ATMOSKILLRSI: SKILL FORMATION AND RECURSIVE REFINEMENT IN ATMOSPHERIC WORKFLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Recursive self-improvement (RSI) raises a central question for scientific agents: how can task experience be consolidated into persistent updates that participate in subsequent adaptation? In ATMOSKILLRSI, we study this question in atmospheric workflow construction using Earth2Studio, where agents build executable workflows evaluated with objective scientific metrics. Agents convert task experience and evaluation feedback into persistent Skills containing reusable procedural guidance and supporting code; these Skills are retained across tasks, guide subsequent workflow exploration, and can themselves be revised using later experience. Across experiments, we hold the base coding model, execution environment, task interface, and evaluator fixed, while examining variations in experience organization, training-case coverage, synthesis evidence, and repeated Skill reuse. We compare Sequential and Task-Parallel experience organization, evaluate Skill effectiveness across atmospheric task families and learning conditions, and track repeated Skill refinement across successive stages. The available results show that both Sequential and Task-Parallel learning improve mean RPF performance over Base, that broader training coverage does not consistently yield larger gains, and that mean performance increases across successive refinement stages under a fixed learning rule. Because later stages also incorporate additional exploration and accumulated history, these stage-wise trends are descriptive rather than a causal estimate of prior-Skill feedback. Overall, ATMOSKILLRSI provides a controlled execution setting for studying persistent Skill formation and recursive refinement in scientific agents.

1 INTRODUCTION

Rapid advances in large language models (LLMs) and autonomous agents have brought recursive self-improvement (RSI) into focus as a research goal for scientific AI (Liu et al., 2026; Zheng et al., 2026; Duan et al., 2026). For scientific agents, RSI seeks to improve the procedures that govern successive cycles of proposing candidate solutions, running experiments, and evaluating outcomes, such that experience from earlier cycles changes how later exploration is conducted. This capability is particularly relevant in atmospheric science, where scientific workflows often involve composing models, data sources, diagnostics, and evaluation procedures into iterative experimental pipelines. Earth2Studio provides a composable environment for constructing such workflows (Earth2Studio Contributors, 2024), enabling controlled study of how agents learn from workflow execution. A central challenge is how experimental experience can be transformed into persistent, reusable procedures that guide subsequent exploration and can themselves be revised by later experience. Here, we study how exploration experience forms reusable Skills in atmospheric workflows, how learned Skills guide subsequent exploration, and how later experience revises those Skills. We ask *how scientific exploration experience is transformed into persistent, reusable Skills, how their effectiveness varies across scientific tasks and learning conditions, and whether reusing them in subsequent exploration supports cumulative refinement.*

A *Skill* packages reusable procedural guidance and supporting code for constructing scientific workflows. For an atmospheric agent, such guidance can encode when and how to perturb initial conditions, combine ensemble members, or apply calibration procedures appropriate to a task. Because these choices correspond to executable computations with constrained inputs and outputs, their utility can be evaluated through the scientific workflows they produce. A useful Skill can therefore

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

preserve procedural knowledge across related tasks, reducing the need to rediscover effective workflow constructions from scratch.

Learning such Skills is difficult because feedback evaluates individual workflows, whereas the resulting Skills must guide workflow construction beyond the cases from which they were learned. A high score may reflect a transferable procedure, a case-specific choice, or a favorable realization. Extracting reusable guidance therefore depends on how experience is organized, which cases it covers, and what evidence is retained for synthesis; procedures useful in one scientific workflow family may also be less effective in another. Recursive reuse further complicates this process: a learned Skill shapes subsequent exploration and thus the evidence available for its own revision, potentially consolidating useful knowledge or reinforcing narrow heuristics.

Recent work shows that agents can use interaction experience to revise persistent skills, policies, and executable structures that guide subsequent behavior (Qian et al., 2026; Zheng et al., 2026; Lee et al., 2026; Chen et al., 2026). However, how experience organization, synthesis evidence, and persistent Skill reuse affect scientific workflow learning remains insufficiently understood under a shared executable evaluation setting.

