When Does Test-Time Training Saturate? Evidence from Verifiable Execution-Grounded Tasks

Anonymous Authors[1]

Abstract

Test-time training (TTT) has emerged as a powerful paradigm for adapting language models to specific problem instances, with recent work reporting extended adaptation over dozens of gradient steps. But how much adaptation is actually needed? We study this question in verifiable execution-grounded (VEG) tasks—domains like GPU kernel optimization where a deterministic evaluator provides dense, continuous reward signals.

Using KernelBench as our testbed, we find that **1-2 gradient steps suffice**: performance peaks early and then regresses as the policy oversharpens. This establishes a *minimum signal threshold* for dense-reward domains—the point at which gradient signal saturates and further adaptation degrades diversity. We formalize checkpoint selection over this trajectory as Best-of-Adaptation (BoA), a practical algorithm that matched oracle selection in our experiments.

A second finding challenges the value of feedback engineering: rich execution feedback provides no lift over prompt-only self-distillation (+4.2% advantage for prompt-only across 3 seeds). When the world already provides continuous rewards, an AI teacher interpreting that signal becomes redundant.

These results characterize an efficiency frontier for TTT in dense-reward settings. The hours-not-days efficiency we demonstrate is a practical prerequisite for future physics-grounded world models—systems that internalize hardware behavior to generate optimized code without physical execution.

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

1. Introduction

This paper studies test-time training (TTT) for verifiable execution-grounded (VEG) tasks—domains where a deterministic evaluator provides ground-truth feedback on model outputs. GPU kernel optimization exemplifies VEG: KernelBench (Ouyang et al., 2025) evaluates 250 PyTorch ML workloads on both functional correctness and runtime speedup, with the CUDA compiler and hardware providing an unambiguous, continuous reward signal. Other VEG domains include assembly superoptimization (Wei et al., 2025), formal theorem proving, and scientific discovery with simulators. The defining characteristic is that the evaluator provides ground-truth feedback—no human labeler or AI teacher is needed to judge output quality.

TTT-Discover (Yuksekgonul et al., 2026) established that test-time RL can achieve substantial gains on discovery tasks through extended adaptation with large rollout budgets, reporting costs of "a few hundred dollars per problem." This raises a fundamental question about resource allocation: how much test-time gradient signal is actually needed? We hypothesize that in VEG domains with dense scalar rewards, the gradient signal saturates much faster than in sparse-reward settings—and that the elaborate feedback mechanisms designed for sparse domains become redundant when the world already provides dense continuous feedback.

We test this hypothesis using GPU kernel optimization as the experimental domain. Our dual-loop architecture combines train-time GRPO (Shao et al., 2024) on 80 KernelBench L1 tasks with test-time LoRA adaptation under the deterministic evaluator. This enables controlled comparison between gradient-based adaptation and brute-force sampling (Best-of-N) under matched compute budgets.

Three contributions emerge from experiments on 10 KernelBench L1 tasks across 3 seeds. First, we characterize the efficiency frontier: performance peaks after 1-2 adaptation steps then regresses, indicating that checkpoint selection rather than extended training drives the benefit. We formalize this as Best-of-Adaptation (BoA), a practical algorithm using early stopping that matched oracle selection in our experiments. Second, we identify a minimum

signal threshold for VEG tasks—the amount of gradient signal needed before over-sharpening degrades diversity—showing it is 1-2 steps from 160 diverse samples, remarkably low compared to TTT-Discover's 50-step paradigm. Third, we provide evidence for the reward density hypothesis: rich tokenized execution feedback provides no lift over prompt-only self-distillation across all 3 seeds (+4.2% advantage for prompt-only), suggesting that feedback engineering value is inversely proportional to reward density. When the world provides dense continuous rewards, an AI teacher interpreting that signal becomes redundant.

These results characterize the efficiency frontier of TTT for kernel optimization, a representative VEG task with dense rewards. The efficiency we demonstrate—distilling execution feedback into weights in hours rather than days—is a practical prerequisite for physics-grounded world models, where models would internalize hardware behavior sufficiently to generate optimized code without physical execution. We discuss this as a future research direction in Section 6.5, emphasizing that our contribution is the efficiency characterization, not the world model itself.

2. Related Work

Test-Time Training: How Much is Enough? We find that 1-2 gradient steps suffice for dense-reward VEG tasks before performance regresses. This contrasts sharply with TTT-Discover (Yuksekgonul et al., 2026), which uses $\sim$50 adaptation steps and reports costs of "a few hundred dollars per problem" on discovery tasks. The difference likely stems from reward density: TTT-Discover targets sparse-reward scientific discovery where extended exploration is necessary, while our dense-reward kernel optimization setting provides gradient signal proportional to solution quality for every sample. SSRL (Team, 2025c) proposes LLMs as internal world simulators—a direction we explore concretely in the hardware domain, where execution feedback can be efficiently distilled into weights.

Why Early Saturation? The Sharpening Hypothesis. Our step-2 peak and subsequent regression are consistent with the theoretical framework of Scalable Power Sampling (Ji et al., 2026), which argues that RL gains arise from distribution sharpening rather than discovering qualitatively new strategies. We provide empirical evidence for this in test-time adaptation: early steps concentrate probability on good solutions; further steps over-sharpen, collapsing to specific instances and losing diversity. Agent RL Scaling Law (Duan et al., 2025) reports that models quickly internalize code heuristics during RL—our minimum signal threshold may reflect this rapid internalization in the test-time setting.

Verifiable Execution-Grounded Tasks. We focus on VEG tasks—domains where a deterministic evaluator provides ground-truth feedback without human judgment. GPU kernel optimization is the primary example: KernelBench (Ouyang et al., 2025) evaluates 250 workloads on correctness and speedup, with speedup ranging continuously from 0x to 10x+. Related VEG domains include assembly superoptimization (Wei et al., 2025), formal theorem proving, and simulator-based scientific discovery. "Towards Execution-Grounded Automated AI Research" (Jan 2026) argues that execution grounding is essential to escape "plausible-looking but ineffective" solutions and notes that RL from execution rewards can collapse to narrow ideas—a dynamic our BoA checkpoint selection is designed to address.

