-time activation hooks. Keys affect query–key compatibility; values affect the information aggregated by attention.

After generating $\mathcal{D}_t$, we restore the original weights θ_t and fit one LoRA adapter (Hu et al., 2022). Every raw record enters training, including unsuccessful, empty, and duplicate completions. For tokenized record $z = (x, y)$, let $\mathcal{T}(z)$ contain its nonpadding causal targets, and let Δ denote the effective LoRA update. The student objective is

$$\mathcal{L}_t(\Delta) = -\frac{1}{|\mathcal{D}_t|} \sum_{z \in \mathcal{D}_t} \frac{1}{|\mathcal{T}(z)|} \sum_{u \in \mathcal{T}(z)} \log p_{\theta_t + \Delta}(z_u \mid z_{<u}). \tag{10}$$

SFT includes prompt targets, whereas calibration uses completion targets only. Every record contributes an equally weighted token-mean loss; the operator changes sequence frequencies, not correctness-based training weights.

The merged adapter yields θ_{t+1}. Restoring native weights prevents direct composition of operators across rounds; their influence passes through the corpora. Final inference uses neither operator nor anchor.

Fixed reference information, no ongoing external feedback. The reference anchor is supplied before the loop and reused unchanged. Recalibration acquires no new annotations, external-model judgments, or verifier feedback. All generated records retain equal training weight; tests only evaluate the resulting students.

4 Experiments

4.1 Protocol and research questions

Learning and evaluation. We study five rounds with Qwen2.5-Coder-1.5B-Instruct (Hui et al., 2024) on MBPP (Austin et al., 2021). After removing three exact normalized train/test prompt overlaps, the disjoint splits contain 291 synthesis, 50 calibration, 30 validation, and 500 evaluation tasks. Each round generates 16 raw completions per synthesis task and fits one LoRA epoch: 4,656 records per round, 23,280 overall. SPECTRUM uses $\tau = 1$, middle and last decoder-layer K/V modules, and rank 8. The anchor and hyperparameters stay fixed.

Native evaluation uses temperature 0.8, top-p 0.95, and no top-k truncation. Every round has 16 samples per task; a separate final pool has 64 for the initial, Vanilla SD, projection, and SPECTRUM checkpoints. SSD is compared under the common 16-sample protocol. We report pass@k, C_k, and D_b, using common eligible tasks for paired conditional contrasts. The longitudinal study uses one training run; intervals use 2,000 task-bootstrap resamples. Appendix C gives complete settings, and Appendix E provides independent initial-model context across four models and five benchmarks.

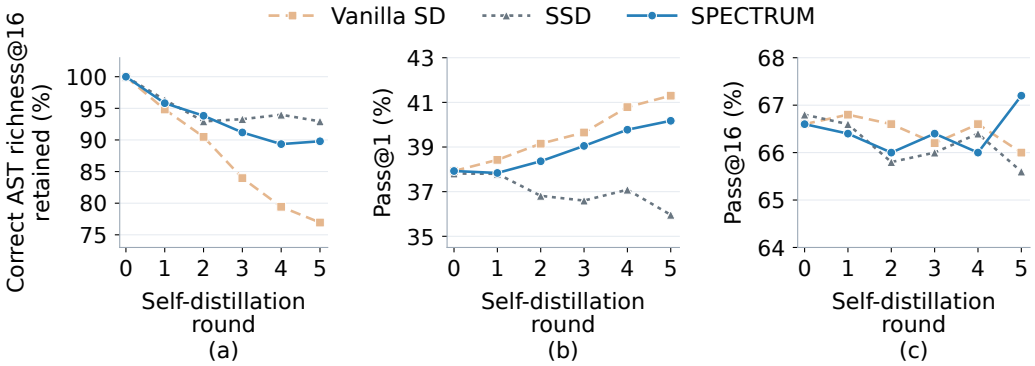


Figure 3: **Generation design separates correctness from solution retention.** Native students on 500 MBPP tasks, 16 samples per task per round. (a) C_{16} relative to each recorded initialization; (b) pass@1; (c) pass@16. All panels show point estimates from one training trajectory per design. SSD's initial model is sampled independently. Additional round-level results appear in Appendix D.

Table 1: **Final native students after five MBPP rounds.** Independent 64-sample evaluation on 500 tasks. Bold marks the better trained-student estimate. D_4 is a marginal mean over 322/321/323 eligible tasks, respectively; paired contrasts follow separately.

Metric ↑	Initial model	Vanilla SD	SPECTRUM
pass@1 (%)	37.94	**41.45**	40.33
pass@8 (%)	61.10	61.86	**61.89**
pass@16 (%)	65.69	65.64	**66.23**
pass@64 (%)	72.00	70.40	**72.60**
Correct AST richness@64 (C_{64})	12.802	8.506	**11.510**
Relative richness retention (%)	100.0	66.4	**89.9**
Correct-count-matched richness@4 (D_4)	3.358	2.856	**3.143**

Comparators and questions. Vanilla SD uses standard sampling; SSD shapes generation through temperature 1.5, top-p 0.8, and top-k 20 (Zhang et al., 2026). A top-half rank projector is a generation-design ablation adapted from Hao et al. (2026). Projection and SPECTRUM share the 50 references and calibration procedure; Vanilla SD and SSD use no anchor. Synthesis tasks, record counts, and optimization schedules are shared. We ask what survives successive updates (**RQ1**), whether gains persist within correct outputs (**RQ2**), how generation design changes the result (**RQ3**), and whether retention transfers (**RQ4**). A separate one-round pilot supplies complementary evidence under a different training budget.

4.2 RQ1: WHAT DO SUCCESSIVE STUDENTS RETAIN?

Vanilla SD's pass@1 rises from 37.92% to 41.30% while C_{16} falls from 4.016 to 3.090. SPECTRUM instead retains 89.8% of initial C_{16}, versus Vanilla SD's 76.9%, with greater richness at every learned round (Figure 3a). Its correctness also improves: pass@1 ends at 40.18%, and pass@16 at 67.2% versus Vanilla SD's 66.0%. The joint trajectory identifies a student that improves success while retaining more of its correct repertoire.

The separate 64-sample pool confirms the retention advantage (Table 1): 11.510 correct AST classes versus 8.506, or 35.3% more, with paired difference 3.004 $[2.568, 3.494]$. Retention reaches 89.9% versus 66.4%. SPECTRUM solves 363 tasks versus 352, a pass@64 gain of 2.2 percentage points $[0.4, 4.2]$. Pass@1 is 2.39 points above initialization and 1.12 below Vanilla SD $[-1.61, -0.63]$. Thus richer native students solve more tasks at high budget, with a smaller single-draw success rate than Vanilla SD.

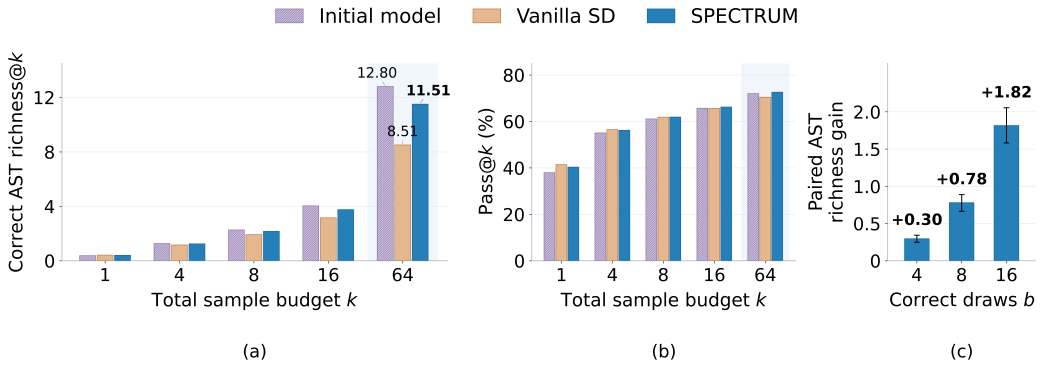


Figure 4: (a,b) Correct AST richness and pass@k from the same final 64-sample pools on 500 MBPP tasks. Budget groups are categorical. (c) Paired SPECTRUM-minus-Vanilla SD D_b gains; whiskers are 95% task-bootstrap intervals on 316, 296, and 254 common eligible tasks for $b = 4, 8, 16$.