We introduce ATMOSKILLRSI, a controlled empirical study of these questions in atmospheric workflows. Selected tasks are formulated as constrained search over executable directed acyclic graphs (DAGs), allowing candidate scientific computations to be executed and scored under a common interface. We compare SEQUENTIAL and TASK-PARALLEL learning, examine Skill effectiveness across atmospheric task families, vary training-case coverage and evidence available during Skill synthesis, and then study repeated Skill reuse and revision across successive learning stages. We operationalize RSI at the Skill layer: persistent procedural state learned at one stage conditions subsequent exploration and is revised using experience produced by that conditioned exploration. The underlying learning strategy is held fixed while the persistent Skill state and retained experience evolve across stages, separating Skill-level refinement from modification of the update mechanism itself.

Across our experiments, both Sequential and Task-Parallel organization produce effective Skills on Regional Probabilistic Forecasting (RPF), while Skill effectiveness varies across atmospheric task families under Sequential learning. Increasing training-case coverage does not consistently improve workflow performance, whereas the trace-enabled Task-Parallel configuration performs better in the available synthesis comparison. Across successive Skill-refinement stages, mean workflow performance continues to improve, supporting the potential of persistent Skill reuse for cumulative refinement. Together, these observations show that accumulating more experience alone does not guarantee better Skills, while experience organization, synthesis evidence, task context, and persistent reuse provide complementary dimensions for understanding Skill formation and refinement.

Our contributions are threefold:

Contribution 1. We introduce a controlled atmospheric workflow setting for studying persistent Skill learning, representing candidate scientific computations as executable DAGs and reusable Skills as procedural state that persists across exploration stages.

Contribution 2. We empirically characterize single-stage Skill learning across experience organizations, atmospheric task families, training-case coverage, and synthesis evidence. We find that learned-Skill effectiveness varies across task families, that training coverage does not consistently improve Skill quality, and that trace-enabled synthesis performs better in the available comparison.

Contribution 3. We study recursive Skill refinement through repeated Skill-guided exploration and revision under a fixed learning rule, observing continued improvement in mean workflow performance across successive refinement stages.

2 RELATED WORK

Scientific Agents for Atmospheric Science. Beyond atmospheric science, scientific agents have combined language models with domain-specific tools and laboratory interfaces to plan and execute chemical research, demonstrating how language-based reasoning can be grounded in executable scientific procedures (Boiko et al., 2023; M. Bran et al., 2024). Recent work has extended LLM applications in atmospheric and climate science from static reasoning toward executable scientific agents

that support planning, tool use, code execution, iterative analysis, and experience reuse (Li et al., 2025a; Varambally et al., 2025; Li et al., 2025b; Wang et al., 2026a; Guo et al., 2025). These systems demonstrate growing capability for autonomous scientific workflow execution, but primarily emphasize end-to-end task performance, within-task refinement, or system-level knowledge accumulation. Less attention has been given to how exploration experience is organized into persistent reusable Skills and how those Skills shape subsequent learning. ATMOSKILLRSI studies these mechanisms through controlled, executable, and objectively evaluated atmospheric workflows.

