ASDA: Automated Skill Distillation and Adaptation for Financial Reasoning

Tik Yu Yim
The University of Hong Kong
Hong Kong SAR, China
tyyim@connect.hku.hk

Wenting Tan
The University of Hong Kong
Hong Kong SAR, China
wt212796@connect.hku.hk

Sum Yee Chan
The University of Hong Kong
Hong Kong SAR, China
sumychan@hku.hk

Tak-Wah Lam
The University of Hong Kong
Hong Kong SAR, China
twlam@cs.hku.hk

Siu Ming Yiu
The University of Hong Kong
Hong Kong SAR, China
smyiu@cs.hku.hk

Abstract

Adapting large language models (LLMs) to specialized financial reasoning typically requires expensive fine-tuning that produces model-locked expertise. Training-free alternatives have emerged, yet our experiments show that leading methods (GEPA and ACE) achieve only marginal gains on the FAMMA financial reasoning benchmark, exposing the limits of unstructured text optimization for complex, multi-step domain reasoning. We introduce Automated Skill Distillation and Adaptation (ASDA), a framework that automatically generates structured skill artifacts through iterative error-corrective learning without modifying model weights. A teacher model analyzes a student model's failures on financial reasoning tasks, clusters errors by subfield and error type, and synthesizes skill files containing reasoning procedures, code templates, and worked examples, which are dynamically injected during inference. Evaluated on FAMMA, ASDA achieves up to +17.33 pp improvement on arithmetic reasoning and +5.95 pp on non-arithmetic reasoning, substantially outperforming all training-free baselines. The resulting skill artifacts are human-readable, version-controlled, and compatible with the Agent Skills open standard, offering any organization with a labeled domain dataset a practical and auditable path to domain adaptation without weight access or retraining.[1]

CCS Concepts: • **Computing methodologies** → **Natural language processing**; **Artificial intelligence**; *Knowledge representation and reasoning*.

Keywords: LLM Adaptation, Financial Reasoning, Skill Distillation, Training-Free Adaptation, Agent Skills

1 Introduction

Financial reasoning poses a distinctive challenge for general-purpose LLMs: it demands simultaneous mastery of multi-step quantitative calculation and deep domain-specific judgment, a combination that pure math or pure knowledge benchmarks do not jointly test [7, 17]. Evaluations across multiple financial benchmarks confirm a persistent performance ceiling: FAMMA reveals that standard, non-reasoning frontier models achieve only 38–45% overall accuracy across eight financial subfields [17];[2] and FinBen finds that while LLMs handle information extraction well, they consistently struggle with advanced reasoning and complex financial QA [7]. FAMMA's error analysis finds that *domain-knowledge gaps* dominate model errors [17], as models misapply financial concepts to the wrong context or lack the expertise to select the correct procedure.

These are not failures that more parameters alone will fix—they require targeted advances in domain reasoning. The standard remedy—domain-specific fine-tuning—is costly, produces *model-locked expertise* that becomes obsolete with each model release, and depends on supervision resources that many organizations lack [15, 18]. This is especially problematic in regulated industries such as financial services, legal, and healthcare, where organizations deploy commercial LLMs via black-box API without weight access. Automated prompt optimization offers a training-free alternative, but methods like GEPA [2] and ACE [1] optimize *flat text strings*—monolithic instruction blocks that lack the modularity and executability required for multi-step reasoning across diverse financial subdomains, as our FAMMA experiments confirm (Section 5). The missing abstraction is not a better prompt, but an *executable skill*: a modular, self-contained reasoning procedure that can be independently composed, tested, and updated for each target domain.

We introduce **Automated Skill Distillation and Adaptation (ASDA)**, a framework that automatically generates executable agent skills from error analysis without modifying model weights. A teacher model diagnoses a student's failures on financial tasks, clusters them by subfield and error type to identify the root causes of *domain-knowledge gaps*,

[1] Code and skill libraries are available at https://github.com/SallyTan13/ASDA-skill.

[2] Extended-thinking models (GPT-o1, DeepSeek-R1, Qwen-QwQ-32B) score 67–76% on FAMMA, and PoT-augmented variants reach 78–86%. We exclude these as thinking budget introduces a confound orthogonal to domain adaptation; our evaluation targets standard non-reasoning models under fixed inference budgets.

and synthesizes skill files containing domain-specific reasoning procedures and code templates, which a selector injects at inference time. Evaluated on FAMMA, ASDA achieves up to **+17.33 pp on arithmetic** and **+5.95 pp on non-arithmetic reasoning** after iterative refinement, significantly outperforming all training-free baselines. The resulting skill library is not a better prompt, but a new representational layer between the model and its deployment context that can be version-controlled, audited, and regenerated for any successor model.

1.1 Contributions

(1) ASDA framework. We introduce the first system to automatically generate executable agent skills for domain-specific reasoning using only black-box LLM access—no weight updates or gradient computation—substantially outperforming all training-free baselines on FAMMA.
(2) Self-sufficient adaptation from questions and answers alone. A self-teaching ablation shows that ASDA can improve a model using only the questions and ground-truth answers in the training set—no superior teacher model required—achieving +6.33 pp (73% of the full gain). This means any organization with a labeled domain dataset can run ASDA on their deployed model directly, without access to a stronger or more expensive model, making the framework practical for real-world enterprise deployment.