Table 2: **Generation designs and realized training tokens.** Round-five native students, 500 MBPP tasks, 16 evaluation draws. Success and richness are higher-is-better; D_4 uses each method's eligible tasks (Appendix D). SFT tokens sum five rounds, in millions. All designs use 23,280 training records; these are not matched-compute measurements.

Design	pass@1 (%)	pass@16 (%)	C_8	C_{16}	D_4	SFT tokens (M)
Vanilla SD	41.30	66.00	1.902	3.090	2.747	6.373
SSD	35.96	65.60	2.091	3.710	3.268	7.326
Projection control	41.35	65.60	1.876	3.066	2.690	6.240
SPECTRUM	40.18	67.20	2.091	3.606	2.997	6.856

4.3 RQ2: DOES THE ADVANTAGE PERSIST WITHIN CORRECT OUTPUTS?

Matching successful draws isolates breadth within correct outputs. At four correct draws, SPECTRUM exceeds Vanilla SD by 0.295 expected AST classes $[0.249, 0.343]$ on 316 common eligible tasks. At eight and sixteen correct draws, the paired gains are 0.780 and 1.816 on 296 and 254 tasks (Figure 4c). The advantage therefore persists after equalizing correct-sample counts; each budget uses its own common eligible population.

The total-budget curves provide the complementary view. At $k = 1$, C_1 equals correctness and favors Vanilla SD. At every displayed larger budget, SPECTRUM produces more distinct correct structures, with its largest pass@k advantage at $k = 64$ (Figure 4a,b). These endpoints distinguish breadth bought by a generation budget from variety conditional on success.

4.4 RQ3: HOW DOES GENERATION DESIGN AFFECT RETENTION AND CORRECTNESS?

Improving correctness relative to decoding-based SSD. SPECTRUM achieves the highest pass@16 among the evaluated designs. Across the five-round trajectories, its pass@1 improves by 2.25 points from its recorded initialization, whereas SSD declines by 1.85 points (Figure 3b). At round five, SPECTRUM exceeds SSD by 4.21 points in pass@1 and 1.60 in pass@16. Their C_8 point estimates are nearly identical (2.091), while SSD has higher C_{16} and marginal D_4 (Table 2). The designs occupy distinct correctness–richness operating points.

Table 3: **Matched-correct richness transfers to both benchmarks.** SPECTRUM minus Vanilla SD after MBPP learning. D_4 uses common eligible tasks; accuracy differences use all tasks. Intervals are 95% paired task-bootstrap intervals. Full absolute results appear in Table 9.

Benchmark	$\Delta D_4 \uparrow$	95% interval	Common tasks	Δpass@1 (pp)	Δpass@16 (pp)
HumanEval+	$+0.080$	$[0.021, 0.142]$	111	-1.52	0.00
APPS Intro	$+0.104$	$[0.007, 0.205]$	42	-1.22	-1.50

Table 4: **Exploratory one-round comparison with SSD.** 64 MBPP tasks, 64 samples per task; a separate training protocol. Values are the archived rounded summaries. Bold marks the larger estimate. Full comparisons and reported intervals are in Appendix H.

Design	pass@1 (%)	pass@8 (%)	pass@32 (%)	pass@64 (%)	D_4	AST Simpson
SSD	**36.89**	**56.79**	61.66	62.50	3.391	0.863
SPECTRUM	36.65	56.18	**63.11**	**65.63**	**3.422**	**0.872**

The accuracy gain uses 8.1% fewer generated completion tokens and 6.4% fewer SFT tokens than SSD. Relative to Vanilla SD, generated tokens are 9.9% higher. These are realized text budgets, not wall-clock or total-compute comparisons: SPECTRUM also performs calibration (Appendix G).

Continuous modulation versus projection. With the same reference anchor, calibration, module locations, and learning schedule, continuous modulation retains 89.9% of initial C_{64} versus projection's 65.5%. It gains 3.122 classes and 2.4 pass@64 points, with pass@1 lower by 1.43 points. On 251 common eligible tasks in the 16-sample pool, paired D_4 gains are 0.312 $[0.253, 0.377]$ for exact programs and 0.293 $[0.233, 0.353]$ for AST classes; the coarser control-flow gain is 0.042 $[-0.011, 0.095]$. The evidence is strongest for implementation breadth. Attenuation strengths differ, so this contrast evaluates the complete operators.

4.5 RQ4: DOES RETAINED DIVERSITY TRANSFER ACROSS BENCHMARKS?

We freeze the MBPP-trained final students and evaluate 164 HumanEval+ tasks and a fixed 200-task APPS Intro subset with 16 samples per task, without further training or recalibration. HumanEval+ uses EvalPlus 0.3.1 (Liu et al., 2023); APPS uses the recorded Python function-interface adaptation (Hendrycks et al., 2021) (Appendix C).

The paired richness advantage is positive on both benchmarks (Table 3). On HumanEval+, SPECTRUM also matches Vanilla SD's pass@16 (86.59%) and raises the C_{16} point estimate from 5.848 to 6.000. APPS C_{16} falls from 1.930 to 1.845, alongside its lower accuracy. Thus the transferable benefit lies in matched-correct implementation breadth, with benchmark-dependent success rates. Appendix F shows recorded equal-correct-count examples.

4.6 COMPLEMENTARY EVIDENCE FROM A ONE-ROUND PILOT

An earlier exploratory pilot evaluates 64 MBPP tasks with 64 samples each. It uses one generate–learn cycle, 128 synthesis tasks with one completion each, and five SFT epochs, separately from the five-round protocol. SPECTRUM improves SSD's pass@32, pass@64, and matched-correct richness (Table 4); the recorded paired D_4 gain is 0.0312 $[0.0095, 0.0550]$, with pass@1 lower by 0.244 points. This supplies complementary evidence under another training budget. Appendix H preserves all four designs and the pilot's post-hoc first-fence extraction protocol.

5 RELATED WORK

Sequence-level self-distillation. SSD fits self-generated, unverified code after shaping its distribution through temperature and truncation (Zhang et al., 2026). This is our closest learning paradigm.

SPECTRUM instead shapes generation with fixed reference-loss geometry and measures what native students retain across complete generation–learning cycles. Vanilla SD and the adapted SSD recipe use no reference completions.

Iterative self-improvement and evolving teachers. STaR and ReST-EM learn from solutions selected through correctness feedback (Zelikman et al., 2022; Singh et al., 2024). Iterative self-distillation has explicit precedents: Rao et al. (2023) use critics and filtering for contexts and rationales, while SCoder improves a code-data synthesizer through checkpoint sampling, scoring, and selection (Zhang et al., 2025). SD-Zero distills reward-conditioned self-revision with teacher synchronization (He et al., 2026); CRISP refreshes an instruction-conditioned teacher for token-level KL training without correctness filtering (Sang et al., 2026). Our setting fixes synthesis prompts and optional reference information, retains every raw completion, and examines successive students as future data sources.

Diversity under recursive learning. Nicolicioiu et al. (2026) show functional and semantic diversity loss with sampled correct demonstrations supplied to a self-teacher. Our loop omits that demonstration-conditioned feedback. UA-RL rewards rare strategies (Hu et al., 2026); SPECTRUM modulates generation without scoring or labeling generated strategies. Recursive model-collapse studies examine synthetic-data degradation, and data accumulation can mitigate it (Shumailov et al., 2024; Gerstgrasser et al., 2024). We hold the historical-data policy fixed and jointly measure pass@k, C_k, and D_b to separate correctness from breadth within correct outputs.

6 LIMITATIONS

Our evaluation has three limitations. First, AST richness captures implementation structure rather than semantic algorithm identity. Structurally different programs may share an underlying algorithm, so semantic and behavioral comparisons would provide a more complete account of solution diversity. Second, finite tests and sample budgets can miss invalid behaviors and rare implementations, motivating broader behavioral checks and adaptive sampling. Third, diversity requires task-specific interpretation: Python AST classes do not directly correspond to distinct proofs, plans, or explanations across languages. Extending this framework requires representations that capture meaningful alternatives within each domain.

7 CONCLUSION AND FUTURE WORK

We introduce Looped Self-Distillation as a framework for repeated self-improvement under a fixed information budget, without ongoing external assessment of generated samples. Each student supplies its next training distribution, and correctness can improve while correct-solution breadth contracts. SPECTRUM addresses this retention problem through evolving K/V geometry and full-rank proximal modulation, transferring a temporary generation intervention into a native student through unfiltered data. After five MBPP rounds, it retains 89.9% of initial 64-sample richness versus Vanilla SD's 66.4%. Matched-correct richness gains also appear on HumanEval+ and APPS Intro.