Persistent Skill Learning and Recursive Self-Improvement. A growing literature studies how interaction experience can be consolidated into persistent agent state, including linguistic memory, executable skills, and reusable scientific procedures (Shinn et al., 2023; Wang et al., 2024a; Qian et al., 2026; Ren et al., 2026). Experience can be retained as cross-task insights in ExpeL, continually updated causal abstractions in CLIN, or reusable functions in TroVE, providing complementary representations for transferring knowledge across subsequent problems (Zhao et al., 2024; Majumder et al., 2024; Wang et al., 2024b). Recent methods further investigate how experience from multiple tasks or trajectories should be organized before synthesis, ranging from joint multi-task distillation to trajectory-level consolidation (Zhou et al., 2026; Ni et al., 2026). Complementary optimization frameworks improve language-model pipelines through metric-guided compilation, textual feedback propagation, and automated search over agent designs, as exemplified by DSPy, TextGrad, and Automated Design of Agentic Systems (Khattab et al., 2024; Yuksekgonul et al., 2025; Hu et al., 2025). Related work on recursive self-improvement closes the loop by allowing persistent agent state or improvement mechanisms to influence the experience used for subsequent updates (Zheng et al., 2026; Lee et al., 2026). At the code level, STOP and the Darwin Gödel Machine explore self-improvement by revising an improver program or the agent's own implementation, whereas our study examines persistent Skill refinement under a fixed learning rule (Zelikman et al., 2024; Zhang et al., 2025). ATMOSKILLRSI brings these directions together by studying Skill formation, multi-task experience organization, and recursive refinement within a shared executable atmospheric workflow environment with objective scientific evaluation.

3 ATMOSKILLRSI: PROBLEM FORMULATION AND FRAMEWORK

ATMOSKILLRSI studies how scientific exploration produces reusable Skills, how their effectiveness varies across tasks and learning conditions, and how those Skills participate in subsequent learning. Within a shared scientific environment, we compare alternative organizations and sources of learning experience, examine Skill effectiveness across atmospheric tasks, and then study recursive refinement when a learned Skill guides further exploration. We distinguish three objects in this process: an executable workflow candidate g, a persistent Skill S, and a learning rule U that governs exploration and Skill consolidation. Candidates vary within task exploration, Skills persist and evolve across learning stages, and U remains fixed within each experimental configuration. Figure 1 summarizes workflow exploration, Skill consolidation, recursive continuation, and assessment of frozen Skills.

3.1 PROBLEM FORMULATION

An atmospheric task τ specifies public inputs, registered scientific assets, an operator catalog, required outputs, and a scientific metric. Its admissible candidate space $\mathcal{G}_\tau$ consists of finite typed directed acyclic graphs (DAGs): nodes specify scientific operations and their parameters, while edges carry intermediate data. A workflow agent A constructs a candidate g using the task specification, any supplied Skill S, and the training information visible to its role.

For a candidate that executes successfully and satisfies the task requirements,

$$g \in \mathcal{G}_\tau, \qquad y = \mathcal{E}_\tau(g), \qquad m_\tau = F_\tau(y), \qquad q_\tau = d_\tau m_\tau, \tag{1}$$

where $\mathcal{E}_\tau$ is the task-specific Executor, y is the resulting workflow output, and F_τ is the scientific evaluator using task-specific private reference data. Here m_τ is the native metric and $d_\tau \in \{-1, +1\}$ maps its direction to a higher-is-better Judge score q_τ. For RPF and FIIF, $m_\tau = L_\tau$ is normalized CRPS; for WPEP, it is mean Brier loss. These three families use $d_\tau = -1$. For PSSC, $m_\tau = C_{\text{precip}}$ is the composite precipitation score and $d_\tau = +1$, so the Judge returns it directly. This mapping aligns score direction across tasks; it does not make their numerical scales comparable.