2 Related Work

2.1 Financial LLM Adaptation

Domain-specific fine-tuning has been the dominant approach to adapting LLMs for financial tasks. BloombergGPT [15] demonstrated the potential of finance-specific pretraining but required approximately 1.3 million GPU hours on a proprietary 363B-token corpus—resources beyond most organizations. FinGPT [18] offered a more accessible alternative through LoRA-based fine-tuning on open financial data. More recently, Xue et al. [17] explored distillation from DeepSeek-R1 to smaller models for financial reasoning. Despite these advances, all fine-tuning approaches share a fundamental limitation: they produce model-locked expertise that requires re-training when the base model is updated or replaced, and many are incompatible with black-box API access to commercial LLMs.

2.2 Training-Free Adaptation

Automated prompt optimization offers a weight-free alternative. Prior work formalizes this as automatic differentiation over text [19] or optimizable module compositions [9]. GEPA [2] achieves state-of-the-art results on several benchmarks through reflective prompt evolution. ACE [1] takes a test-time knowledge accumulation approach, building contextual expertise during inference. While these methods avoid fine-tuning costs, their output is a flat text string—a monolithic instruction block that cannot represent the modular, multi-step procedural knowledge required for complex domain reasoning, as our FAMMA experiments confirm (Section 5).

A complementary line of work treats failure analysis as the primary learning signal. LEMMA [4] synthesizes error-type-grounded training data for mathematical reasoning, consistently outperforming correction-agnostic data augmentation baselines, establishing that failure-driven analysis yields richer adaptation signal—a principle ASDA extends to training-free, executable skill generation. At the other end of the adaptation spectrum, test-time training (TTT) methods temporarily update model weights at inference, requiring gradient access incompatible with black-box API deployments; ASDA occupies the gap between these approaches.

2.3 Skill-Based Agent Architectures

Recent agent work has established skills as reusable, first-class knowledge artifacts. Voyager [14] demonstrated that composable executable skills—stored as JavaScript code—enable continual capability growth in open-ended Minecraft environments without parameter updates. The Agent Skills open standard [5] formalized portable Markdown skill files with routing metadata, progressive disclosure, and embedded code templates; ASDA generates skill artifacts compatible with this standard.

Recent benchmarking work quantifies when skills help or hurt. SkillsBench [10] finds that curated skills raise average pass rate by +16.2 *pp* across 11 domains, while self-generated skills provide no benefit on average—models cannot reliably author the procedural knowledge they benefit from consuming. SWE-Skills-Bench [8] reports that 39 of 49 public SWE skills produce no measurable improvement (average gain: +1.2%), with three actively degrading performance due to context-mismatched guidance. Together, these results establish that skill–task alignment is a precondition for benefit—the gap ASDA's error-driven distillation addresses by generating skills calibrated to a specific model's failure distribution on a target domain.

Several concurrent systems also explore automated skill generation from failure signals, though they differ from ASDA in inputs and setting. EvoSkill [3] iteratively diagnoses execution failures in coding agents—using ground-truth-supervised feedback in an online, interactive loop—to propose and validate new skill folders; ASDA targets static domain QA with no execution loop, relying solely on question–answer pairs, and adds a dual-phase iterative refinement that explicitly balances coverage gains against regression risk. Trace2Skill [13] runs an agent across many tasks, collects the resulting execution logs, and analyzes

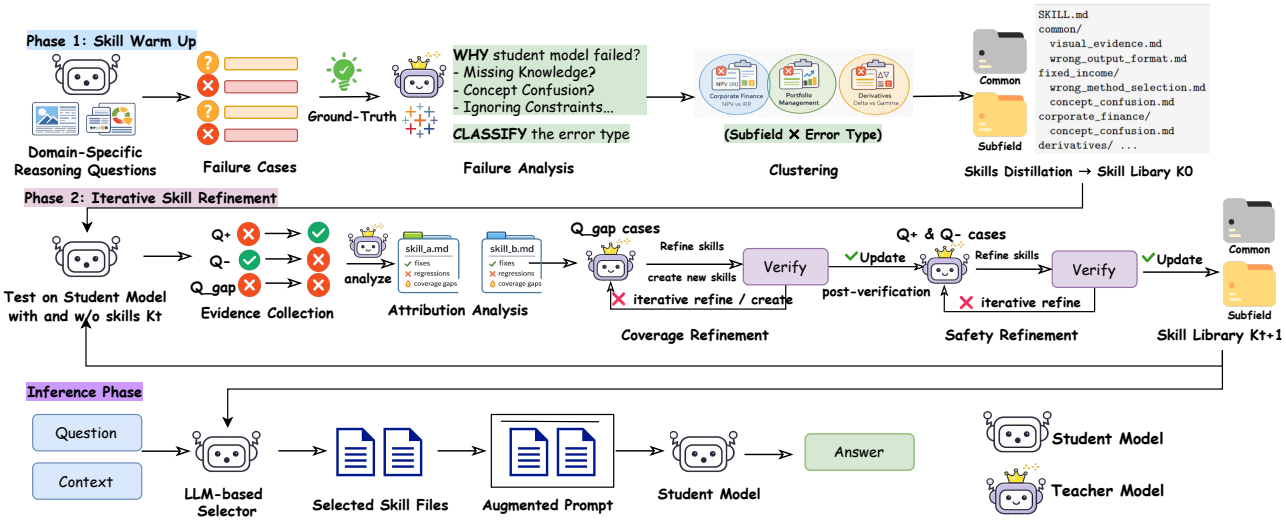

Figure 1. Overview of the ASDA framework. **Phase 1 (warm-up):** the teacher model analyzes student failures, producing structured annotations that are clustered by subfield and error type to synthesize an initial skill library $\mathcal{K}_0$. **Phase 2 (iterative refinement):** the library is refined through two sequential phases, coverage refinement (resolving uncovered failures in Q^{gap}) followed by safety refinement (suppressing regressions in Q^{-}), with every skill update gated by a correctness threshold. **Inference:** a selector reads `SKILL.md` and injects the relevant skill files into the student's prompt.