Retention and transfer matter for long-horizon recursive self-improvement. Future work will investigate whether lifelong agents can consolidate experience into reusable concepts and strategies, then recombine them to solve unfamiliar problems without additional task-specific training.

REPRODUCIBILITY STATEMENT

The appendix specifies the metric estimators, proofs, information flow, losses, data splits, prompts, decoding, and resource accounting. Supplementary tables report complete round-level results, matched-correct comparisons, transfer effects, and resource use. Additional analyses use saved task-level results; training and evaluation protocols are documented separately from these analyses.

AI USE STATEMENT

AI assistants supported literature retrieval, experiment design and software, mathematical derivation and checking, result analysis, manuscript drafting, and figure preparation. The studied models generated the synthetic training corpora as specified in the protocol. Reported values come from recorded evaluations and identified analyses of saved outputs.

ETHICS STATEMENT

This study uses existing public coding benchmarks and involves no newly recruited human participants. Passing finite tests does not establish that generated code is safe or suitable for deployment; deployment requires independent validation.

REFERENCES

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, 2021. URL https://arxiv.org/abs/2108.07732. arXiv preprint.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374. arXiv preprint.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *International Conference on Learning Representations*, 2019. URL https://arxiv.org/abs/1807.03819.

Gemma Team, Aishwarya Kamath, Johan Ferret, Shreya Pathak, Nino Vieillard, Ramona Merhej, Sarah Perrin, Tatiana Matejovicova, et al. Gemma 3 technical report, 2025. URL https://arxiv.org/abs/2503.19786.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, Daniel A. Roberts, Diyi Yang, David L. Donoho, and Sanmi Koyejo. Is model collapse inevitable? breaking the curse of recursion by accumulating real and synthetic data, 2024. URL https://arxiv.org/abs/2404.01413. arXiv preprint.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Y. Wu, Y. K. Li, Fuli Luo, Yingfei Xiong, and Wenfeng Liang. DeepSeek-Coder: When the large language model meets programming—the rise of code intelligence, 2024. URL https://arxiv.org/abs/2401.14196.

Guangya Hao, Yitong Shang, Yunbo Long, Zhuokai Zhao, and Hanxue Liang. Self-policy distillation via capability-selective subspace projection, 2026. URL https://arxiv.org/abs/2605.22675. arXiv preprint.

Yinghui He, Simran Kaur, Adithya Bhaskar, Yongjin Yang, Jiarui Liu, Narutatsu Ri, Liam Fowl, Abhishek Panigrahi, Danqi Chen, and Sanjeev Arora. Self-distillation zero: Self-revision turns binary rewards into dense supervision, 2026. URL https://arxiv.org/abs/2604.12002.

Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, and Jacob Steinhardt. Measuring coding challenge competence with APPS, 2021. URL https://arxiv.org/abs/2105.09938.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=nZeVKeeFYf9.

Zhiyuan Hu, Yucheng Wang, Yufei He, Jiaying Wu, Yilun Zhao, See-Kiong Ng, Cynthia Breazeal, Anh Tuan Luu, Hae Won Park, and Bryan Hooi. Rewarding the rare: Uniqueness-aware RL for creative problem solving in LLMs. In *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 39765–39790, San Diego, California, United States, July 2026. Association for Computational Linguistics. doi: 10.18653/v1/2026.findings-acl.1982. URL https://aclanthology.org/2026.findings-acl.1982/.

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, 2024. URL https://arxiv.org/abs/2409.12186. arXiv preprint.

Stuart H. Hurlbert. The nonconcept of species diversity: A critique and alternative parameters. *Ecology*, 52(4):577–586, 1971. doi: 10.2307/1934145. URL https://esajournals.onlinelibrary.wiley.com/doi/10.2307/1934145.

Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code, 2024. URL https://arxiv.org/abs/2403.07974.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with AlphaCode, 2022. URL https://arxiv.org/abs/2203.07814.

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 *Advances in Neural Information Processing Systems*, volume 36, 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/43e9d647ccd3e4b7b5baab53f0368686-Abstract-Conference.html.

Andrei Liviu Nicolicioiu, Mohammad Pezeshki, and Aaron Courville. On-policy self-distillation with sampled demonstrations reduces output diversity, 2026. URL https://arxiv.org/abs/2606.26091v1.

Neal Parikh and Stephen Boyd. Proximal algorithms. *Foundations and Trends in Optimization*, 1(3), 2014. doi: 10.1561/2400000003. URL https://web.stanford.edu/~boyd/papers/prox_algs.html.

Kavel Rao, Liwei Jiang, Valentina Pyatkin, Yuling Gu, Niket Tandon, Nouha Dziri, Faeze Brahman, and Yejin Choi. What makes it ok to set a fire? iterative self-distillation of contexts and rationales

for disambiguating defeasible social and moral situations. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023. doi: 10.18653/v1/2023.findings-emnlp.812. URL https://aclanthology.org/2023.findings-emnlp.812/.

Hejian Sang, Yuanda Xu, Zhengze Zhou, Ran He, Zhipeng Wang, and Jiachen Sun. CRISP: Compressed reasoning via iterative self-policy distillation, 2026. URL https://arxiv.org/abs/2603.05433v7.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. AI models collapse when trained on recursively generated data. *Nature*, 631:755–759, 2024. doi: 10.1038/s41586-024-07566-y. URL https://www.nature.com/articles/s41586-024-07566-y.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2024. URL https://arxiv.org/abs/2312.06585.

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.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. STaR: Bootstrapping reasoning with reasoning, 2022. URL https://arxiv.org/abs/2203.14465.

Ruixiang Zhang, Richard He Bai, Huangjie Zheng, Navdeep Jaitly, Ronan Collobert, and Yizhe Zhang. Embarrassingly simple self-distillation improves code generation, 2026. URL https://arxiv.org/abs/2604.01193. arXiv preprint.

Xinyu Zhang, Changzhi Zhou, Linmei Hu, Luhao Zhang, Xiancai Chen, Haomin Fu, Yang Yang, and Mengdi Zhang. SCoder: Progressive self-distillation for bootstrapping small-scale data synthesizers to empower code LLMs. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 20825–20841, 2025. doi: 10.18653/v1/2025.findings-emnlp.1136. URL https://aclanthology.org/2025.findings-emnlp.1136/.

A METRICS, ESTIMATORS, AND PROBABILITY ANALYSIS

A.1 CORRECTNESS AND CONDITIONAL IMPLEMENTATION DISTRIBUTIONS

For a fixed task, let $V = v_x(Y)$ and $J = \phi(Y)$. Equation (2) is simply $\Pr(V = 1, J = j) = \Pr(V = 1)\Pr(J = j \mid V = 1)$. If $a = 0$, unconditional correct richness is zero and q is undefined. An observed pool with no successes need not have $a = 0$; its matched-correct estimator is unavailable. For countably many classes the occupancy sums converge, because $1-(1-p)^k \le kp$ gives $C_k \le ka$ and $D_b \le b$. These quantities count expected observed classes at the stated budget, not all possible implementations.

Proof of Proposition 1. Let I_j indicate observing at least one correct output from class j in k independent draws. Its expectation is $1-(1-aq_j)^k$. Linearity of expectation gives C_k. The same argument for b draws from the conditional correct distribution yields D_b. All k draws fail with probability $(1-a)^k$, so pass@k depends only on a. For $q = (1, 0, \ldots)$, $D_b = 1$. For uniform mass on $J \ge 2$ classes, $D_b = J[1-(1-1/J)^b] > 1$ when $b \ge 2$. Hence identical correctness at every budget does not identify conditional breadth. □

More generally, spreading mass from a larger to a smaller conditional class, without reversing their order, cannot decrease D_b: $f(q) = 1-(1-q)^b$ is concave for $b \ge 2$. The same applies to C_k at fixed a and $k \ge 2$. This describes a change in a probability distribution; it does not assert that the spectral operator induces that change in every network.