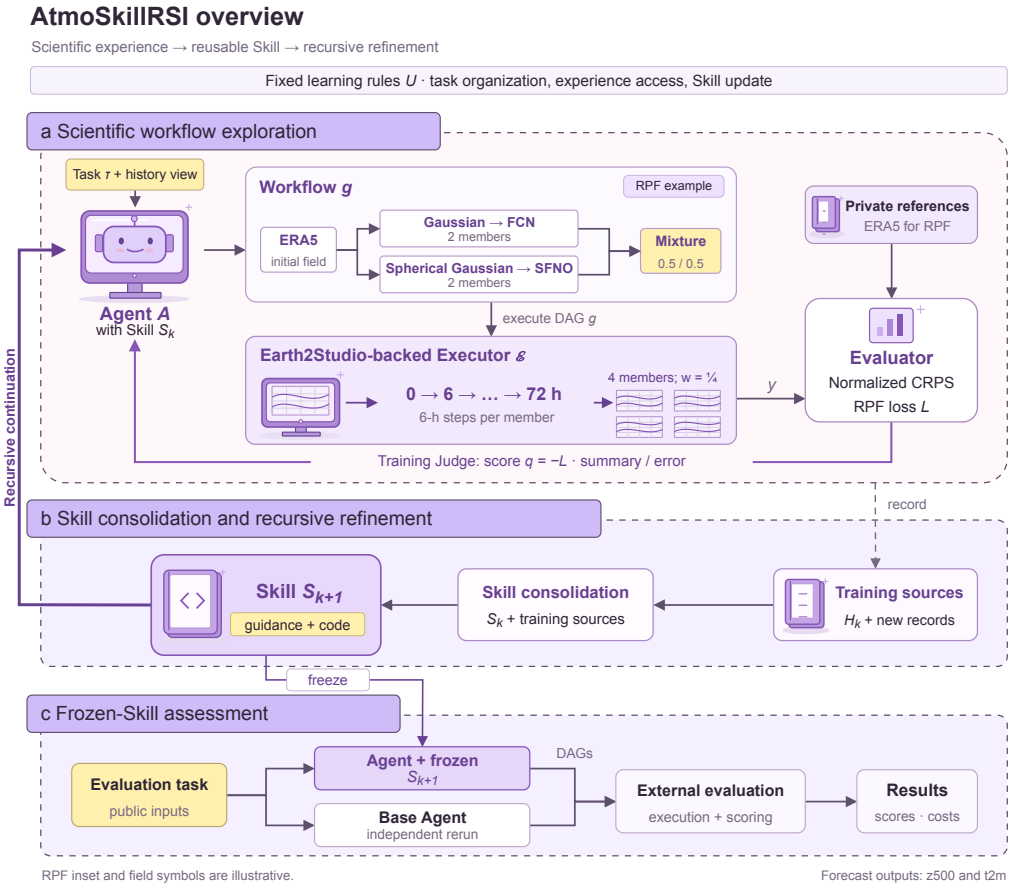

Figure 1: **AtmoSkillRSI overview.** (a) Under fixed learning rules U, an agent proposes executable atmospheric workflow DAGs, which are run by the Executor and scored by a scientific evaluator; training feedback is returned to the learner. (b) Training experience and the incoming Skill are consolidated into an updated Skill S_{k+1}, which can guide subsequent exploration. Initially, $S_0 = H_0 = \emptyset$. (c) Each learned Skill is frozen before assessment against an independently rerun Base under the same task interface and evaluator. Evaluation outcomes are kept separate from recursive training inputs. The RPF inset illustrates one executable workflow instance.

Task solving seeks a high-performing admissible workflow g. Skill learning instead seeks reusable procedural state S that improves the workflows constructed on subsequent tasks or learning stages. The learning rule U determines how experience is collected, organized, and consolidated into this persistent state. Thus, g, S, and U represent distinct levels of the learning process: workflow candidates change during exploration, Skills persist across tasks and stages, and the learning rule remains fixed within a configuration.

The Agent proposes workflow DAGs, which the Executor executes and the scientific evaluator scores. During training, the Judge returns valid scores and public feedback summaries; evaluation agents receive no Judge feedback. Appendix C.1 details execution and scientific feedback.

3.2 Atmospheric Workflow Task Families

We design four atmospheric task families specifically for our DAG-based scientific environment and iterative Skill learning: Regional Probabilistic Forecasting (RPF), Forecasting from Incomplete Initial Fields (FIIF), Wind Power Event Probability Prediction (WPEP), and Precipitation Spatial Structure Correction (PSSC). Each task family exposes composable scientific operations and family-

specific decisions, allowing agents to explore executable workflows and consolidate the resulting experience into reusable Skills that guide subsequent exploration and refinement. Appendix A provides the task-family design principles, detailed definitions, evaluation metrics, example workflows, and training and evaluation cases.