them in parallel to identify recurring patterns that are consolidated into skills, reporting strong cross-model and out-of-distribution transfer; ASDA requires only labeled Q&A pairs, making it compatible with the labeled-dataset setting common in regulated domains where full trajectory logs are unavailable. SkillRL [16] co-evolves skills with a reinforcement learning policy, still requiring SFT and weight updates—whereas ASDA is entirely training-free and compatible with API-only models.

Across these threads, ASDA is, to our knowledge, the first system to automatically generate training-free, executable skills for domain-specific reasoning from error analysis on static Q&A pairs alone, without execution traces, gradient access, or a stronger teacher model.

3 Method: ASDA Framework

ASDA operates through a teacher–student architecture comprising two phases: (1) a *warm-up* pipeline that establishes an initial skill library from systematic error analysis, and (2) an *iterative refinement* loop that refines skills through repeated evidence collection and validation. Figure 1 provides an overview.

Before describing each phase, we introduce three terms used throughout. An **error type** is one of ten categories from a predefined taxonomy.[3] A **pattern** is a single, named failure scenario within a skill file, one specific knowledge gap or recurring mistake the model makes. A **skill file** groups all patterns that share the same financial subfield and error type; for example, `fixed_income/wrong_method_selection.md` collects all wrong-method failures observed in the fixed income subfield.

3.1 Skills Warm-Up

3.1.1 Failure Analysis and Structured Annotation. The warm-up stage constructs an initial skill library $\mathcal{K}_0$ from the student model's failures on the training set. For each question the student answers incorrectly, the teacher model receives the question, the student's incorrect answer and reasoning trace, and the ground-truth answer. The teacher is then prompted to perform failure analysis and output a structured annotation in the following format:

```
{
  "subfield":    "fixed_income",
  "error_type":  "wrong method selection",
  "root_cause":  "Lacks knowledge that forward rates
    must be composed sequentially as discount
    factors, not applied independently per period"
}
```

The `error_type` field is constrained to the taxonomy defined above. The `root_cause` field captures the underlying knowledge gap rather than a surface description of what was

[3]The ten error types are: visual evidence, wrong method selection, concept confusion, missed multi-step computation, unit/currency mistakes, missed constraints, wrong targets, wrong output format, code execution errors (PoT-specific), and other. The taxonomy was generated by prompting an LLM to categorize recurring failure patterns from the student's errors on the training set, then refined through author review.

computed incorrectly, ensuring the diagnosis is actionable for skill synthesis.

3.1.2 Skill Library Organization.

The annotated failures are clustered by their `(subfield, error_type)` pair. Each cluster becomes one skill file. The library is organized as a two-level hierarchy:

```
SKILL.md                          % navigation + routing table
common/
  visual_evidence.md              % cross-subfield patterns
  wrong_output_format.md
fixed_income/
  wrong_method_selection.md       % one file per subfield x error type
  concept_confusion.md
corporate_finance/
  concept_confusion.md
derivatives/
...
```

Within each skill file, the teacher synthesizes one *pattern* per distinct failure scenario identified in the cluster. Each pattern contains: a concise description of the addressed knowledge gap, explicit "when to use" conditions, step-by-step reasoning procedures, and worked examples or code templates. Figure 2 shows one pattern from `fixed_income/wrong_method_selection.md`; the full file contains five additional patterns covering other recurring failure scenarios in that subfield.

The library also includes a top-level `SKILL.md` navigation file that summarizes the scope of each skill and provides a structured mapping from subfield keywords and failed financial patterns to skill file paths. This navigation file allows the downstream selector to identify relevant skills without parsing every skill file in full.

3.1.3 Skill Selection and Injection.

At inference time, an LLM-based selector reads the question text and the `SKILL.md` mapping table to identify the relevant financial subfield and match the question's characteristics against listed patterns. The selector may choose multiple skill files for a single question, since complex financial reasoning often requires combining guidance from several patterns (e.g., a fixed income pricing question may need both `wrong_method_selection.md` and `common/missed_constraints.md`). Selected skill files are injected into the student's prompt as domain knowledge, guiding it to follow the appropriate reasoning procedure for the question.

3.2 Dual-Phase Iterative Skill Refinement

The warm-up stage captures the most systematic failure patterns but leaves two residual problems: *coverage gaps*, where some failures remain because existing skills do not yet address the required knowledge; and *regressions*, where skill injection causes previously correct answers to become incorrect because a skill overfits a narrow failure pattern and misleads the model on related but distinct questions. We address these through iterative refinement, alternating between a *coverage phase* that expands skill coverage and a

Figure 2. One pattern from an ASDA skill file produced during warm-up from Haiku 3.5 failure analysis by Sonnet 4.5. The full file contains five additional patterns.

safety phase that suppresses regressions. Across both phases, every candidate skill update passes through a *verification gate*: the updated skill is tested by having the student resolve the target questions with the new skill injected, and is committed to the library only if the resulting accuracy meets a predefined threshold τ. If it fails, the teacher regenerates a revised proposal, repeating up to $N_{\max}$ attempts before falling back to the previous skill version. The full procedure is summarized in Algorithm 1.