For $a > 0$, the two richness endpoints are linked by the number of successful draws. Let $B \sim \mathrm{Binomial}(k, a)$ and set $D_0(q) = 0$. Conditioning on B gives

$$C_k(a, q) = \mathbb{E}\big[D_B(q)\big] = \sum_{b=0}^{k} \binom{k}{b} a^b (1-a)^{k-b} D_b(q). \tag{11}$$

Given $B = b$, the b correct classes are drawn from q, so their expected richness is $D_b(q)$. Thus total-budget richness combines how many correct samples are obtained with how broadly those samples are distributed. Matching the correct-draw budget isolates the latter quantity on the eligible tasks.

A.2 FINITE-SAMPLE ESTIMATORS

Suppose a task has n generated outputs, m of which pass its verifier, and correct-class counts m_j with $\sum_j m_j = m$. We compute

$$\widehat{\text{pass@}k} = 1 - \frac{\binom{n-m}{k}}{\binom{n}{k}}, \qquad 1 \le k \le n, \tag{12}$$

$$\widehat{C}_k = \sum_j \left[1 - \frac{\binom{n-m_j}{k}}{\binom{n}{k}}\right], \qquad 1 \le k \le n, \tag{13}$$

$$\widehat{D}_b = \sum_j \left[1 - \frac{\binom{m-m_j}{b}}{\binom{m}{b}}\right], \qquad 1 \le b \le m. \tag{14}$$

We set $\binom{u}{v} = 0$ for nonnegative integers $u < v$. Equation (12) is the standard pass@k estimator (Chen et al., 2021). Equations (13)–(14) are rarefaction estimators (Hurlbert, 1971). A uniformly chosen subset of k outputs misses correct class j with probability $\binom{n-m_j}{k}/\binom{n}{k}$. Summing its complement gives the exact expected subset richness conditional on the recorded pool; under independent sampling it estimates the corresponding population quantity. Conditioning on correct outputs gives the analogous calculation for D_b.

At $k = n$, $\widehat{C}_n$ is the number of observed correct classes. At $k = 1$, $\widehat{C}_1 = m/n = \widehat{\text{pass@}1}$. At $b = 1$, $D_1 = 1$ wherever defined. Thus neither C_1 nor D_1 distinguishes how probability is distributed among multiple correct classes. The plotted budgets are those present in the archived estimates; we do not reconstruct a missing pass@32 value by interpolation.

A.3 AST FINGERPRINTS AND CONCENTRATION

Python programs are extracted using the recorded first-fenced-block rule. The primary fingerprint parses the extracted program and conservatively normalizes its AST. Formatting and docstrings are removed; local identifiers are normalized only when scope handling is unambiguous. Function names, external identifiers, literals, and uncertain scopes are retained. Consequently, distinct fingerprints can reflect structural implementation variants within one algorithm. Conversely, a fingerprint does not establish functional equivalence beyond the recorded tests. Parse failures do not silently become new correct classes. Secondary exact-program and control-flow fingerprints provide different granularities.

An additional recorded concentration statistic is correct-output Simpson diversity,

$$\widehat{S} = 1 - \frac{\sum_j m_j(m_j - 1)}{m(m-1)}, \qquad m \geq 2. \tag{15}$$

It estimates the probability that two independently drawn correct outputs fall in different classes. Its aggregate uses tasks with enough correct samples and complete labels. It complements richness by emphasizing concentration, while sharing the limitations of the class definition. Exact-program, AST, and control-flow metrics need not move equally; the coarser control-flow comparisons do not establish a uniform increase in algorithmic families.

A.4 TASK AGGREGATION AND UNCERTAINTY

Pass@k and C_k are macro-averaged over the full task set. A marginal D_b mean uses $\mathcal{E}_b(M) = \{x : m_x(M) \geq b \text{ and all correct outputs have AST labels}\}$ for model M and reports its size. A paired contrast between M_1, M_2 averages task-level differences on $\mathcal{E}_b(M_1) \cap \mathcal{E}_b(M_2)$. Subtracting marginal means with different eligible sets is not this paired effect. Figure 5b pairs each round with initialization at $b = 8$; its eligible population can change between rounds. It is a pointwise trajectory, not a single fixed cohort.

The five-round MBPP intervals use 2,000 task-bootstrap resamples. Transfer intervals also use 2,000 resamples, pairing by exact task ID and conditioning on common eligibility. Percentile endpoints are the 2.5th and 97.5th percentiles. The analysis script records the means, intervals, populations, and input hashes. These pointwise intervals describe evaluation-task uncertainty conditional on the saved training runs, not variation across independent training runs; no multiplicity adjustment is applied.

B OPERATOR DERIVATION AND MATHEMATICAL SCOPE

B.1 REFERENCE LOSS AND POSITIONWISE SENSITIVITY

Let $\mathcal{T}_i$ contain the nonpadding completion targets of reference i. The scalar differentiated during calibration is

$$\ell_i(\theta) = -\frac{1}{|\mathcal{T}_i|} \sum_{u \in \mathcal{T}_i} \log p_\theta(y^\star_{iu} \mid x_i, y^\star_{i,<u}). \tag{16}$$

This loss is computed separately for each reference before collecting gradients. For selected module ℓ, let $h^{(\ell)}_{iu} \in \mathbb{R}^d$ be its native linear output at position u, written as a column here, and define $g^{(\ell)}_{iu} = \nabla_{h^{(\ell)}_{iu}} \ell_i(\theta_t)$. The second moment in Equation (4) averages the outer products of these gradients over every nonpadding position, including prompt and zero-gradient positions. Completion masking selects the loss targets; it does not prevent prompt activations from influencing subsequent completion predictions.

The statistic has a direct directional interpretation. For a unit vector v, perturbing one selected output position by ϵv gives

$$\ell_i\big(h^{(\ell)}_{iu} + \epsilon v\big) - \ell_i\big(h^{(\ell)}_{iu}\big) = \epsilon\, g^{(\ell)\top}_{iu} v + o(\epsilon), \tag{17}$$

where the notation denotes an intervention on that intermediate output, with the remaining computation propagated normally. Consequently,

$$v^\top C_{t,\ell} v = \frac{1}{M_\ell} \sum_{i,u,\text{nonpad}} \big(g^{(\ell)\top}_{iu} v\big)^2. \tag{18}$$

This averages squared first-order responses to individual positionwise perturbations. It does not include the cross-position terms arising from a coherent simultaneous perturbation of several outputs. Opposite gradient signs contribute positively, so $C_{t,\ell}$ measures sensitivity rather than a signed descent direction. It is an uncentered gradient second moment, not a Hessian of the reference loss.

When $\lambda_{\max}(C_{t,\ell}) > 0$, normalization produces $0 \preceq \overline{C}_{t,\ell} \preceq \mathbf{I}$ and a maximum eigenvalue of one. Empty or zero-signal calibration is rejected because the normalization would otherwise be undefined. The same anchor is reused, but the gradients and normalized statistic are recomputed at the current student parameters.

B.2 Proof of Proposition 2: proximal solution and bounds

For fixed $\overline{C}$ and row activation h, write the objective in Equation (5) as

$$F_h(z) = \frac{1}{2}\|z - h\|_2^2 + \frac{\tau}{2} z(\mathbf{I} - \overline{C})z^\top. \tag{19}$$

Its Hessian is $H = \mathbf{I} + \tau(\mathbf{I} - \overline{C})$. If $\mu_j \in [0, 1]$ are the eigenvalues of $\overline{C}$, the eigenvalues of H are $1 + \tau(1 - \mu_j) \in [1, 1 + \tau]$. Thus F_h is strongly convex and has a unique minimizer. Its first-order condition is

$$z - h + \tau z(\mathbf{I} - \overline{C}) = 0, \qquad zH = h. \tag{20}$$

It follows that $z = hH^{-1} = hT_\tau$. Diagonalizing H gives

$$T_\tau = U \operatorname{diag}\left(\frac{1}{1 + \tau(1 - \mu_j)}\right) U^\top, \tag{21}$$

which is Equation (6).

All gains are positive at finite τ, and lie in $[1/(1 + \tau), 1]$. Because T_τ is symmetric positive definite, its singular values are precisely these gains. For any row vector δ,

$$\frac{\|\delta\|_2}{1 + \tau} \le \|\delta T_\tau\|_2 \le \|\delta\|_2. \tag{22}$$

Moreover,

$$\|T_\tau - \mathbf{I}\|_2 = \max_j \frac{\tau(1 - \mu_j)}{1 + \tau(1 - \mu_j)} \le \frac{\tau}{1 + \tau}. \tag{23}$$

These establish Equation (7). The condition number is at most $1 + \tau$, and $T_0 = \mathbf{I}$.