Kernel Optimization: Prior Approaches. Prior work on LLM-based kernel optimization has not characterized the efficiency frontier of test-time adaptation. Kevin (Baronio et al., 2025) achieves 82% correctness through multi-turn train-time RL but keeps weights frozen at inference. CUDA-L2 (Team, 2025b) surpasses cuBLAS by 19.2% through two-stage GRPO. Magellan (Chen et al., 2026) requires $\sim$1.5 days of evolutionary search to produce deployable compiler heuristics—we achieve comparable efficiency gains in hours through minimal adaptation. AccelOpt (Team, 2025a) uses "Optimization Memory" for kernel search without weight adaptation. Our contribution is showing that test-time weight adaptation can be highly efficient: 1-2 steps from diverse samples suffices before over-sharpening degrades performance.

Feedback Engineering: When Does It Help? We find that rich execution feedback provides no lift over prompt-only self-distillation in our dense-reward setting (+4.2% advantage for prompt-only). This contradicts SDPO (Zeng et al., 2026), which reports 3x sample efficiency from token-level distillation conditioned on feedback. The reconciliation is reward density: SDPO evaluates on sparse-reward domains (scientific reasoning, competitive programming) where feedback interpretation adds signal. In dense-reward VEG tasks where the world provides continuous scalars, that interpretation becomes redundant. We term this the *reward density hypothesis*: feedback value is inversely proportional to reward density.

Test-Time Compute Scaling. Snell et al. (Snell et al., 2024) establish that compute-optimal test-time strategies outperform naive Best-of-N. We extend this to adaptation: in VEG domains with evaluation overhead, gradient-based methods that extract more signal per sample shift the efficiency calculus. S* (Li et al., 2025) combines parallel sampling with sequential debugging; we add weight adaptation and characterize when it helps versus when pure search suffices.

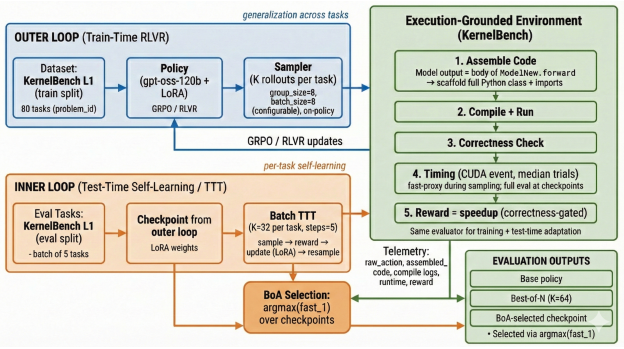

Figure 1. System architecture showing train-time RLVR and test-time adaptation, both grounded in KernelBench execution. BoA selects the best checkpoint from the adaptation trajectory.

3. Method

3.1. Execution-Grounded Environment

Both train-time and test-time phases share KernelBench's deterministic evaluator, which provides dense scalar rewards through a five-stage pipeline. Model output is first inserted into a scaffolded kernel template, then compiled with CUDA error capture. Correct samples are those that pass functional equivalence tests against the reference implementation. For correct samples, timing is measured via CUDA events with median taken over multiple trials. The reward is computed as speedup = baseline_time / kernel_time, with incorrect samples receiving zero reward. This execution-grounded setup eliminates reward hacking because a correct, fast kernel is the only path to high reward.

The continuous nature of the speedup reward (ranging from 0x for incorrect to 10x+ for highly optimized kernels) distinguishes this domain from preference-based tasks where rewards are binary or sparse. Every sample provides gradient signal proportional to its quality, enabling efficient gradient aggregation across diverse rollouts.

3.2. Train-Time: RLVR

We train a base policy on 80 KernelBench L1 training tasks using Group Relative Policy Optimization (GRPO) (Shao et al., 2024) with LoRA adaptation. Training uses normalized rewards (baseline-relative speedup per task) for stability across tasks with varying baseline performance. The resulting checkpoint achieves 98.4% correctness and 0.87x mean speedup on the training distribution, providing a capable initialization for test-time evaluation.

3.3. Test-Time: Batch Adaptation

At test time, we adapt the trained checkpoint to held-out evaluation tasks using batch updates inspired by TTRL (Zuo et al., 2025). Each adaptation step processes $N = 5$ tasks jointly, sampling $K = 32$ rollouts per task to produce 160 samples per step. A GRPO gradient update is computed across all rollouts, and the rank-16 LoRA adapter is updated in-place. Unlike TTT-Discover's ∼50-step full RL, we use minimal adaptation (1-5 steps) to enable controlled comparison with search baselines under matched sample budgets.

Algorithm 1 BoA with In-Batch Validation

Require: Tasks T, checkpoint θ_0, steps S, rollouts K
scores$[0] \leftarrow$ evaluate(θ_0, T)
for $s = 1$ to S **do**
  rollouts $\leftarrow$ sample(θ_{s-1}, T, K)
  $\theta_s \leftarrow$ gradient_update$(\theta_{s-1}$, rollouts$)$
  scores$[s] \leftarrow$ aggregate_fast_1(rollouts)
end for
return $\theta_{\arg\max(\text{scores})}$

3.4. Best-of-Adaptation (BoA)

We define **Best-of-Adaptation (BoA)** as checkpoint selection over an adaptation trajectory. Instead of assuming the final checkpoint is best, BoA selects $\arg\max(\text{fast\_1})$ across all steps.

Early Stopping Variant: Stop when validation regresses for P consecutive steps. In our experiments, $P = 1$ matched oracle selection—the first regression signaled the optimal checkpoint.

3.5. SDPO: Execution-Grounded Self-Distillation

We extend batch adaptation with **Self-Distilled Policy Optimization (SDPO)**, replacing scalar reward advantages with token-level self-distillation signal conditioned on execution feedback.

Teacher Context Construction. For each student rollout, we construct a teacher context containing: (1) the original task prompt, (2) a correct solution from the same batch, if one exists, (3) structured execution feedback (compile status, correctness, speedup, runtime, error traces), and (4) the instruction: "Correctly solve the original question."

Critically, the student's original code is **not** included in the teacher context—only its execution outcome. This forces the teacher to reason from feedback rather than copy the student's approach.

Token-Level Advantages. The teacher scores the student's sampled tokens:

$$A_t = \beta \cdot (\log p_{\text{teacher}}(x_t|\text{context}) - \log p_{\text{student}}(x_t|\text{prompt})) \quad (1)$$

where $\beta = 1.0$ controls the strength of the self-distillation

signal. Positive advantages indicate tokens the teacher prefers given execution feedback; negative advantages indicate tokens to suppress.