3.3 Experience and Skill Lifecycle

A Skill S packages reusable procedural guidance and supporting code for use across tasks. Training history H contains the recorded experience available to subsequent learning, including permitted agent interactions, candidate and feedback records, and derived training artifacts. The learning strategy U specifies task organization, prompts, experience access, and Skill distillation or synthesis. Within each experimental configuration, these learning rules remain fixed.

A learning stage $\mathcal{L}_U$ uses the incoming Skill and permitted history to collect new training experience and consolidate an updated persistent state:

$$(S_{k+1}, H_{k+1}) = \mathcal{L}_U\left(\mathcal{T}_{\text{train}}; S_k, H_k\right), \tag{2}$$

with $S_0 = H_0 = \emptyset$ denoting the absence of prior learned artifacts. All stages operate under the fixed execution constraints and safeguards described in Appendix B.

At the initial stage, exploration experience is consolidated into S_1, making the Skill an output of learning. Recursive continuation turns this output into an input to subsequent learning: the current Skill S_k guides new workflow exploration, and the resulting experience contributes to the update from S_k to S_{k+1}. We operationalize *skill-level recursive self-improvement* by requiring the persistent Skill S_k to influence the exploration that generates evidence for its own subsequent refinement.

Section 4 instantiates this dependency through Sequential and Task-Parallel learning and repeated Skill-guided refinement.

Frozen-Skill assessment. Each stage's Skill is frozen and assessed against an independently rerun Base under the same model, task interface, environment, and evaluator. Evaluation agents receive no Judge feedback, and evaluation outcomes are not used as continuation inputs (Appendix C.7).

4 Skill Learning and Recursive Refinement

We study two aspects of Equation 2: how a learning stage produces a Skill, and how the resulting Skill participates in further learning. Sequential and Task-Parallel learning organize experience within a stage, while recursive refinement feeds a learned Skill into subsequent stages (Figure 2). For each training task, the learner collects three successful scientific evaluations and distills reusable procedures and supporting code into a Skill package. Appendix C.8 specifies the feedback and session protocol.

4.1 Single-Stage Skill Learning

Both organizations collect scientific feedback and consolidate it into a shared Skill for subsequent use. Their difference lies in how experience moves between tasks and reaches consolidation.

4.1.1 Sequential Learning

Sequential processes training tasks in a prescribed order within one continuous agent session (Figure 2a). Experience accumulates across tasks, and the same learner distills a shared Skill after completing the full sequence.

4.1.2 Task-Parallel Learning

Task-Parallel (TP) uses independent learner sessions for the training tasks, each producing a local Skill; a fresh synthesizer then combines their outputs into a shared Skill (Figure 2b). Scientific evaluation occurs in the learner sessions; synthesis inputs and reporting are detailed in Appendix C.2.