3.2.1 Evidence Collection and Attribution.

At the start of each refinement iteration t, every training question is evaluated under two conditions: once with the current skill library $\mathcal{K}_t$ injected, and once without any skills. By comparing results, we partition the training set into three disjoint groups: Q_t^+ (correct with skills), Q_t^- (incorrect with skills but correct without, i.e., regressions introduced by the current library), and Q_t^{gap} (incorrect under both conditions, i.e., coverage failures that the library has not yet resolved).

Since multiple skill files may be loaded for each question, a naïve per-question outcome cannot be attributed to a specific file. An *attribution* step follows: for each question in Q_t^+, Q_t^-, and Q_t^{gap}, the teacher examines the student's reasoning trace alongside the loaded files and identifies the single file most responsible for the outcome. Questions are then re-grouped by their attributed file, producing per-file evidence sets that

Algorithm 1 Dual-Phase Iterative Skill Refinement

Require: Skill library $\mathcal{K}_0$, training set $\mathcal{D}$, student M_s, teacher M_t, max iterations T
Ensure: Refined skill library $\mathcal{K}^*$

```
for t = 1 to T do
    // Evidence Collection
    Q+, Q−, Q^gap ← EvalWithWithout(D, K_t, M_s)
    AttributeToFiles(Q+, Q−, Q^gap, M_t)

    // Coverage Phase
    for each file f with attributed Q_f^gap ≠ ∅ do
        f' ← M_t.ProposeExpansion(f, Q_f^gap)
        if Verify(f', Q_f^gap, M_s) ≥ τ_cov then
            K_t[f] ← f'
        end if
    end for
    Q̃+, Q̃− ← PostCoverageVerify(K_t, Q+, Q^gap, M_s)

    // Safety Phase
    for each file f with attributed Q̃_f^− ≠ ∅ do
        f' ← M_t.ProposeRepair(f, Q̃_f^+, Q̃_f^−)
        if Verify(f', Q̃_f^+, M_s) ≥ τ_safe then
            K_t[f] ← f'
        end if
    end for
    K_{t+1} ← K_t
end for
return K_T
```

isolate each file's individual contribution to fixes, regressions, and remaining coverage gaps.

3.2.2 Coverage Phase. For each skill file, the teacher examines the Q_t^{gap} cases attributed to that file and diagnoses why coverage fails. Common causes include missing procedures for an edge case, trigger conditions that are too narrow to fire on relevant questions, or the absence of a worked example for a pattern the file does not yet contain. Based on this diagnosis, the teacher proposes an update, either by refining an existing pattern or by adding a new one. Each proposal is submitted to the verification gate: the student re-solves the attributed Q_t^{gap} cases with the candidate update injected, and the update is accepted only if the recovery rate exceeds τ_{cov}.

After all per-file coverage updates are committed, a post-coverage verification pass re-evaluates the entire affected set. Cases in Q_t^{gap} that are now solved are promoted into an updated positive set $\tilde{Q}_t^+$. Cases in Q_t^+ that regress under the modified skills are merged with the existing Q_t^- to form an updated regression set $\tilde{Q}_t^-$. This updated partition forms the input to the safety phase.

3.2.3 Safety Phase. The safety phase resolves regressions in $\tilde{Q}_t^-$, including both pre-existing ones and those newly introduced by the coverage phase, without disrupting correct behavior on $\tilde{Q}_t^+$. For each skill file, the teacher receives both sets simultaneously as contrastive evidence: the $\tilde{Q}_t^+$ cases (annotated with what the current skill gets right) serve as preservation constraints, and the $\tilde{Q}_t^-$ cases (annotated with what goes wrong) serve as repair targets. The teacher proposes a revised skill that removes or narrows the guidance responsible for the regressions while preserving the reasoning steps that produce correct answers on the positive set. Each proposal again passes through the verification gate with threshold τ_{safe}, which requires that accuracy on the positive cases not degrade too much while recovering as many negative cases as possible.

After both phases complete, the updated library $\mathcal{K}_{t+1}$ becomes the input for the next iteration.

4 Experimental Setup

4.1 Benchmark: FAMMA

FAMMA-Basic [17] provides the scale, arithmetic/non-arithmetic decomposition, and self-contained textual context our pipeline requires.[4] It comprises 1,945 questions sourced from university textbooks and professional finance exams, spanning eight financial subdomains (e.g., corporate finance, derivatives, portfolio management) across three difficulty levels, with an explicit decomposition between arithmetic and non-arithmetic questions that enables separate evaluation of procedural and conceptual skill effectiveness. We use the FAMMA-Basic-Txt release, which provides OCR-extracted textual context for each question, ensuring that evaluation targets reasoning ability rather than retrieval.[5]

4.2 Data Filtering and Split

We restrict the corpus to the 1,378 English-language questions to control for language variation, ensuring that observed performance differences reflect reasoning capability alone. Since FAMMA provides no official train–test split, we construct our own: separating the English corpus into arithmetic and non-arithmetic subsets and applying stratified 60/40 splits based on difficulty level (easy, medium, hard) and question type (multiple-choice vs. open-ended). This produces 448 training and 300 test questions for arithmetic, and 378 training and 252 test questions for non-arithmetic. All reported results are on these held-out test sets and are

[4] We also considered FinMR [6], FinanceMath [20], FinanceQA [12], and FinMME [11], but these are either too small for reliable train–test splits, withhold ground-truth answers for their primary test sets, yield baseline results that differ substantially from published figures, or focus on visual and retrieval-based reasoning rather than procedural financial reasoning.
[5] FAMMA also includes a LivePro subset of 103 expert-curated questions, but only 35 are English-language and most are open-ended, making it too small for our pipeline.