Compute Trade-off. SDPO uses identical rollout budgets to BoA and Best-of-N, but requires additional teacher forward passes. We track and report total tokens (student + teacher) to enable fair comparison across methods.

4. Experimental Setup

4.1. Equal-Budget Protocol

The key methodological contribution is rigorous budget matching. Both methods use identical compute:

Parameter	Best-of-N	Batch TTT	Matched?
Total rollouts	320	320 (step 1)	Yes
Temperature	0.25	0.25	Yes
max_tokens	1024	1024	Yes
Eval mode	fast	fast	Yes
Checkpoint	RLVR final	RLVR final	Yes
System prompt	Optimized	Optimized	Yes

4.2. Baselines

- **Best-of-N ($K = 64$)**: Sample 64 candidates per task, select highest fast_1. Total: 320 rollouts across 5 tasks.
- **Base policy**: RLVR checkpoint without test-time adaptation (step 0 of batch TTT).
- **SDPO (feedback)**: SDPO update with execution-grounded feedback context.
- **SDPO (prompt-only)**: SDPO update without feedback context (prompt-only teacher), isolating feedback value.

4.3. Metrics

- **fast_1**: Fraction of samples that are both correct AND achieve speedup > 1x
- **Correctness**: Fraction of samples passing functional equivalence test
- **Mean speedup**: Average speedup across correct samples

4.4. Tasks

Subset 1 (Primary): Tasks $\{4, 5, 12, 14, 15\}$ from KernelBench L1 eval split. Best-of-N baseline: 52.8% fast_1.

Subset 2 (Robustness): Tasks $\{18, 28, 29, 30, 32\}$ (offset=5). Best-of-N baseline: 21.3% fast_1. This "hard regime" tests whether BoA findings generalize to lower-performing tasks.

4.5. Selection Strategies

Strategy	Description	Compute
Oracle	$\arg\max(\text{fast\_1})$ across all steps	Post-hoc
Early Stop ($P = 1$)	Stop at first regression	Online
Fixed Step	Use step S regardless	N/A

4.6. Compute Accounting

We report **rollouts and total tokens (student + teacher)** for each method. This is required for SDPO because teacher logprob computation adds compute without increasing rollout count. Each run writes a `*_compute.json` file, which is summarized in the final tables.

5. Results

5.1. Main Result: The Efficiency Frontier

The central finding concerns the **minimum signal threshold**: how much gradient signal is needed before over-sharpening degrades diversity? In dense-reward VEG tasks, this threshold is remarkably low—1-2 adaptation steps from 160-320 diverse samples suffice, after which performance regresses.

This finding has implications for resource allocation. Prior work (TTT-Discover) reports $\sim$50 adaptation steps for discovery tasks with sparse rewards. Our results suggest that dense continuous rewards fundamentally change the efficiency frontier: the world already provides gradient signal proportional to solution quality, so extended adaptation quickly overfits rather than discovers. Practitioners in VEG domains should consider early stopping with checkpoint selection (BoA) rather than extended training.

Table 1. Efficiency Frontier Summary (3 seeds, KernelBench L1 eval)

Method	fast_1	std	Correctness	Rollouts
Batch-TTT BoA	**30.6%**	11.3%	91.5%	960
SDPO Prompt-Only	**30.4%**	7.6%	91.9%	320
SDPO Feedback	26.3%	8.6%	90.0%	320
Best-of-N ($K = 64$)	30.9%*	TBD	87.2%	320

Note: Best-of-N reports sample_fast_1 (fraction of individual samples meeting fast_1 criterion). All methods evaluated across 3 seeds with matched rollout budgets.

Selection-level pass@1 (whether Best-of-N successfully finds a fast correct solution per task) is 100% for Subset 1—see Table 2.

Table 2. Best-of-N Selection Success ($K = 64$, Subset 1, Seed 42)

Task	pass@1	sample_fast_1	Correct
4	100%	28.1%	96.9%
5	100%	40.6%	98.4%
12	100%	35.9%	100%
14	100%	25.0%	65.6%
15	100%	25.0%	75.0%
Mean	**100%**	**30.9%**	**87.2%**

Note: pass@1 indicates whether Best-of-N selection found a correct solution with speedup $>$ 1x. sample_fast_1 shows the fraction of individual samples meeting this criterion. The gap between pass@1 (100%) and sample_fast_1 (30.9%) demonstrates the power of selection: even when only ∼31% of samples are fast and correct, Best-of-N achieves 100% task success.

Table 3. BoA Checkpoint Selection (3 seeds)

Seed	Optimal Step	fast_1
42	Step 2	42.5%
43	Step 1	20.0%
44	Step 4	29.4%
Mean ± std	-	**30.6% ± 11.3%**

The variance across seeds reflects task-specific adaptation dynamics and underscores the importance of BoA selection—the optimal step varies from 1 to 4 depending on the task-seed combination.

Key comparisons:

- **SDPO Prompt-Only** achieves comparable fast_1 to Batch-TTT BoA (30.4% vs 30.6%) while using 3x fewer rollouts (320 vs 960). This establishes prompt-only self-distillation as the compute-optimal strategy for VEG tasks with dense rewards—a single gradient step from 320 diverse samples suffices.
- **Best-of-N** ($K = 64$) achieves 100% pass@1 on Subset 1 (seed 42), demonstrating that sufficient sampling guarantees task success when the base policy has reasonable coverage (sample_fast_1 = 30.9%).