B.3 Calibration stability

For normalized statistics A, B with spectra in $[0, 1]$, write $H_A = \mathbf{I} + \tau(\mathbf{I} - A)$ and $H_B = \mathbf{I} + \tau(\mathbf{I} - B)$. The inverse identity gives

$$T_\tau(A) - T_\tau(B) = H_A^{-1}(H_B - H_A)H_B^{-1} \tag{24}$$

$$= \tau T_\tau(A)(A - B)T_\tau(B). \tag{25}$$

Both outer factors have operator norm at most one. Submultiplicativity therefore yields

$$\|T_\tau(A) - T_\tau(B)\|_2 \le \tau\|A - B\|_2. \tag{26}$$

The argument requires neither an eigengap nor shared eigenvectors. This completes the proof of Proposition 2. □

The result concerns normalized statistics. To relate it to raw second moments, let $C, D \succeq 0$ with $c = \lambda_{\max}(C) \ge \gamma > 0$ and $d = \lambda_{\max}(D) \ge \gamma > 0$. Since $\|D\|_2 = d$ and $|c - d| \le \|C - D\|_2$,

$$\left\|\frac{C}{c} - \frac{D}{d}\right\|_2 \le \frac{\|C - D\|_2}{c} + \|D\|_2\left|\frac{1}{c} - \frac{1}{d}\right| \tag{27}$$

$$= \frac{\|C - D\|_2 + |d - c|}{c} \le \frac{2\|C - D\|_2}{\gamma}. \tag{28}$$

Combining the two estimates gives

$$\left\|T_\tau\left(\frac{C}{c}\right) - T_\tau\left(\frac{D}{d}\right)\right\|_2 \le \frac{2\tau}{\gamma}\|C - D\|_2. \tag{29}$$

This is a deterministic perturbation bound under a nonvanishing normalization scale. No sampling-concentration claim for the finite-anchor estimator is required.

B.4 Relative reference sensitivity

For a nonzero row vector h, define its sensitivity per unit squared norm as

$$s_{\overline{C}}(h) = \frac{h\overline{C}h^\top}{\|h\|_2^2}. \tag{30}$$

Equation (8) states that modulation cannot decrease this quantity. To prove it, write $\overline{C} = U\,\mathrm{diag}(\mu_j)U^\top$ and set

$$w_j = \frac{(hu_j)^2}{\|h\|_2^2}, \qquad r_j = \frac{1}{[1+\tau(1-\mu_j)]^2}. \tag{31}$$

The weights sum to one and $r_j > 0$. Direct substitution yields

$$s_{\overline{C}}(hT_\tau) - s_{\overline{C}}(h) = \frac{\mathbb{E}_w[\mu r]}{\mathbb{E}_w[r]} - \mathbb{E}_w[\mu] \tag{32}$$

$$= \frac{\mathrm{Cov}_w(\mu, r)}{\mathbb{E}_w[r]} \tag{33}$$

$$= \frac{\sum_{i,j} w_i w_j (\mu_i - \mu_j)(r_i - r_j)}{2\mathbb{E}_w[r]} \geq 0. \tag{34}$$

The final inequality follows because r_j is nondecreasing in μ_j. For $\tau > 0$, it is strictly increasing, so equality holds exactly when all eigendirections carrying nonzero energy from h share the same eigenvalue. At $\tau = 0$, modulation is the identity.

The relative increase does not require an increase in absolute sensitivity energy. Indeed, since $0 < r_j \leq 1$,

$$(hT_\tau)\overline{C}(hT_\tau)^\top = \sum_j \mu_j r_j (hu_j)^2 \leq \sum_j \mu_j (hu_j)^2 = h\overline{C}h^\top. \tag{35}$$

Thus modulation does not increase activation norm or absolute sensitivity energy, while sensitivity per unit squared norm does not decrease. This is a property of a local quadratic statistic; it does not assert a decrease in the original reference loss.

B.5 Folding, finite precision, and the projection control

For row activations and a linear module with weight W and bias b,

$$(xW^\top + b^\top)T = x(T^\top W)^\top + (T^\top b)^\top. \tag{36}$$

Thus replacing W and b by $T^\top W$ and $T^\top b$ implements the same local map. This is the algebra underlying temporary weight folding. The implementation computes products in float32 and casts back to the model dtype; the exact invertibility statements concern real arithmetic. The insertion point is the native K/V linear output before rotary position operations and any subsequent key normalization. Grouped-query dimensions come from each selected module's actual output dimension. The original weights are saved and restored before learning.

The projection design uses $P_r = U_{1:r}U_{1:r}^\top$, with eigenvectors ordered by decreasing eigenvalue and r equal to half the module output dimension in the completed run. It preserves retained directions and nulls the remainder. At $\tau = 1$, SPECTRUM's gains instead lie in $[1/2, 1]$. The two designs differ in spectrum and effective strength, so this comparison does not match their distance from the identity.

For completeness, the infinite-strength limit of the proximal family is

$$\lim_{\tau\to\infty} T_\tau = \sum_{j:\mu_j=1} u_j u_j^\top. \tag{37}$$

It projects onto the maximum-eigenvalue eigenspace of the normalized statistic, not an arbitrary prescribed top-r space. Finite-strength full rank is therefore the relevant property for the completed experiment.

B.6 Local operator properties and learned solution distributions

Positive gains preserve distinctions between inputs to the local linear map. Subsequent attention, nonlinear layers, vocabulary projection, stochastic decoding, and finite-data learning determine whether those distinctions affect output classes. The operator results establish bounded modulation, calibration stability, and relative sensitivity enrichment. The measurements of C_k and D_b establish how the resulting students distribute their correct outputs.

A separate idealization clarifies why learning from self-generated sequences is not automatically an improvement step. For a fixed prompt x, consider the completion-only, unnormalized sequence-NLL objective

$$L_t(\theta) = -\mathbb{E}_{Y\sim p_{\theta_t}(\cdot|x)}\big[\log p_\theta(Y \mid x)\big]. \tag{38}$$

When the sequence distribution is normalized and differentiation can be interchanged with summation, its gradient at the generating parameters is

$$\nabla_\theta L_t(\theta)|_{\theta=\theta_t} = -\sum_y p_{\theta_t}(y \mid x)\ \nabla_\theta \log p_\theta(y \mid x)|_{\theta=\theta_t} = -\ \nabla_\theta \sum_y p_\theta(y \mid x)\Big|_{\theta=\theta_t} = 0. \tag{39}$$

This identity assumes exact model-distribution sampling and the complete sequence log likelihood. The recorded learner in Equation (10) includes prompt targets and uses example-wise token averaging; a sequence-dependent length normalization need not preserve the zero-gradient identity. The practical loop also uses a decoding distribution, finite corpora, a fixed optimization schedule, and, for SPECTRUM, reference-calibrated generation. Its improvement and retention trajectories are measured empirically.

C Algorithm, information flow, and protocol

C.1 A complete round

The following sequence implements a complete round of SPECTRUM, separating the temporary generator from the native learner.

1. Fix synthesis prompts $\mathcal{Q}$, reference anchor $\mathcal{A}$, held-out evaluation tasks, decoding settings, and the initial checkpoint θ_0 before the loop.
2. At round t, calculate reference-completion loss under θ_t, collect selected K/V-output gradients, and form $\overline{C}_{t,\ell}$ on the unchanged anchor.
3. Construct $T_{t,\ell}$ with the fixed strength τ and temporarily fold the operators into the current checkpoint's native linear weights.
4. Sample S completions for each synthesis prompt. Keep every raw record. No execution outcome, reward, uniqueness judgment, or AST label determines membership or weight in $\mathcal{D}_t$.
5. Restore the original θ_t weights. Fit one LoRA adapter on $\mathcal{D}_t$, merge it into the base weights, and save θ_{t+1}.
6. Evaluate θ_{t+1} through common native inference, storing all outputs and verification labels. These labels do not affect the next training round. Repeat from step 2 using θ_{t+1}.