Table 1. Evaluation protocol modifications.

	Original FAMMA	Ours
Question handling	Grouped sub-questions	Independent
MC evaluation	LLM judge	Rule-based exact match
Open-ended evaluation	LLM judge (GPT-4o)	LLM judge (Qwen-Max)
Arithmetic execution	PoT	PoT + MC mapping step

therefore not directly comparable to those reported by Xue et al. [17].

4.3 Evaluation Protocol

ASDA's distillation pipeline operates on individual question–answer pairs, so we evaluate each question independently rather than in grouped LLM calls as in the original FAMMA protocol. FAMMA stores shared context only in the first sub-question of each group, so we propagate this context to all sub-questions to preserve information completeness.[6] Arithmetic questions use Program-of-Thought (PoT) code execution, with an additional selection step that maps numeric outputs to the closest multiple-choice option where applicable. For evaluation, we adopt a hybrid approach: rule-based exact matching for multiple-choice questions and LLM-based judging for open-ended questions, replacing the original protocol's use of LLM judging for all question types. Table 1 summarizes these modifications.

Validation confirms these modifications do not inflate gains: swapping Qwen-Max for GPT-4o yields 99.3% agreement across 1,104 judgments (delta: 0.00 pp arithmetic, −0.39 pp non-arithmetic); adopting the full FAMMA LLM-judge strategy confirms the same shift.

4.4 Baselines

We compare ASDA against two leading training-free adaptation methods that operate under the same deployment constraint—black-box API access without weight modifications: GEPA [2] and ACE [1]. Both methods optimize a single monolithic text applied uniformly to all questions. The baseline condition uses the student model with a standard task prompt and no injected skills or optimized prompts. Neither GEPA nor ACE has previously been evaluated on FAMMA; the results reported here are, to our knowledge, the first published evaluations of both methods on this benchmark.

To ensure a fair comparison, we adapt training conditions to each method's architectural requirements.[7]

[6]Additional preprocessing: we enforce expression-based PoT outputs rather than `print()` statements to prevent execution failures.

[7]GEPA follows its original 3-way split protocol, training on 50% of our training pool (222 arithmetic, 188 non-arithmetic) and selecting the best prompt on the remaining 50%. ACE and ASDA use the full training pool (448 arithmetic, 378 non-arithmetic).

Table 2. ASDA pipeline cost (Haiku 3.5 student, Sonnet 4.5 teacher). Teacher priced at Sonnet 4.5 rates ($3 / $15 per M tokens in/out); student at Haiku 3.5 rates ($0.80 / $4 per M tokens in/out). Qwen-based judging and MC selection tokens are included in student-phase totals and conservatively priced at Haiku 3.5 rates; actual DashScope costs are approximately 10× lower.

		Teacher (S4.5)		Student (H3.5)		
Config	Stage	In	Out	In	Out	Cost
Arith	Warm-up	545K	121K	8.07M	331K	$11
	Refine (E1–E2)	16.1M	2.1M	65.4M	741K	$135
Non-Arith	Warm-up	485K	94K	5.80M	194K	$8
	Refine (E1–E2)	9.2M	1.2M	51.2M	598K	$89

4.5 Implementation Details

Two components are fixed across all experimental conditions: Qwen-Max serves as the LLM evaluation judge for open-ended questions, and Qwen-Turbo performs the MC selection step that maps numeric PoT outputs to answer choices. All inference is run at temperature 0 for reproducibility.[8]

Table 2 reports token usage and estimated cost for both the warm-up pipeline and iterative refinement through two epochs (the paper's best operating point). Warm-up cost is dominated by skill-augmented evaluation; refinement is substantially more expensive due to repeated train-and-test evaluation and teacher-driven analysis at each epoch.

5 Results and Analysis

5.1 Main Results

We evaluate ASDA across two student models on both arithmetic and non-arithmetic tasks. For Haiku 3.5, we additionally include GEPA and ACE as training-free reference points; both achieve only marginal gains on FAMMA, which suggests the structural limitations of flat-text optimization.

ASDA consistently improves performance for Claude-family student models. On arithmetic, Haiku 3.5 gains +8.67 pp at warm-up and +17.33 pp after two refinement epochs. Haiku 4.5, despite its stronger 64.67% baseline, achieves +5.99 pp—showing that ASDA adds value even for more capable models. Non-arithmetic gains are smaller but consistent for Haiku 3.5 (+2.78 pp warm-up, +5.95 pp at E2); Haiku 4.5 sees modest non-arithmetic improvement only after refinement (+1.60 pp at E2).

Effect of iterative refinement. The warm-up stage targets the most frequent failure patterns and delivers the first large wave of gains. Iterative refinement then addresses residual failures that remain after initial skill injection. For arithmetic, accuracy improves from 49.67% at warm-up to 54.67%

[8]All models are accessed via the Anthropic API, with two exceptions: Haiku 3.5 via OpenRouter (no longer available on the Anthropic API directly) and Qwen models via DashScope.