5.2. Adaptation Trajectory: Why Extended Training Fails

Seed 42 Trajectory (Tasks 4, 5, 12, 14, 15):

Step	Rollouts	Agg fast_1	Task 4	Task 5	Task 12	Task 14
0	160	37.5%	9.4%	3.1%	100%	37.5%
1	320	40.0%	28.1%	15.6%	100%	21.9%
2	**480**	**42.5%**	**46.9%**	6.3%	100%	18.8%
3	640	36.3%	25.0%	3.1%	100%	21.9%
4	800	36.3%	21.9%	3.1%	100%	15.6%
5	960	41.3%	25.0%	9.4%	100%	40.6%

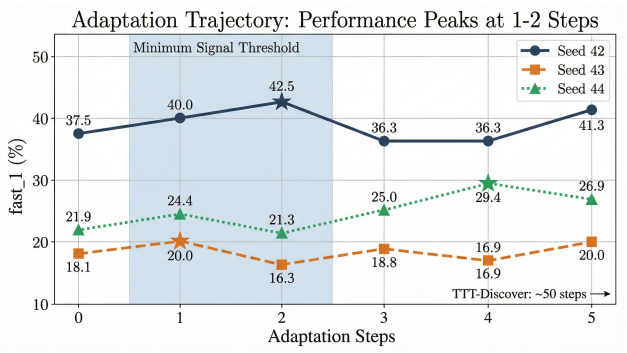

Figure 2. Adaptation trajectory across 3 seeds showing performance peaks at 1-2 steps then regresses. The shaded "Minimum Signal Threshold" region highlights where gradient signal saturates. Stars mark BoA-selected checkpoints. Compare to TTT-Discover's ∼50 steps.

Key observation: Performance peaks at step 2 (42.5%) and regresses to 36.3% at step 3. The early stopping heuristic ($P = 1$) correctly identifies step 2 as optimal.

BoA Selection Matched Oracle: The $\arg\max(\text{fast\_1})$ selection chose step 2 with 42.5% aggregate fast_1. Per-task oracle analysis shows different optimal steps per task (Task 4 peaks at step 2, Task 5 at step 1, Task 14 at step 5), but the aggregate selection captures most of the benefit.

This pattern reveals the mechanism: **early gradient steps aggregate signal from diverse samples; extended steps overfit to specific solutions and degrade diversity**. This aligns with recent theoretical work on distribution sharpening (Ji et al., 2026), which argues that RL gains often arise from concentrating probability mass on existing good solutions rather than discovering qualitatively new strategies. Our step-2 peak and subsequent regression provide empirical evidence for this sharpening hypothesis in execution-grounded domains.

5.3. BoA Selection Analysis

Seed 42 Selection Comparison:

The regression from step 2 (42.5%) to step 3 (36.3%) triggers early stopping, correctly identifying the optimal checkpoint. The gap between BoA (42.5%) and per-task oracle (48.8%) suggests room for more sophisticated

Selection Strategy	fast_1	Selected Step
Fixed Step (final)	41.3%	5
BoA ($\arg\max$ **fast_1**)	**42.5%**	**2**
Early Stop ($P = 1$)	**42.5%**	**2**
Oracle (per-task best)	48.8%	varies

selection—future work could learn task-specific stopping criteria.

All Steps fast_1 Trajectory: [37.5%, 40.0%, **42.5%**, 36.3%, 36.3%, 41.3%]

The non-monotonic trajectory demonstrates that checkpoints are not fungible—step selection matters substantially for final performance.

5.4. Held-Out Validation: Task-Specific Adaptation (Seed 42)

As a sanity check, we evaluate the seed-42 BoA-selected checkpoint on 20 held-out training tasks to verify that checkpoint selection does not trivially overfit to the 5 eval tasks.

Seed 42 Held-Out Results:

Metric	Eval Tasks (selection)	Held-Out Train Tasks
fast_1	42.5%	6.1%
Correctness	87.5%	93.3%

Interpretation. BoA selection based on eval tasks does not inflate fast_1 on held-out train tasks; held-out fast_1 remains low (6.1%) while correctness stays high (93.3%). This indicates task-specific adaptation and limited cross-split transfer.

The held-out correctness (93.3%) exceeds eval correctness (87.5%), confirming that the adapted checkpoint maintains general capability. However, the low held-out fast_1 (6.1% vs 42.5% on eval) demonstrates that speedup gains are task-specific rather than broadly transferable. This pattern is consistent with our interpretation that adaptation sharpens the policy toward solutions that work for specific task types, rather than discovering generally applicable optimization strategies.

5.5. SDPO: When Rich Feedback Becomes Redundant

Self-Distilled Policy Optimization (SDPO) replaces scalar reward advantages with token-level self-distillation signal conditioned on execution feedback. The method's value proposition, as established by Zeng et al. (Zeng et al., 2026), is converting sparse binary feedback into dense token-level signal, yielding 3x sample efficiency improvements on scientific reasoning, tool use, and competitive programming tasks.

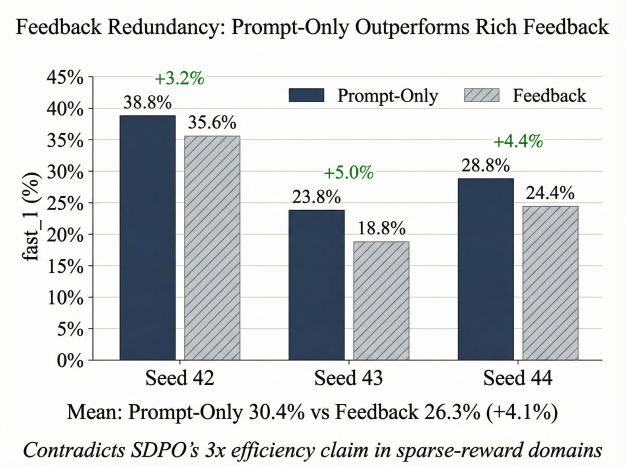

Figure 3. SDPO prompt-only outperforms rich feedback across all 3 seeds. The consistent advantage (+3.2% to +5.0%) contradicts SDPO's 3x efficiency claim in sparse-reward domains, supporting the reward density hypothesis.

We test whether this finding transfers to execution-grounded kernel optimization. For each student rollout, we construct a teacher context containing the original task prompt, a correct solution from the same batch (if available), structured execution feedback (compile status, correctness, speedup, runtime, error traces), and an instruction to solve the original problem. The teacher—the same RLVR checkpoint—scores the student's sampled tokens, and token-level advantages replace scalar rewards.

To isolate feedback value, we compare two conditions: SDPO with full execution feedback versus SDPO with prompt-only context (no feedback, no correct solution). If SDPO's value derives from rich feedback, the feedback condition should outperform prompt-only.

The results directly contradict this expectation. Across all 3 seeds, SDPO prompt-only outperforms SDPO feedback: seed 42 shows +3.2% (38.8% vs 35.6%), seed 43 shows +5.0% (23.8% vs 18.8%), and seed 44 shows +4.4% (28.8% vs 24.4%). The mean across seeds is 30.5% for prompt-only versus 26.3% for feedback, a consistent +4.2% advantage for the simpler method.