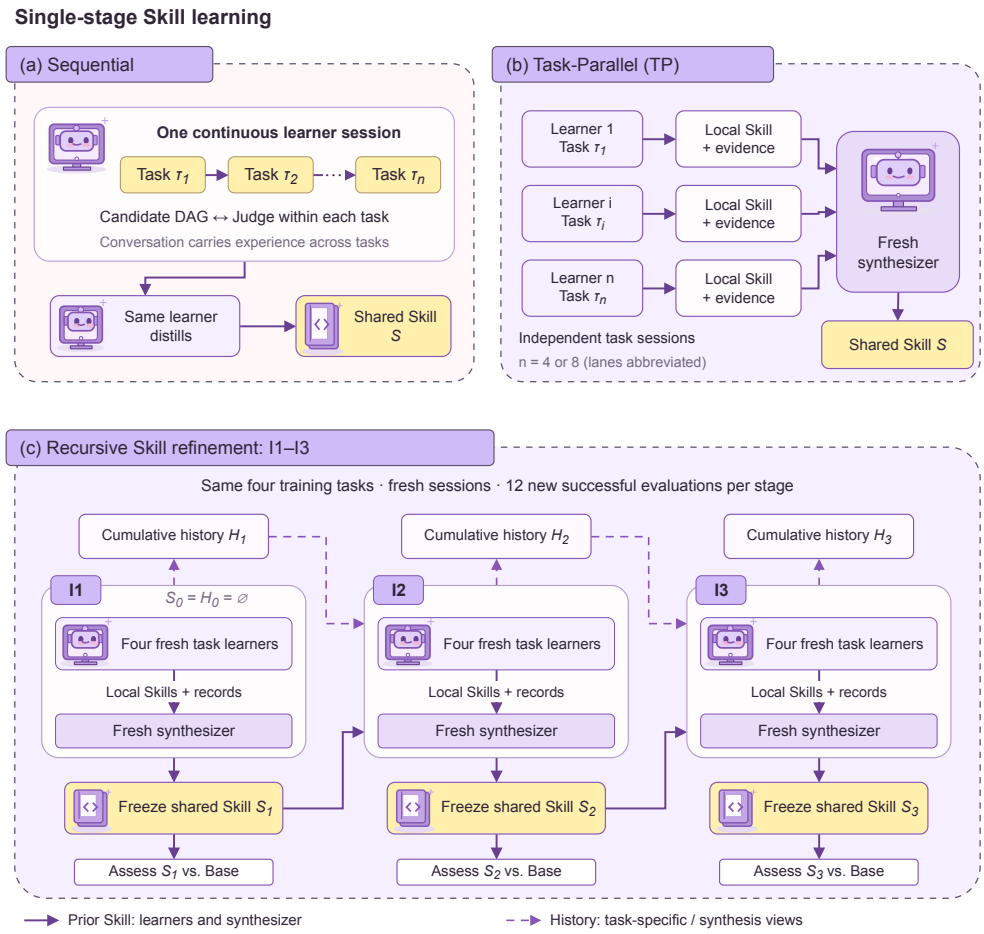

Figure 2: **Skill learning and recursive refinement.** (a) Sequential processes training tasks in one continuous session with experience carried across tasks. (b) Task-Parallel (TP) explores tasks independently before synthesizing their local Skills and evidence. (c) Recursive refinement repeats TP across I1–I3, with each stage receiving the preceding shared Skill and accumulated training history. Each resulting Skill is frozen for separate assessment.

We denote synthesis with recorded learner-interaction traces as TP and the configuration without these traces as TP_NI. Here, NI denotes no interaction-trace input to synthesis. Both configurations provide local Skills, task inputs, trial records, and available public feedback. TP additionally provides the recorded learner-interaction traces detailed in Appendix C.2. These traces therefore expose how candidate recommendations arose, beyond the candidate-level submissions and scores available to TP_NI. The recursive refinement experiments below use TP.

4.2 Recursive Skill Refinement: I1–I3

We instantiate recursive refinement through repeated Skill-guided TP learning (Figure 2c). We denote the three learning iterations as I1, I2, and I3. I1 produces the initial Skill, while I2 and I3 recursively refine it. The three iterations produce shared Skills S_1, S_2, and S_3, respectively, with $S_0 = H_0 = \emptyset$ at I1.

For the transitions I1→I2 and I2→I3, the preceding shared Skill is reused to guide new exploration on the same four training tasks. Each learner must then collect three *new* successful evaluations; historical trials provide evidence but do not count toward the new-stage quota. The synthesizer

combines the resulting local Skills and experience with the previous shared Skill and permitted history to produce S_k. Appendix C.3 specifies the role-specific inputs and session protocol.

The learning organization and update rule U remain fixed, while the persistent Skill and accumulated evidence evolve across stages. Appendix C.3 gives the full exploration–synthesis formulation.

The three stages yield frozen packages S_1, S_2, and S_3, each assessed against an independently rerun Base as defined in Section 3.3. Continuation inputs are restricted to the training artifacts described above, and evaluation outcomes are not fed back into later stages. Later stages differ from earlier stages in both the incoming Skill and the amount of accumulated training experience. At I2 and I3, the preceding learned Skill participates directly in generating the exploration experience used for its subsequent revision.