Table 3. ASDA results across student models on FAMMA. For Haiku 3.5, GEPA and ACE are shown as reference baselines. All experiments use Sonnet 4.5 as the teacher model. WU = Warm-Up; E2 = best refinement epoch. Δ denotes absolute improvement in percentage points over each model's own baseline.

Student	Method	Arithmetic		Non-Arithmetic	
		Acc. (%)	Δ	Acc. (%)	Δ
Haiku 3.5	Baseline	41.00	—	49.21	—
	GEPA [2]	42.33	+1.33	50.79	+1.58
	ACE [1]	44.30	+3.30	49.60	+0.39
	ASDA WU **(ours)**	49.67	+8.67	51.98	+2.78
	ASDA E2 **(ours)**	**58.33**	**+17.33**	**55.16**	**+5.95**
Haiku 4.5	Baseline	64.67	—	57.14	—
	ASDA WU **(ours)**	69.67	+5.00	56.35	−0.79
	ASDA E2 **(ours)**	**70.66**	**+5.99**	**58.74**	**+1.60**

at epoch 1 and reaches 58.33% at epoch 2. A smaller but consistent trend holds for non-arithmetic, where epoch 2 peaks at 55.16%. Both domains regress at epoch 3 (arithmetic: 54.33%; non-arithmetic: 51.98%), indicating overfitting to residual training-set patterns after two refinement passes. In practice, two refinement epochs represent the optimal operating point.

Gains by question type. Skills consistently produce larger gains on multiple-choice questions than on open-ended ones. For Haiku 3.5 arithmetic at warm-up, MC accuracy rises by +14.39 pp compared to +3.73 pp for open-ended; Haiku 4.5 shows the same pattern (MC +7.91 pp, open-ended +2.48 pp). A skill that narrows the solution procedure is most useful when the answer space is already constrained to a few options.

Regressions reveal the complementary risk. In sampled cases, the most common pattern is skill-induced overreasoning: the model elaborates beyond what the question requires and revises an already-correct judgment. Because the selector loads all skill files for a subfield as a bundle, every question receives guidance regardless of whether it needs it—questions the baseline already handles correctly can be destabilized by unnecessary procedural elaboration.

5.2 Qualitative Analysis

To illustrate how skill artifacts operate at inference time, Figure 3 presents an arithmetic case from the Haiku 3.5 warm-up evaluation where the baseline fails and skill injection succeeds. The same skill file was credited with 7 additional fixes on related fixed income questions, illustrating the reusability of subfield-specific skill artifacts.

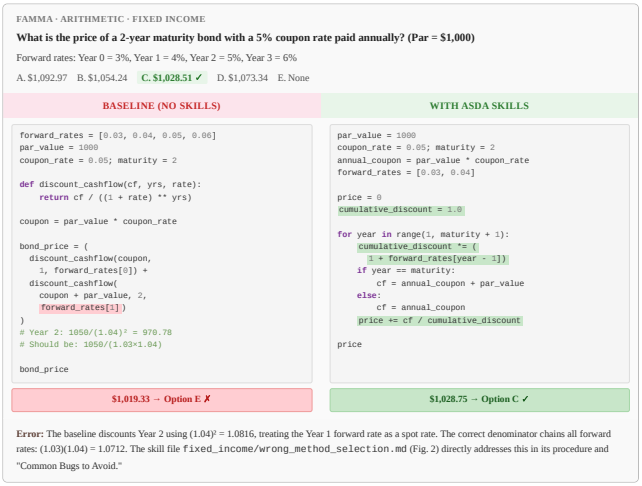

Figure 3. Baseline vs. skill-augmented output on a FAMMA fixed income question (Haiku 3.5, warm-up). The skill file in Figure 2 provides the domain-specific procedure that corrects the baseline error. The same skill file was credited with 7 additional fixes on related fixed income questions.

Table 4. Self-teaching ablation. Each model serves as both student and teacher; no superior model is involved. Full ASDA (Sonnet 4.5 teacher) results are shown for reference.

Student	Domain	Baseline	With Skills	Δ
Full ASDA (Sonnet 4.5 teacher), for reference				
Haiku 3.5	Arithmetic	41.00	49.67	+8.67
Haiku 3.5	Non-Arith	49.21	51.98	+2.78
Self-Teaching (student = teacher)				
Haiku 3.5	Arithmetic	41.00	**47.33**	**+6.33**
Haiku 3.5	Non-Arith	49.21	50.79	+1.58

5.3 Self-Teaching Ablation

ASDA's gains could come from two sources: knowledge contributed by a superior teacher model, or knowledge drawn from the training data itself. To separate these, we run a self-teaching configuration where the student model acts as its own teacher—it analyzes its own failures, builds the skill library, and then uses those skills at inference. No stronger model is involved.

For Haiku 3.5 arithmetic, self-teaching achieves +6.33 pp—73% of the +8.67 pp gain from using a Sonnet 4.5 teacher. This shows that most of the benefit comes from the training data, not from the teacher's superior knowledge. It is worth noting that the training questions provide only the question text and the correct answer—not worked solutions or step-by-step reasoning. Even so, seeing where it consistently goes wrong is enough for the model to identify its recurring failure patterns and synthesize skills to address

Table 5. Skill portability: Haiku 4.5 arithmetic (300 eval questions). Own skills are generated from Haiku 4.5's failures; cross-transfer uses skills generated from Haiku 3.5's failures. OE = open-ended questions.