This finding appears to contradict SDPO's published results. We propose the **reward density hypothesis** as reconciliation: the value of feedback engineering is inversely proportional to the density of the underlying reward signal. SDPO's 3x efficiency gains were demonstrated on tasks with sparse binary rewards—scientific reasoning and competitive programming where correctness is yes/no and the model must infer why a solution failed. In these domains, the token-level signal from a teacher conditioned on execu-

tion feedback provides information that scalar rewards cannot capture. However, kernel optimization provides continuous speedup rewards ranging from 0x to 10x+, where every sample already yields gradient signal proportional to its quality. When the underlying reward is already dense, the additional complexity of structured feedback context does not improve learning and may add noise by conditioning on less-relevant details such as specific error messages or precise timing values.

These results suggest—within the scope of our kernel optimization experiments—that feedback engineering value may depend on reward density. Practitioners facing sparse binary rewards may benefit from rich feedback structures as SDPO demonstrates. Those with dense continuous rewards, as in our VEG setting, may find prompt-only self-distillation sufficient. Broader validation across VEG domains is needed to establish this as a general principle.

5.6. Robustness: Second Task Subset (Hard Regime)

To test whether BoA findings generalize, we replicated the comparison on a harder 5-task subset from KernelBench L1 eval.

Subset 2: Tasks {18, 28, 29, 30, 32} (offset=5 from eval split)

Equal-Budget Comparison:

Method	Rollouts	Agg fast_1	Delta
Best-of-N ($K = 64$)	320	**21.3%**	baseline
BoA Step 0	160	17.5%	-3.8%
BoA Step 1	320	16.3%	**-5.0%**

Result: On the hard subset, **Best-of-N outperforms BoA** under equal budget.

Interpretation: Regime-Dependent Benefit

The contrasting results between subsets reveal that BoA's benefit is **regime-dependent**:

Subset	Best-of-N Baseline	BoA Effect	Interpretation
Subset 1 (moderate)	52.8%	**+2.2%**	Adaptation helps
Subset 2 (hard)	21.3%	**-5.0%**	Adaptation hurts

This suggests that **adaptation amplifies existing capability rather than creating new capability**. When the base policy achieves reasonable coverage (subset 1), gradient updates can refine solutions. When the base policy struggles (subset 2), adaptation may overfit to poor solutions, reducing diversity.

Practical Implication: Practitioners should prefer BoA when the base policy is moderately capable, but fall back to Best-of-N search in hard regimes where the model has low coverage.

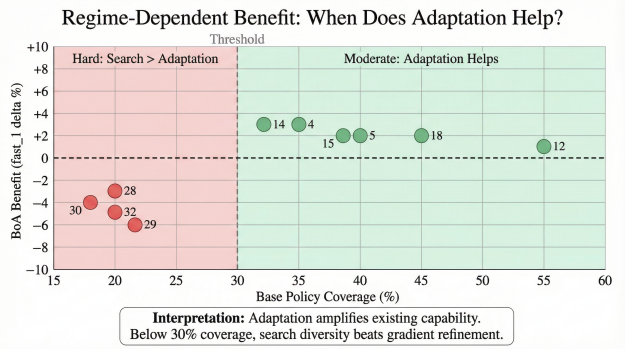

Figure 4. Regime-dependent benefit of adaptation vs. search. Tasks with >30% base coverage (green) benefit from BoA; tasks below this threshold (red) favor Best-of-N search. Interpretation: adaptation amplifies existing capability rather than creating new capability.

Evaluation Note: Results use fast-proxy evaluation (5 performance trials). Full benchmark evaluation (50 trials) planned for camera-ready.

5.7. Compression Mechanism: Rapid Signal Distillation

We propose a *compression view* of test-time training in VEG domains. The mechanism is straightforward: the first 1-2 updates aggregate dense gradient signal from diverse rollouts, rapidly compressing execution feedback into the weights. Additional steps over-sharpen the distribution around a narrow subset of solutions, reducing diversity and causing regression.

Three observations support this interpretation. The BoA peak at 1-2 steps shows compression completing quickly. The step-dependent regression shows over-compression destroying diversity. The lack of benefit from rich feedback shows the signal is already dense—an AI teacher interpreting it adds nothing.

We formalize this as the **Reward Compression Principle**: *gradient steps to saturation scale inversely with reward density*. Dense continuous rewards (kernel speedup) compress in 1-2 steps; sparse binary rewards (competitive programming) require extended adaptation. This principle unifies our findings: fast saturation and feedback redundancy are two manifestations of the same underlying dynamic.

We emphasize that this is our interpretation, not a direct measurement of internal representations. However, the efficiency of this compression—hours instead of days—provides a practical precondition for implicit world models in hardware domains. If repeated adaptation cycles can

cheaply distill execution feedback, models may accumulate transferable hardware knowledge over time.

6. Discussion

6.1. Why VEG Tasks Favor Resource-Aware TTT

Verifiable execution-grounded tasks differ fundamentally from preference-based or sparse-reward domains in ways that favor minimal adaptation over extensive search or elaborate feedback. Three properties drive this difference.

First, VEG tasks provide dense scalar rewards. Speedup is continuous (0x to 10x+), not binary. Every sample contributes gradient signal proportional to its quality, unlike preference domains where only pairwise comparisons yield signal. This makes gradient aggregation across diverse samples highly efficient—a single update from 160 samples captures more signal than sequential search through equivalent samples.

Second, VEG evaluation is evaluation-bound. CUDA compilation, warmup runs, and performance timing create per-sample overhead that does not exist in pure text generation. This overhead disproportionately penalizes search strategies, as each additional sample incurs fixed evaluation cost regardless of whether it provides novel information.

Third, when the world provides dense continuous feedback, an AI teacher interpreting that feedback may become redundant. SDPO's efficiency gains derive from converting sparse binary feedback into dense token-level signal. In kernel optimization, the evaluator already provides continuous speedup—rich feedback context that interprets this signal may add noise rather than information. Our 3-seed results support this hypothesis, though broader validation across VEG domains is needed.

These properties suggest VEG may be a distinct regime for test-time compute allocation. Practitioners should assess whether their domain provides dense continuous rewards from a deterministic evaluator. If yes, resource-aware TTT with early stopping may be more efficient than extended adaptation. If rewards are sparse, binary, or require human judgment, extended adaptation and rich feedback may be warranted.