For Vanilla SD or SSD, the same generate–learn–update loop uses the corresponding standard or decoding-shaped policy, omitting reference calibration and weight folding. The anchor itself is never expanded with newly judged samples. It is reused as supervision at every calibration step, so "fixed" describes the information source rather than an initialization-only operation. In the completed study, there is no paired intervention-on generator diagnostic at every round; the native endpoints establish retained differences after learning, not the fraction of an immediate generator benefit transferred by SFT.

C.2 Data and prompts

The MBPP preparation starts from the full official splits. Exact normalized prompt overlaps with held-out prompts are removed before partitioning the training pool; the recorded excluded training

Table 5: **Completed training and evaluation settings.** The final 64-sample MBPP supplement is a separate sample pool; only 16-sample measurements exist for all historical rounds.

Component	Recorded setting
Backbone	Qwen/Qwen2.5-Coder-1.5B-Instruct; BF16, SDPA
Rounds / records per round	5 / $291 \times 16 = 4,656$
Calibration	50 fixed references; completion-only mean NLL; length limit 1,536
Geometry	Middle and last decoder layers; K/V outputs before RoPE; full second moment
SPECTRUM / projection control	$\tau = 1$ / rank fraction 0.5
LoRA	Rank 8, alpha 8, dropout 0.05; attention Q/K/V/O projections
SFT	1 epoch; learning rate 10^{-5}; weight decay 0.01; warm-up ratio 0.03; gradient clipping 1.0
SFT batching / length	Microbatch 1, accumulation 16; maximum 1,536 tokens
SFT target mask	All nonpadding causal targets, including prompt targets
Common generation	Temperature 0.8, top-p 0.95, top-k 0
SSD training generation	Temperature 1.5, top-p 0.8, top-k 20
MBPP generation limits	512 new tokens, 1,024 prompt tokens
Transfer generation limits	1,024 new tokens, 4,096 prompt tokens
Native evaluation draws	MBPP every round: 16; final supplement: 64; transfer: 16
Uncertainty	2,000 task-bootstrap resamples; one training run

IDs are 602, 704, and 872. The resulting sizes are 291 synthesis, 50 calibration, 30 validation, and 500 evaluation tasks. A Python signature stub is extracted mechanically from the reference AST to specify the interface. Its implementation and evaluation assertions are not placed in the synthesis/evaluation prompt. The calibration partition supplies complete reference solutions.

The fixed MBPP prompt body is:

```
[task description]
Implement this Python function.  Return only the complete
implementation.
[signature stub]
```

Brackets identify fields in a template, not missing manuscript results. The same template is used across methods; it does not request multiple strategies or select a strategy. Model-native chat formatting is applied by the tokenizer. Prompt and token provenance are stored in the experiment records.

HumanEval+ follows the recorded EvalPlus task interface and full base-plus tests, using release v0.1.10 with EvalPlus 0.3.1. APPS Intro is a fixed 200-task evaluation subset using the stored function-interface adapter. For stdin/stdout tasks, the prompt requests `solve(stdin: str) -> str`; the evaluator checks the returned output string against supplied cases using whitespace-token equality. This adapted interface, task subset, and evaluation budget are part of the result and differ from official benchmark leaderboard settings. No HumanEval+ or APPS references recalibrate the frozen MBPP students.

C.3 WHAT THE STUDENT OPTIMIZES

Let s denote the concatenated tokenized prompt and generated completion, and let $\mathcal{T}(s)$ contain its nonpadding causal targets. The recorded microbatch-one schedule accumulates example-level token-mean losses,

$$\ell_s(\theta) = -\frac{1}{|\mathcal{T}(s)|} \sum_{u \in \mathcal{T}(s)} \log p_\theta(s_u \mid s_{<u}), \qquad \mathcal{L}_{\mathrm{SFT}} = \frac{1}{|\mathcal{D}_t|} \sum_{s \in \mathcal{D}_t} \ell_s(\theta). \tag{40}$$

Gradient accumulation follows the recorded minibatch schedule. Including prompt targets and averaging each example by its own length are explicit choices, distinct from completion-only calibration and globally token-weighted corpus optimization. Every record receives the same outer example weight;

token normalization within each record follows its own length. Correctness and source subspace do not determine training weights. Truncation at the fixed context limit is a sequence-length constraint, not quality filtering. The completed training summaries record zero truncated training examples.

C.4 Information budgets and comparators

Vanilla SD and SSD do not use the reference anchor. SPECTRUM and the projection control use the same 50 references at each calibration step. Hence their direct comparison controls access to calibration information, whereas comparing either to Vanilla SD or SSD includes that difference in information use. SSD here adapts the published decoding idea to the small-model MBPP budget; it does not reproduce the original larger-scale results. UA-RL, which uses rewards for rare strategies, is discussed as related work but is not a completed numerical comparator in this archive.

D Complete measured results

D.1 Longitudinal MBPP results at 16 samples

Table 6 contains all recorded rounds. The initial SSD row is an independently sampled evaluation of the same starting model, retained separately from the original trajectory. Small differences in initialization estimates are sampling/protocol-stream differences, not different learned checkpoints. No 64-sample intermediate-round results are inferred from these rows.

D.2 Paired retention trajectories

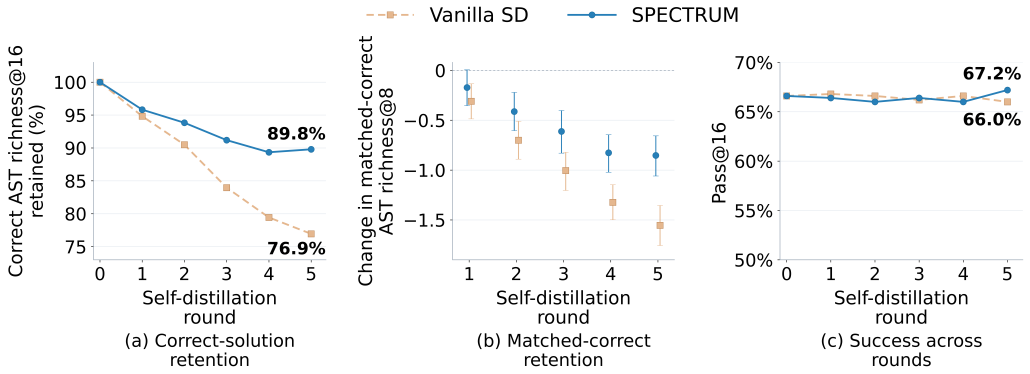


Figure 5: **Within-correct retention over successive updates.** Original 16-sample MBPP comparison of Vanilla SD and SPECTRUM. (a) C_{16} relative to initialization; (b) paired D_8 changes from initialization on each contrast's common eligible tasks, with pointwise 95% task-bootstrap intervals; (c) pass@16. Panels (a,c) show point estimates. The complete three-design comparison is Figure 3.

D.3 Final projection comparison and correct-count matching

Sparse D_{64} estimates are available on few fully correct tasks but do not form a primary claim: the paired SPECTRUM–Vanilla SD population is only 29 tasks. At the recorded 16-sample round-five endpoint, the SPECTRUM-minus-projection difference in exact-program D_4 is 0.312 with interval $[0.253, 0.377]$, and the AST difference is 0.293 $[0.233, 0.353]$. The coarser control-flow difference is 0.042 $[-0.011, 0.095]$. All three contrasts use 251 common eligible tasks and distinguish implementation structure from coarser control-flow variation.

D.4 Transfer contrasts and the projection control

The projection control's HumanEval+ scores are pass@1 52.74%, pass@16 84.76%, $C_{16} = 5.921$, and $D_4 = 3.473$ on 115 eligible tasks. Its APPS Intro scores are 14.88%, 34.50%, 1.880, and 3.685

Table 6: **Every completed native-student round on MBPP, 16 draws per task.** Full 500-task means for pass@1 and C_{16}; marginal D_4 means with eligibility. SSD is the separate continuation.