5 EXPERIMENTS

We study how scientific exploration experience forms reusable atmospheric workflow Skills, how their effectiveness varies across tasks and learning conditions, and whether persistent Skill reuse supports cumulative refinement. Our experiments address the following research questions:

RQ1. Can single-stage Skill learning produce effective reusable Skills across alternative experience organizations and atmospheric task families?

RQ2. How does Skill quality vary with training-case coverage and access to learner-interaction traces during synthesis?

RQ3. When learned Skills are reused to guide subsequent exploration and revision, do the subsequently learned Skills achieve better workflow performance?

The available results are development observations on fixed evaluation cases. RPF supports the experience-organization, training-evidence, and recursive-refinement analyses, while RPF, FIIF, WPEP, and PSSC support the task-family comparison. The evaluation cases have informed development, so all reported comparisons are treated descriptively.

Experimental overview. We compare frozen Skills with independently executed Base evaluations on fixed evaluation cases, reporting relative improvements in the corresponding scientific metrics. Appendix C.4 details the model and training configurations, evaluation cases, repetition protocol, metrics, Base references, and review criteria.

5.1 SINGLE-STAGE SKILL LEARNING

To answer *RQ1*, Figure 3 compares single-stage Sequential and Task-Parallel (TP) learning across experience organizations and atmospheric task families.

Results and analysis. On RPF, Sequential and TP achieve mean normalized CRPS values of 0.027481 and 0.028021, corresponding to 13.88% and 12.18% reductions relative to the pooled Base reference. Thus, both experience organizations produce learned Skills with better mean workflow performance than Base on the fixed RPF evaluation cases.

Across task families, improvement varies substantially. Sequential learning yields relative improvements of 16.79% on FIIF and 1.05% on WPEP, while TP yields 18.34% and 2.06%, respectively. On PSSC, both strategies improve the mean composite score by 1.76%; TP uses three runs, whereas the Sequential estimate is based on two successful runs. FIIF, WPEP, and PSSC have passed numerical replay, while semantic review and scientific acceptance remain pending.

Summarized Answer to *RQ1*. Single-stage Skill learning produces workflows with better mean performance than Base under both Sequential and Task-Parallel organization on RPF, while the magnitude of learned-Skill improvement varies across atmospheric task families.

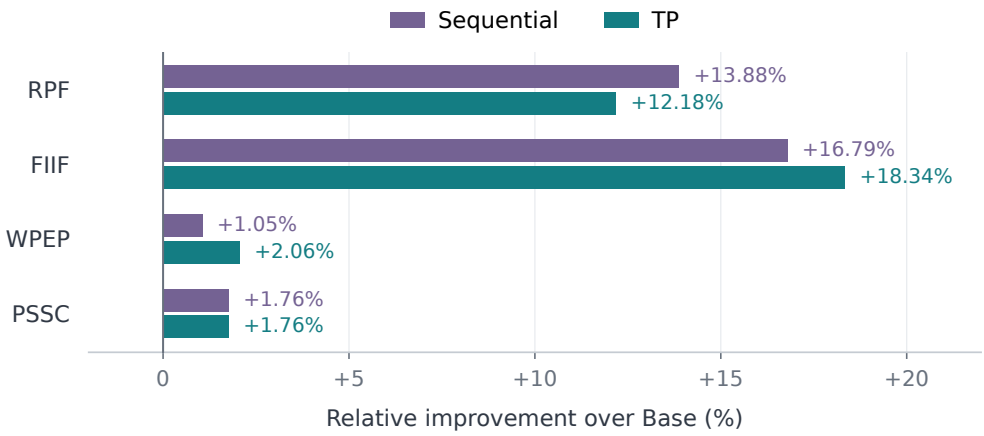

Figure 3: **Single-stage Skill learning across experience organizations and task families.** Bars show mean improvement over Base: lower normalized CRPS for RPF/FIIF, lower Brier score for WPEP, and higher composite score for PSSC. Each run is evaluated on four fixed cases.