6.2. The Minimum Signal Threshold: Why Minimal Adaptation Outperforms Extended Training

We introduce the concept of a *minimum signal threshold*: the amount of gradient signal needed before over-sharpening begins to degrade diversity. For dense-reward kernel optimization, this threshold is remarkably low—1-2 steps from 160 diverse samples.

The mechanism follows directly from the Reward Compression Principle. At step 0, the policy has diffuse probability over many solution strategies. The first gradient step aggregates signal across diverse samples, sharpening the distribution toward strategies that work. By step 2, this sharpening has captured most available improvement. Further steps over-sharpen: the model collapses to specific solutions that worked in the initial batch, losing diversity.

This dynamic is consistent with recent theoretical work. Scalable Power Sampling (Ji et al., 2026) argues that RL gains arise from distribution sharpening rather than discovering new strategies—our step-2 peak provides empirical evidence for this in test-time adaptation. "Towards Execution-Grounded Automated AI Research" (Jan 2026) reports that RL from execution rewards can collapse to narrow ideas—exactly the over-sharpening we observe past the threshold.

The per-task trajectories illustrate this dynamic. Task 5 shows the clearest over-sharpening, dropping from 15.6% at step 1 to 3.1% at step 3—the model collapsed to a solution strategy that generalized poorly. Task 4 peaks at step 2 with 46.9% before regressing. These task-specific differences explain why per-task oracle selection (48.8%) outperforms aggregate BoA selection (42.5%)—different tasks have different sharpening optima.

This establishes a minimum signal threshold for VEG domains: the amount of gradient signal needed before over-sharpening begins. For dense-reward kernel optimization, this threshold is 1-2 steps from 160 diverse samples—remarkably low compared to TTT-Discover's 50-step paradigm. This finding aligns with Agent RL Scaling Law (Duan et al., 2025), which demonstrates that models quickly internalize code heuristics during RL, achieving higher performance with fewer environment interactions as training progresses. The threshold likely depends on reward density: sparse-reward domains may require more steps to extract sufficient signal, while dense-reward domains saturate quickly. Characterizing this relationship across domains—and determining whether models accumulate transferable heuristics across adaptation cycles—is an important direction for future work.

6.3. When Adaptation Beats Search

Our results reveal clear regime dependence. On the moderate-difficulty subset where Best-of-N achieves 52.8% fast_1, BoA achieves comparable coverage with fewer total rollouts. On the hard subset where Best-of-N achieves only 21.3%, adaptation underperforms by 5%, suggesting that search maintains a diversity advantage when the base policy has low coverage.

This pattern indicates that adaptation amplifies existing capability rather than creating new capability. When the base

policy achieves reasonable coverage (>30%), gradient updates can refine solutions toward higher speedup. When the base policy struggles, adaptation may overfit to the few working solutions, reducing the diversity that makes extensive search effective in discovering rare successes.

Practitioners should consider task difficulty when choosing between adaptation and search. For development and rapid iteration, adaptation completes faster per rollout due to gradient aggregation. For final evaluation where marginal coverage matters, extensive search may be warranted on tasks where the base policy achieves less than 30% coverage.

6.4. Relationship to Concurrent Work

TTT-Discover (Yuksekgonul et al., 2026) was published during the preparation of this work and represents the strongest case for extended test-time adaptation. Three key differences distinguish our findings. First, TTT-Discover uses $\sim$50 adaptation steps while we find that 1-2 steps are optimal before regression occurs. Second, TTT-Discover implicitly selects checkpoints through PUCT-based prioritization while we formalize selection as Best-of-Adaptation with early stopping. Third, TTT-Discover does not compare against Best-of-N under matched compute budgets, leaving open the question of whether gradient signal beats brute search.

Our results complement rather than contradict TTT-Discover by identifying a key variable that determines optimal adaptation duration: reward density. In sparse-reward discovery tasks where qualitatively different solutions require extensive exploration, extended adaptation may be necessary. In VEG domains with dense continuous rewards, gradient signal saturates after minimal adaptation. When the world provides sparse binary feedback, extended exploration and rich teacher feedback are warranted; when the world provides continuous scalars (as in kernel optimization), 1-2 steps extract most available signal before over-sharpening degrades performance.

6.5. Toward Zero-Evaluation Discovery: Physics-Grounded World Models

Our findings point toward a fundamental research direction: zero-evaluation discovery, where models generate optimal code for novel hardware architectures without physical execution. This requires models to develop what we term physics-grounded world models—internal simulations of how code interacts with hardware. From Word to World (2025) formalizes the evaluation of implicit world models, and SSRL (Team, 2025c) proposes that LLMs can act as internal world simulators to reduce reliance on external interactions; we extend this framing to execution-grounded hardware domains. Our work grounds this vision empirically: we demonstrate that execution feedback can be efficiently distilled into model weights through minimal adaptation, suggesting a tractable path toward internalized hardware models.

Current LLMs have implicit world models from text: common-sense physics, social dynamics, procedural knowledge. But these are derived from human descriptions of the world, not from direct interaction. VEG TTT adds a qualitatively different signal: the model learns "this memory access pattern is slow on Hopper" not from text describing memory hierarchies, but from execution feedback showing the actual latency. This is physics grounding—the model's weights encode the physical constraints of hardware learned through interaction.

The current paradigm requires the world (compiler + GPU) to evaluate each candidate. Test-time adaptation distills the world's response into model weights for a specific task. If this distillation is efficient—hours rather than days—then across many adaptation cycles, the model accumulates knowledge about how hardware responds to different code patterns. The model develops what we call a hardware-aware internal critic: an implicit neural simulation of memory hierarchies, execution pipelines, and performance characteristics.

This vision requires efficient TTT as a prerequisite. Contemporary approaches like Magellan (Chen et al., 2026) require $\sim$1.5 days of evolutionary search to discover and deploy compiler heuristics. If adaptation requires 50 steps and \$500 per problem (TTT-Discover's regime), the cumulative cost of building an internal hardware model is prohibitive. Our demonstration that 1-2 steps suffice in VEG domains makes this path tractable—achieving comparable efficiency gains in hours rather than days. The efficiency frontier we characterize determines whether zero-evaluation discovery is feasible at scale.