Method	Round	pass@1 (%)	C_{16}	D_4	Eligible
Initial	0	37.92	4.016	3.310	262
Vanilla SD	1	38.42	3.808	3.173	261
Vanilla SD	2	39.15	3.634	3.080	269
Vanilla SD	3	39.65	3.372	2.895	257
Vanilla SD	4	40.79	3.190	2.803	270
Vanilla SD	5	41.30	3.090	2.747	268
SPECTRUM	1	37.84	3.848	3.222	261
SPECTRUM	2	38.36	3.768	3.159	259
SPECTRUM	3	39.05	3.662	3.063	262
SPECTRUM	4	39.78	3.588	2.991	263
SPECTRUM	5	40.18	3.606	2.997	263
Projection control	1	37.92	3.692	3.144	257
Projection control	2	39.20	3.592	3.059	267
Projection control	3	39.40	3.314	2.889	256
Projection control	4	41.09	3.162	2.774	266
Projection control	5	41.35	3.066	2.690	262
SSD initial	0	37.81	3.992	3.311	260
SSD	1	37.80	3.846	3.232	260
SSD	2	36.81	3.710	3.200	250
SSD	3	36.60	3.724	3.196	256
SSD	4	37.09	3.752	3.181	257
SSD	5	35.96	3.710	3.268	255

Table 7: **Final 64-sample projection ablation.** Same 500 tasks, fixed reference access, and five-round training schedule. Marginal D_4 means are descriptive; eligible tasks are 321 and 323.

Generator	pass@1 (%)	pass@64 (%)	C_{64}	Retention (%)	D_4
Subspace projection	41.77	70.20	8.388	65.5	2.810
SPECTRUM	40.33	72.60	11.510	89.9	3.143

on 47 eligible tasks. The SPECTRUM-minus-projection paired D_4 effects are 0.039 $[-0.021, 0.102]$ on 109 HumanEval+ tasks and 0.067 $[-0.003, 0.142]$ on 42 APPS tasks. The paired intervals specify the precision of these smaller transfer contrasts.

Table 8: **Paired final MBPP richness contrasts, SPECTRUM minus Vanilla SD.** Common eligible tasks are selected separately for each correct-draw budget. Intervals are the archived 95% task-bootstrap intervals.

Endpoint	Paired difference	95% interval	Tasks
C_{64}	3.004	$[2.568, 3.494]$	500
D_4	0.295	$[0.249, 0.343]$	316
D_8	0.780	$[0.665, 0.889]$	296
D_{16}	1.816	$[1.582, 2.055]$	254

Table 9: **Frozen-student transfer after MBPP learning.** Sixteen samples per task, without transfer training or recalibration. Success and C_{16} average over all tasks. D_4 is the marginal mean over each row's eligible tasks; paired common-task contrasts appear in the text. Higher values indicate greater success or richness; eligibility is descriptive.

Benchmark	Student	pass@1 (%)	pass@16 (%)	C_{16}	D_4	Eligible
HumanEval+ (164)	Initial	53.09	85.37	5.768	3.410	116
	Vanilla SD	53.47	86.59	5.848	3.422	120
	SPECTRUM	51.94	86.59	6.000	3.497	114
APPS Intro (200)	Initial	15.47	41.50	2.070	3.772	50
	Vanilla SD	15.19	35.50	1.930	3.632	46
	SPECTRUM	13.97	34.00	1.845	3.721	43

Table 10: **Paired transfer effects of SPECTRUM relative to Vanilla SD.** Differences use all tasks for pass@1 and C_{16}, and common eligible tasks for D_4. Intervals use 2,000 paired task-bootstrap resamples of the saved task records. Students receive no transfer training or recalibration.

Benchmark	Endpoint	Difference	95% interval	Tasks
HumanEval+	pass@1 (pp)	-1.524	$[-3.088, 0.076]$	164
	C_{16}	0.152	$[-0.128, 0.439]$	164
	D_4	0.080	$[0.021, 0.142]$	111
APPS Intro	pass@1 (pp)	-1.219	$[-1.937, -0.530]$	200
	C_{16}	-0.085	$[-0.210, 0.055]$	200
	D_4	0.104	$[0.007, 0.205]$	42

E CROSS-MODEL AND CROSS-BENCHMARK BASELINE CONTEXT

The completed matrix evaluates Qwen2.5-Coder-1.5B-Instruct, DeepSeek-Coder-6.7B-Instruct (Guo et al., 2024), Gemma-3-4B-IT (Gemma Team et al., 2025), and Qwen3-8B (Yang et al., 2025) on MBPP, HumanEval+, APPS Intro, CodeContests (Li et al., 2022), and LiveCodeBench (Jain et al., 2024). Table 11 gives pass@16; Table 12 reports the archived task-bootstrap intervals. These are initial-model evaluations, independent of the sample pools used for the five-round and frozen-student comparisons. Consequently, even the same initial checkpoint can have different finite-sample estimates across tables.

All twenty cells completed generation and verification. Task counts are 500, 164, 200, 165, and 200, respectively. MBPP pass@16 is estimated from 64 samples per task; the other benchmarks use 16. HumanEval+ uses official EvalPlus verification; APPS Intro, CodeContests, and LiveCodeBench use the recorded native adapted verifiers. The Gemma–LiveCodeBench run uses eager attention. These protocol choices define the reported benchmark setting.

Table 11: **Completed cross-model baseline matrix.** Pass@16 (%) for initial models across five coding benchmarks. Each column has a common recorded evaluation setting across models. These rows measure initial-model performance, not trained SPECTRUM students.

Model	MBPP	HumanEval+	APPS Intro	CodeContests	LiveCodeBench
Qwen2.5-Coder-1.5B	66.51	85.98	36.50	0.61	22.00
DeepSeek-Coder-6.7B	67.48	87.80	50.50	6.06	22.50
Gemma-3-4B	58.07	73.78	38.00	3.03	24.00
Qwen3-8B	62.38	93.29	51.00	5.45	40.50

Table 12: **Archived uncertainty for the baseline matrix.** Pointwise 95% task-bootstrap intervals for pass@16 on the percentage scale, conditional on the recorded model and sample pool.

Model	Benchmark	pass@16 (%)	95% interval
Qwen2.5-Coder-1.5B	MBPP	66.51	[62.83, 70.31]
	HumanEval+	85.98	[80.49, 90.85]
	APPS Intro	36.50	[30.00, 43.00]
	CodeContests	0.61	[0.00, 2.42]
	LiveCodeBench	22.00	[16.50, 28.00]
DeepSeek-Coder-6.7B	MBPP	67.48	[63.69, 71.39]
	HumanEval+	87.80	[82.91, 92.68]
	APPS Intro	50.50	[44.00, 57.00]
	CodeContests	6.06	[2.42, 9.70]
	LiveCodeBench	22.50	[17.00, 28.50]
Gemma-3-4B	MBPP	58.07	[53.64, 62.21]
	HumanEval+	73.78	[67.06, 79.88]
	APPS Intro	38.00	[31.50, 44.50]
	CodeContests	3.03	[0.61, 6.06]
	LiveCodeBench	24.00	[18.50, 30.00]
Qwen3-8B	MBPP	62.38	[58.31, 66.59]
	HumanEval+	93.29	[89.02, 96.95]
	APPS Intro	51.00	[44.00, 58.00]
	CodeContests	5.45	[2.42, 9.09]
	LiveCodeBench	40.50	[33.50, 47.50]

F ACTUAL EQUAL-CORRECT-COUNT EXAMPLES

The compact archive retains correctness and program fingerprints but removes program text. We therefore show actual class-frequency records rather than reconstructing purported code samples. For each sign of the SPECTRUM-minus-Vanilla SD observed AST-count difference, choose the numerically lowest HumanEval task ID with equal correct counts of at least four. This rule yields HumanEval/3 for a positive example and HumanEval/2 for a negative example (Figure 6). Selection conditions on the sign and is illustrative, not an estimate of prevalence.

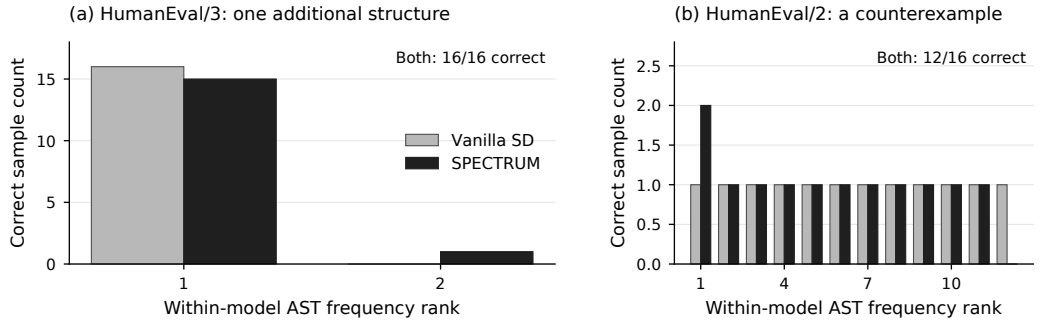


Figure 6: **The same number of correct outputs can conceal different structures.** Actual saved round-five HumanEval+ samples. (a) Both methods pass all 16 draws on HumanEval/3. Vanilla SD has one AST class with count 16; SPECTRUM has counts 15 and 1, giving $\widehat{D}_4 = 1$ and 1.25. (b) Both pass 12 of 16 draws on HumanEval/2; Vanilla SD has twelve singleton classes, while SPECTRUM has one doubleton and ten singletons. Their $\widehat{D}_4$ values are 4 and 3.909. Classes are ranked separately within each method, so horizontal position does not assert matching class identity. Both cases follow the stated lowest-ID rule.