Skills Source	Base	w/ Skills	Δ	MC Δ	OE Δ
Own skills (H4.5)	64.67	**69.67**	**+5.00**	+7.91	+2.48
H3.5 skills (cross-transfer)	64.67	62.33	−2.33	+2.16	−6.21

them. In other words, the model already possesses much of the relevant domain knowledge; the distillation process gives it the structure to apply that knowledge reliably. The remaining 2.34 pp gap reflects the teacher's contribution—a stronger model produces sharper failure diagnoses and more precise skill formulations. The same pattern holds for non-arithmetic (+1.58 pp self-teaching vs. +2.78 pp with a Sonnet teacher).

5.4 Cross-Transfer Experiment

The self-teaching results raise a natural follow-on question: once skills are distilled for one model, can they benefit another? We apply skills generated from Haiku 3.5's failures to Haiku 4.5 (a stronger model in the same family) and compare against skills derived from Haiku 4.5's own failures.

Cross-model transfer produces a net regression of −2.33 pp, driven by a sharp open-ended decline (−6.21 pp) that overwhelms modest MC gains (+2.16 pp). In contrast, Haiku 4.5 with its own ASDA skills gains a consistent +5.00 pp across both question types.

The practical implication is direct: skills should be generated for each deployed model independently. Reusing skills across model generations is unlikely to yield positive returns and can actively harm the stronger model. As shown in Table 2, student-specific warm-up distillation costs under $12, with two refinement epochs adding $89–135, making per-model regeneration operationally feasible at deployment scale.

6 Discussion and Conclusion

Discussion. Two experimental findings illuminate the underlying mechanism of ASDA. The self-teaching result—where a model acting as its own teacher recovers 73% of the full arithmetic gain—shows that the primary source of improvement is not the teacher's superior knowledge but the structure imposed by the distillation process itself. Systematically enumerating failure patterns across a training set forces the model to externalize domain knowledge it already implicitly holds but cannot reliably apply during single-pass inference. The cross-transfer failure reinforces this view from the opposite direction: skills are not generic domain knowledge that transfers across models, but artifacts of a specific model's failure distribution. Applying one model's skills to a stronger model actively constrains it by the weaker model's blind spots.

Together, these results characterize skills as *model-specific failure remedies*—a distinction with direct consequences for how skill libraries should be managed in practice. For organizations in regulated industries— financial services, legal, and healthcare—that rely on commercial LLMs via black-box API for knowledge-intensive workflows, ASDA offers a concrete operational path: run the distillation pipeline once on a labeled in-domain dataset, version-control the resulting skill files alongside application code, and regenerate them when the base model is upgraded. The skill library then functions as an auditable, inspectable knowledge layer that domain experts can review, compliance teams can certify, and engineering teams can update without touching model weights.

ASDA's gains are largest when failure patterns are clustered and the task has well-defined procedural structure, as with arithmetic reasoning via Program-of-Thought. When errors are more dispersed—as in non-arithmetic tasks—the signal available for distillation is weaker and regression risk increases. This boundary condition suggests that diagnostic quality (how cleanly errors cluster by type and subfield) predicts adaptation success alongside raw model capability.

Implications for skill design. Our results offer two practical lessons for how skills should be generated and deployed. First, our self-teaching ablation appears to contradict SkillsBench's [10] finding that models cannot generate useful skills on their own—but the difference is in *how* the skills are generated. SkillsBench prompts the model to write skills from scratch using only what it already knows, whereas ASDA's self-teaching configuration generates skills by systematically analyzing the model's own failures on a labeled dataset. Asking a model to write skills without grounding in specific failure evidence yields little benefit; structuring the generation around systematic error analysis, as ASDA does, is what drives the gains. Second, the cross-transfer regression (−2.33 *pp*) shows that skills derived from one model's failures do not automatically benefit a different model, and can introduce regressions. In practice, regenerating skills for each deployed model is the safer default, though whether targeted cross-transfer strategies can overcome this remains an open direction.

Limitations. Results are confined to FAMMA and the Claude model family. While the self-teaching ablation demonstrates that ASDA does not require a stronger teacher, we have not yet validated whether the framework generalizes to non-Claude student models, other domains, or different error taxonomies. FAMMA's OCR-extracted text introduces an additional confound: skills distilled from questions containing OCR artifacts may encode corpus-specific correction patterns that produce false negatives when applied to

correctly formatted test questions, potentially inflating regression counts relative to cleaner datasets. Finally, when the selector does match a subfield, it loads all associated skill files as a bundle rather than selecting only the specific patterns relevant to the question. As noted in our results, this coarse routing can introduce unnecessary procedural guidance that destabilizes questions the baseline already handles correctly—a limitation that finer-grained, pattern-level selection could address.

Future Work. *Cross-domain transfer.* Healthcare and legal reasoning are strong candidates for future evaluation—both are procedural, knowledge-intensive, and carry auditability requirements that align with the skill library's inspectable format.

Skill compression. The current pipeline generates 10–30 skill files per configuration. A pruning step that measures per-skill regression rates and merges narrowly scoped files could reduce regressions while sharpening routing precision.

Cross-model generalization. Our results are limited to Claude-family models in both the student and teacher roles. Evaluating ASDA with student or teacher models from other families (e.g., Qwen, DeepSeek) would clarify whether the framework's gains depend on model-family-specific instruction-following behavior or generalize more broadly.