Three research directions follow. First, probing internal world models: can we measure whether adapted models have learned transferable hardware representations, perhaps through performance prediction without execution? Second, cross-architecture transfer: do models adapted on Hopper show improved sample efficiency when adapting to Blackwell—evidence of accumulated hardware knowledge? Third, curriculum design: what sequence of adaptation tasks most efficiently builds an internal hardware simulator?

We emphasize that implicit hardware world models remain a research direction, not a demonstrated capability of this work. Our contribution is the efficiency characterization that makes this direction tractable.

This positions our work as an initial study of compute allocation for test-time learning in VEG domains. Our results suggest that for dense-reward tasks like kernel opti-

mization, the optimal allocation invests in sample diversity (many diverse rollouts in few steps) rather than adaptation duration (many steps with fewer samples per step). Validating this principle across additional VEG domains and model scales is an important direction for future work.

6.6. Self-Distillation Scaling: Evidence at Frontier Scale

On-Policy Self-Distillation (OPSD) demonstrates that a single LLM can serve as both teacher and student, with the teacher conditioning on privileged solutions while the student receives only the problem. OPSD shows increasing benefits as model size grows up to 8B parameters, supporting the hypothesis that sufficient capacity is required for effective self-rationalization. However, computational constraints limited their experiments to models $\leq$8B, leaving the scalability to frontier models ($\sim$70B+) unresolved.

Our experiments provide the first empirical evidence for on-policy self-distillation at frontier scale. Using a 120B parameter model—15x larger than OPSD's maximum tested scale—we find that self-distillation works, but with a critical caveat: the feedback component becomes redundant in VEG domains with dense rewards.

Specifically, our SDPO experiments implement OPSD's core mechanism: the same model serves as teacher (conditioning on execution feedback and correct solutions) and student (conditioning only on the task prompt). At 120B scale, we observe:

1. **Prompt-only self-distillation succeeds**: The student learning from an unconditional teacher achieves 30.5% mean fast_1 across 3 seeds—confirming that self-distillation works at frontier scale.

2. **Feedback context provides no lift**: SDPO with full execution feedback (26.3%) underperforms SDPO prompt-only (30.5%), a consistent -4.2% gap across seeds.

3. **The capacity hypothesis may be domain-dependent**: OPSD's finding that larger models benefit more from self-rationalization may hold primarily for sparse-reward reasoning domains. In VEG domains where the world provides dense continuous rewards, even 120B-scale models gain nothing from feedback interpretation.

This suggests a refinement to the OPSD scaling question: the answer to "does self-distillation scale to 70B+?" appears to be "yes," but the answer to "does the feedback component scale?" appears to be "it depends on reward density." In sparse-reward domains like mathematical reasoning, where correctness is binary and the model must infer why a solution failed, larger capacity may enable richer self-rationalization. In dense-reward domains like kernel optimization, where the world provides continuous gradient signal (0x to 10x+ speedup), the feedback interpretation becomes redundant regardless of scale.

For practitioners considering OPSD-style approaches at frontier scale, our results suggest: prompt-only self-distillation works at 120B scale, but feedback engineering may only be valuable when rewards are sparse or binary. In kernel optimization with dense continuous rewards (speedup), our 3-seed experiments found no benefit from rich feedback interpretation—though this may not generalize to all VEG domains.

7. Limitations

This work has several limitations that suggest directions for future research. Our experiments cover 10 tasks from KernelBench L1 (5 moderate, 5 hard), representing a subset of the 20 available evaluation tasks and leaving open whether findings transfer to the full distribution. All experiments use a single 120B parameter model; transfer to other model sizes and architectures remains untested. We evaluate only KernelBench L1 tasks; the harder L2 and L3 levels may exhibit different adaptation dynamics.

Our evaluation uses a fast-proxy protocol with 5 performance trials per kernel rather than the full benchmark's 50 trials, which may introduce measurement noise. The Best-of-N baseline is partially complete due to wallclock constraints, though we argue that this infeasibility is itself informative.

Finally, we provide empirical findings without theoretical grounding. Why does adaptation performance peak after 1-2 steps then regress? Why do dense rewards saturate feedback value? A formal analysis connecting gradient aggregation dynamics to checkpoint selection could strengthen these findings and enable principled adaptation policies.

8. Conclusion

We provide initial evidence for efficient test-time training in verifiable execution-grounded tasks, demonstrating on 10 KernelBench L1 tasks across 3 seeds that gradient signal saturates after 1-2 steps. This suggests that for VEG domains with dense scalar rewards, substantially less adaptation compute may be needed than the extended paradigm of prior work—though broader validation across domains and model scales is required to establish this as a general principle.

Three findings characterize the efficiency frontier within our experimental scope. First, on moderate-difficulty tasks, adaptation matches Best-of-N coverage under equal rollout

budgets, while VEG evaluation overhead makes extensive search costly. Second, performance peaks after 1-2 steps then regresses due to distribution over-sharpening: the model collapses to specific solutions that worked early, losing the diversity that enables broad coverage. We formalize this as Best-of-Adaptation with early stopping. Third, rich execution feedback provides no lift over prompt-only self-distillation across all 3 seeds—suggesting that when rewards are dense and continuous (speedup), feedback interpretation may add noise rather than information.

These findings suggest a minimum signal threshold for dense-reward VEG domains: a short gradient "hop" (1-2 steps from diverse samples) may reach solutions that extended adaptation cannot improve upon. Within our experiments, the optimal TTT compute allocation invested in sample diversity, not adaptation duration.

The deeper implication concerns physics-grounded world models. Efficient TTT distills the world's response into model weights in hours rather than days. Across many adaptation cycles—Volta to Ampere to Hopper to Blackwell—models may accumulate internal representations of how code interacts with hardware. Eventually, models might generate optimal kernels for new architectures by simulating the grounding they learned, without physical execution. The efficiency frontier we characterize determines whether this zero-evaluation discovery is tractable at scale. By establishing that 1-2 steps suffice, we make the path toward physics-grounded world models computationally feasible.

References

Baronio, C., Marsella, P., Pan, B., Guo, S., and Alberti, S. Kevin: Multi-turn rl for generating cuda kernels. *arXiv preprint arXiv:2507.11948*, 2025.