These examples instantiate the measurement distinction without supplying semantic algorithm labels. At equal observed n and m, every finite-sample pass@k estimator agrees between methods, while

the correct-class occupancy can differ in either direction. Equal estimated pass curves are not proof that the underlying true success probabilities are identical.

G RESOURCES AND REALIZED SAMPLE LENGTHS

All methods use the same 23,280 generated training records over five rounds. Table 13 records their realized lengths. SPECTRUM generates 9.9% more completion tokens than Vanilla SD; equal record count is therefore not equal token count or FLOPs. Matching the number of correct evaluation samples does not control training length or compute. It answers a distributional question about the saved students.

Table 13: **Training-generation resources across five rounds.** Counts are summed from saved round budgets; mean length is generated tokens divided by records. Length-capped records remain in training. SFT token counts include prompt targets under the recorded loss mask.

Method	Records	Generated tokens	Mean length	Capped	SFT tokens
Vanilla SD	23,280	4,859,928	208.76	617	6,373,288
SSD	23,280	5,812,400	249.67	1,498	7,325,760
SPECTRUM	23,280	5,342,368	229.48	833	6,855,728
Projection control	23,280	4,727,089	203.05	619	6,240,449

For a selected module of output dimension d and M gradient positions, dense second-moment accumulation costs $O(Md^2)$ and eigendecomposition costs $O(d^3)$, excluding the reference forward/backward passes. Dense folding of a $d \times p$ weight costs $O(d^2p)$. These operations are performed during calibration/construction, not at each generation token. The final student has the original dense architecture. Archived runtime records cover different hardware and stages; we make no matched wall-clock or GPU-memory efficiency claim from their combination.

H EXPLORATORY ONE-ROUND PILOT

This earlier experiment uses Qwen2.5-Coder-1.5B-Instruct with 128 synthesis tasks, one generated completion per task, one generate–learn cycle, and five SFT epochs. Evaluation uses 64 held-out MBPP tasks and 64 samples per task. Five SFT epochs within a cycle are not five self-distillation rounds. Its task counts and training schedule differ from the main five-round study, so the two experiments are reported separately.

Recorded extraction and verification. The pilot report is a post-hoc sensitivity analysis of code extraction. The original strict rule accepted only completions consisting of exactly one code fence; accompanying text caused most answers to fail extraction. The recorded first-fence analysis instead extracted the first Python, `py`, or untagged fenced block and re-verified it against the task tests using local execution. All four designs use this same revised rule. The archived summary and artifact checksums document these results; this revision copies the reported values and intervals without regenerating programs or rerunning verification. Raw pilot metric records are not part of the compact evidence bundle.

The pilot combines higher high-budget success and matched-correct richness than SSD with slightly lower single-draw correctness. The full five-round experiment subsequently measures a different operating point: SPECTRUM has higher correctness than SSD, while SSD has higher C_{16} and marginal D_4. Different synthesis budgets, task sets, and optimization schedules preclude attributing this difference solely to loop length.

Table 14: **Complete exploratory pilot comparison.** One generate–learn cycle, 64 MBPP tasks, 64 samples each. Values are archived rounded summaries; bold marks the largest estimate at shown precision. AST and exact-program statistics are structural proxies.

Metric ↑	Vanilla SD	SSD	Projection	SPECTRUM
Correct samples / 4,096	1,508	**1,511**	1,510	1,501
pass@1 (%)	36.82	**36.89**	36.87	36.65
pass@8 (%)	56.49	**56.79**	55.78	56.18
pass@32 (%)	61.61	61.66	60.84	**63.11**
pass@64 (%)	62.50	62.50	60.94	**65.63**
Solved tasks / 64	40	40	39	**42**
Exact-program unique fraction	0.802	0.797	0.796	**0.814**
AST unique fraction	0.622	0.622	0.616	**0.649**
AST Simpson diversity	0.864	0.863	0.869	**0.872**
Correct-count-matched AST D_4	3.393	3.391	3.409	**3.422**

Table 15: **Reported paired pilot effects relative to SSD.** SPECTRUM minus SSD; pointwise task-bootstrap intervals copied from the archived report. These are not newly computed intervals or uncertainty across independent training runs.

Endpoint	Difference	95% interval
Correct-count-matched AST D_4	$+0.0312$	$[0.0095, 0.0550]$
Exact-program unique fraction	$+0.0083$	$[0.0009, 0.0171]$
AST Simpson diversity	$+0.0094$	$[0.0034, 0.0162]$
pass@1 (pp)	-0.244	$[-0.854, 0.342]$
pass@64 (pp)	$+3.125$	$[0, 7.813]$

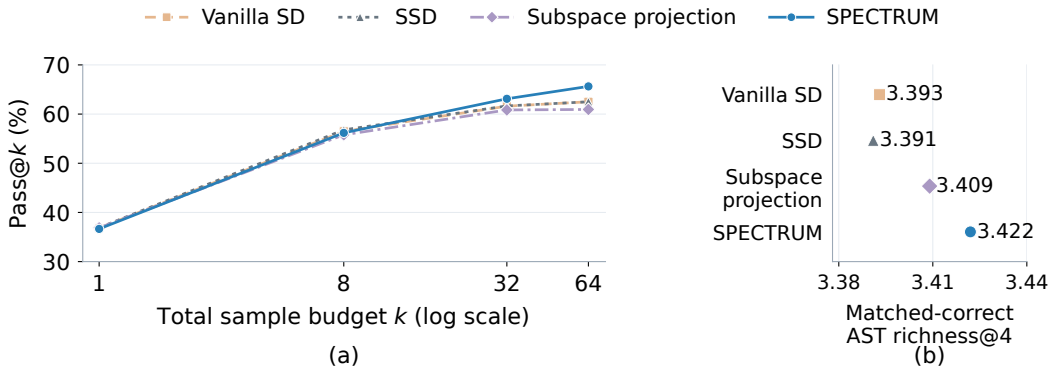


Figure 7: **Complementary evidence from the one-round pilot.** (a) Pass@k at recorded sampling budgets; (b) correct-count-matched AST D_4. All four generation designs are included. Marks use archived rounded point estimates from 64 tasks and 64 samples per task; marginal error bars are unavailable. Table 15 gives the recorded paired uncertainty. This protocol is separate from the five-round study.

I RELATION TO ADJACENT SELF-DISTILLATION SETTINGS

Sequence-level and token-feedback distillation. Our learner uses sampled sequences as cross-entropy targets. Demonstration-conditioned on-policy distillation instead supplies dense teacher feedback conditioned on privileged context. SPECTRUM shapes generation through fixed reference calibration, then trains one native student on its raw corpus. Looped Self-Distillation names the outer generate–learn–update process.

Iterative self-improvement. Iterative context/rationale learning, code-data synthesis, and teacher refresh have been studied with different feedback mechanisms (Rao et al., 2023; Zhang et al., 2025; Sang et al., 2026). Our study combines a fixed-information protocol, a continuous spectral generation intervention, and longitudinal measurements of conditional correct-implementation richness. Demonstration-conditioned diversity analyses (Nicolicioiu et al., 2026) address a different teacher-feedback construction; the present results measure contraction and retention through repeated learning from unselected raw corpora.

Evidence sources. The longitudinal evidence comprises the full MBPP five-round archive, its separate final 64-sample evaluation, the SSD continuation, and frozen-student HumanEval+/APPS Intro results. The four-model, five-benchmark matrix in Appendix E reports independent initial-model evaluations. The earlier one-round pilot is reported separately in Appendix H; software-smoke runs are excluded from research evidence.