Conclusion. The central finding of this work is that failure-driven distillation can externalize latent domain knowledge into an explicit, inspectable form that single-pass inference cannot access without modifying model weights. Our ablations further reveal that the resulting skills function as *model-specific failure remedies*: artifacts of a particular model's failure distribution that cannot be transferred across model generations, but can be cheaply regenerated for any deployed model given only a labeled domain dataset—at a one-time warm-up cost of under $12, with optional iterative refinement through two epochs costing $89–135 additional (Table 2). The skill library is not a better prompt: it is a new representational layer between the raw model and its deployment context, one that can be version-controlled, audited, and regenerated for successor models. Whether this layer generalizes to other knowledge-intensive domains, and how far self-teaching can substitute for stronger supervision, remain the central open questions.

Acknowledgments

The authors used generative AI tools to assist with manuscript editing and experimental pipeline development. All scientific content, including the hypotheses, experimental design, and conclusions, is entirely the work of the human authors.

The authors have no competing interests to declare that are relevant to the content of this article.

References

[1] Zhang, Q., Hu, C., Upasani, S., Ma, B., Hong, F., Kamanuru, V., Rainton, J., Wu, C., Ji, M., Li, H., Thakker, U., Zou, J., Olukotun, K. Agentic Context Engineering: Evolving Contexts for Self-Improving Language Models. arXiv preprint arXiv:2510.04618 (2025).

[2] Agrawal, L.A., Tan, S., Soylu, D., Ziems, N., Khare, R., Opsahl-Ong, K., Singhvi, A., Shandilya, H., Ryan, M.J., Jiang, M., Potts, C., Sen, K., Dimakis, A.G., Stoica, I., Klein, D., Zaharia, M., Khattab, O. GEPA: Reflective prompt evolution can outperform reinforcement learning. arXiv preprint arXiv:2507.19457 (2025).

[3] Alzubi, S., Provenzano, N., Bingham, J., Chen, W., Vu, T. EvoSkill: Automated Skill Discovery for Multi-Agent Systems. arXiv preprint arXiv:2603.02766 (2026).

[4] Pan, Z., et al. LEMMA: Learning from errors for mathematical advancement in LLMs. In *Findings of ACL 2025*. arXiv preprint arXiv:2503.17439 (2025).

[5] Anthropic. Agent Skills: An open standard for executable agent knowledge. https://agentskills.io (2025).

[6] Deng, S., Peng, H., Xu, J., Mao, R., Giurcăneanu, C.D., Liu, J. FinMR: A Knowledge-Intensive Multimodal Benchmark for Advanced Financial Reasoning. arXiv preprint arXiv:2510.07852 (2025).

[7] Xie, Q., et al. FinBen: A holistic financial benchmark for large language models. In *NeurIPS 2024*. arXiv preprint arXiv:2402.12659 (2024).

[8] Han, T., Zhang, Y., Song, W., Fang, C., Chen, Z., Sun, Y., Hu, L. SWE-Skills-Bench: Do Agent Skills Actually Help in Real-World Software Engineering? arXiv preprint arXiv:2603.15401 (2026).

[9] Khattab, O., Potts, C., Zaharia, M. DSPy: Compiling declarative language model calls into self-improving pipelines. arXiv preprint arXiv:2310.03714 (2023).

[10] Li, X., et al. SkillsBench: Benchmarking How Well Agent Skills Work Across Diverse Tasks. arXiv preprint arXiv:2602.12670 (2026).

[11] Luo, J., et al. FinMME: Benchmark Dataset for Financial Multi-Modal Reasoning Evaluation. arXiv preprint arXiv:2505.24714 (2025).

[12] Mateega, S., Georgescu, C., Tang, D. FinanceQA: A Benchmark for Evaluating Financial Analysis Capabilities of Large Language Models. arXiv preprint arXiv:2501.18062 (2025).

[13] Ni, J., Liu, Y., Liu, X., Sun, Y., Zhou, M., Cheng, P., Wang, D., Zhao, E., Jiang, X., Jiang, G. Trace2Skill: Distill Trajectory-Local Lessons into Transferable Agent Skills. arXiv preprint arXiv:2603.25158 (2026).

[14] Wang, G., Xie, Y., Jiang, Y., Mandlekar, A., Xiao, C., Zhu, Y., Fan, L., Anandkumar, A. Voyager: An open-ended embodied agent with large language models. arXiv preprint arXiv:2305.16291 (2023).

[15] Wu, S., Irsoy, O., Lu, S., Daber, V., Dredze, M., Gehrmann, S., Gupta, P., Ishrat, S., Jha, A., Johnston, S., et al. BloombergGPT: A large language model for finance. arXiv preprint arXiv:2303.17564 (2023).

[16] Xia, P., et al. SkillRL: Evolving agents via recursive skill-augmented reinforcement learning. arXiv preprint arXiv:2602.08234 (2026).

[17] Xue, Z., et al. FAMMA: A benchmark for financial domain multilingual multimodal question answering. arXiv preprint arXiv:2410.04526 (2024).

[18] Yang, H., Liu, X., Wang, C. FinGPT: Open-source financial large language models. arXiv preprint arXiv:2306.06031 (2023).

[19] Yüksekgonül, E., Bianchi, F., Boen, J., Liu, T., Zou, J. TextGrad: Automatic "differentiation" via text. arXiv preprint arXiv:2406.07496 (2024).

[20] Zhao, Y., Liu, H., Long, Y., Zhang, R., Zhao, C., Cohan, A. FinanceMath: Knowledge-Intensive Math Reasoning in Finance Domains. arXiv preprint arXiv:2311.09797 (2024).