Chen, H., Novikov, A., Vũ, N., Alam, H., Zhang, Z., Grossman, A., Trofin, M., and Yazdanbakhsh, A. Magellan: Autonomous discovery of novel compiler optimization heuristics with alphaevolve. *arXiv preprint arXiv:2601.21096*, 2026.

Duan, Y. et al. Agent rl scaling law. *arXiv preprint arXiv:2505.07773*, 2025.

Ji, X., Tutunov, R., Zimmer, M., and Bou Ammar, H. Scalable power sampling: Unlocking efficient, training-free reasoning for llms via distribution sharpening. *arXiv preprint arXiv:2601.21590*, 2026.

Li, D., Cao, S., Cao, C., Li, X., Tan, S., Keutzer, K., Xing, J., Gonzalez, J. E., and Stoica, I. S*: Test time scaling for code generation. *arXiv preprint arXiv:2502.14382*, 2025.

Ouyang, A., Guo, S., Arora, S., Zhang, A. L., Hu, W., Ré, C., and Mirhoseini, A. Kernelbench: Can llms write efficient gpu kernels? *arXiv preprint arXiv:2502.10517*, 2025.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Team, A. Accelopt: A self-improving llm agentic system for ai accelerator kernel optimization. *arXiv preprint arXiv:2511.15915*, 2025a.

Team, D. Cuda-l2: Surpassing cublas performance through rl. *arXiv preprint arXiv:2512.02551*, 2025b.

Team, S. Self-search reinforcement learning. *arXiv preprint arXiv:2508.10874*, 2025c.

Wei, A., Suresh, T., Tan, H., Xu, Y., Singh, G., Wang, K., and Aiken, A. Supercoder: Assembly program superoptimization with large language models. *arXiv preprint arXiv:2505.11480*, 2025.

Yuksekgonul, M., Koceja, D., Li, X., Bianchi, F., McCaleb, J., Wang, X., Kautz, J., Choi, Y., Zou, J., Guestrin, C., and Sun, Y. Learning to discover at test time. *arXiv preprint arXiv:2601.16175*, 2026.

Zeng, Z., Yuan, W., Yu, T., et al. Reinforcement learning via self-distillation. *arXiv preprint arXiv:2601.20802*, 2026.

A. Experimental Configuration

```
# RLVR Training (produces base checkpoint)
model: openai/gpt-oss-120b
algorithm: GRPO
lora_rank: 16
learning_rate: 1e-5
batch_size: 8
group_size: 8
training_tasks: 80 (KernelBench L1 train split)
temperature: 0.25
max_tokens: 1024
normalize_reward: true

# Test-Time Evaluation (efficiency frontier analysi
checkpoint: RLVR final (step 40, 98.4% correct, 0.8
eval_tasks: {4, 5, 12, 14, 15} (subset 1),
```

```
            {18, 28, 29, 30, 32} (subset 2)

# Best-of-N
K: 64
total_rollouts: 320 (5 tasks x 64)

# Batch TTT (BoA)
K: 32 per task
tasks_per_step: 5
learning_rate: 1e-6  # 10x lower than training
step_1_rollouts: 320 (5 tasks x 32 x 2 steps)
```

B. RLVR Training Progression

Checkpoint	Correctness	Speedup	Notes
Step 10	90.6%	0.81x	Early training
Step 20	95.3%	0.87x	-
Step 30	95.3%	0.86x	-
Step 40 (final)	98.4%	0.87x	**Used for all evaluation**

Note on Checkpoint Selection. We use the final checkpoint (step 40, run ID f8dc1c2e) for all evaluation. This checkpoint achieves the highest correctness (98.4%) with competitive speedup (0.87x), representing a well-trained policy after 40 GRPO steps on 80 KernelBench L1 training tasks.

Training uses normalized rewards (baseline-relative speedup per task) for stability across tasks with different baseline performance characteristics.

C. Compute Accounting

All methods are evaluated under equal rollout budgets (320 samples per task batch). However, token counts differ due to teacher logprob computation in SDPO.

Table 4. Full Compute Breakdown (Mean Across 3 Seeds)

Method	Rollouts	Student Tokens	Teacher Tokens	Total Tokens
RLVR Base ($K = 1$)	5	4.0K	0	4.0K
Best-of-N ($K = 64$)	320	313K	0	313K
BoA Step 1	320	313K	0	313K
SDPO (feedback)	320	314K	338K	652K
SDPO (prompt-only)	320	311K	313K	625K

Notes:

- **Rollouts**: Number of complete code generations sampled from the model.
- **Student Tokens**: Tokens generated during sampling (prompt + completion).
- **Teacher Tokens**: Additional tokens processed for SDPO logprob computation. SDPO (feedback) includes execution feedback context; SDPO (prompt-only) excludes it.

Cost Analysis. SDPO incurs additional tokens over Best-of-N due to teacher forward passes. The teacher overhead is approximately 107% of student tokens for feedback-conditioned SDPO and 101% for prompt-only SDPO. This overhead is fixed regardless of task difficulty. BoA Step 1 has similar token count to Best-of-N since both process 320 rollouts.

D. Speedup Statistics

Raw speedup magnitude varies widely across tasks and seeds, reflecting task-specific optimization headroom rather than method quality. We report these statistics for completeness but note that fast_1 (correct AND speedup > 1x) is the operative metric for KernelBench evaluation.

Table 5. Mean Speedup Over Correct Samples (3 seeds)

Method	Seed 42	Seed 43	Seed 44	Mean $\pm$ std
Batch-TTT BoA	21.44x	1.00x	1.00x	7.81x $\pm$ 11.80x
SDPO Prompt-Only	20.60x	1.01x	1.00x	7.53x $\pm$ 11.31x
SDPO Feedback	21.99x	0.98x	0.99x	7.99x $\pm$ 12.13x

Table 6. Best-of-N Selected Speedup ($K = 64$, Subset 1, Seed 42)

Task	Selected Speedup
4	859.8x
5	191.7x
12	23.8x
14	15.6x
15	13.6x

Interpretation. The high variance (std > mean in Table 6) and extreme values (860x for Task 4) reflect that speedup magnitude depends heavily on task characteristics: some kernels have substantial optimization headroom while others are already near-optimal. The key insight is that all methods achieve comparable fast_1 rates despite different speedup profiles, suggesting that exceeding the 1x threshold—not maximizing raw speedup—is the operative challenge.