5.2 Training Coverage and Synthesis Evidence

To answer *RQ2*, we examine two aspects of the experience available for Skill learning: training-case coverage and the interaction evidence available during synthesis.

Experimental setup. Figure 4(a) compares four- and eight-task training for Sequential and TP on the same four RPF evaluation cases. For each strategy, both the four-task and eight-task conditions are evaluated with three independently initialized learning runs ($N = 3$ per condition), with three successful evaluations required per training task. Figure 4(b) compares four-task TP_NI and TP, each with five learning runs ($N = 5$ per condition). Both receive local Skills and trial records with public feedback, while TP additionally receives recorded learner-interaction traces, as defined in Section 4.1.2. We report mean normalized CRPS for the resulting workflows and show the common Base reference in panel (b).

Results and analysis. Increasing the training set from four to eight tasks changes Sequential mean loss from 0.027358 to 0.027517 and TP mean loss from 0.028021 to 0.027881. Thus, broader training coverage does not produce a consistent improvement across the available runs. For synthesis evidence, TP_NI and TP achieve mean losses of 0.034564 and 0.028021, respectively. The trace-enabled TP configuration performs better in the available comparison, achieving substantially lower observed loss.

Summarized Answer to *RQ2*. Skill quality does not improve consistently with broader training-case coverage, while Skills synthesized with learner-interaction traces achieve better workflow performance in the available comparison.

5.3 Recursive Skill Refinement

The preceding experiments treat the learned Skill as the output of a single learning stage. To answer *RQ3*, we instead make the persistent Skill an input to subsequent learning and examine how workflow performance evolves across successive stages of reuse and revision. We operationalize recursive self-improvement at the Skill level: the TP learning rule and exploration–synthesis organization remain fixed, while the shared Skill produced at one stage guides subsequent exploration and is then revised using the resulting experience.

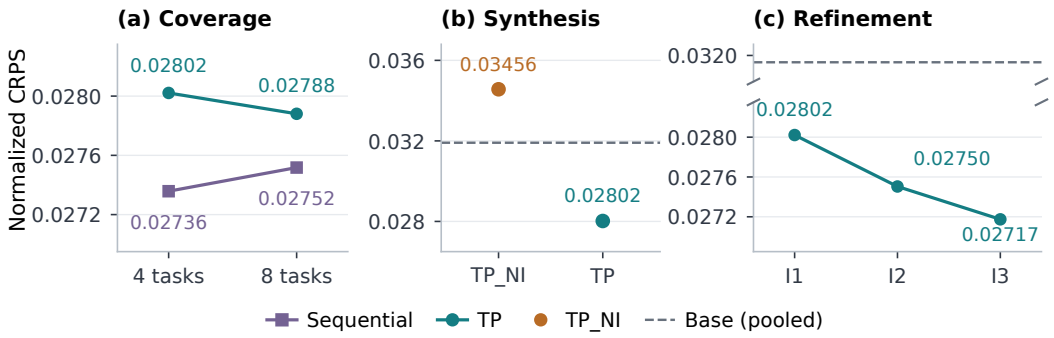

Figure 4: **Training coverage, synthesis evidence, and recursive Skill refinement.** All panels show mean normalized CRPS on the same four RPF evaluation cases (lower is better). (a) Four- versus eight-task training for Sequential and TP ($N = 3$ per condition). (b) TP_NI versus trace-enabled TP ($N = 5$ per condition). (c) Five TP lineages across refinement stages I1–I3. Dashed lines denote the pooled Base reference; diagonal marks in (c) indicate a broken vertical axis.

Experimental setup. We follow five TP learning lineages across three stages, denoted I1, I2, and I3, using the same RPF training and evaluation interface. I1 produces the initial shared Skill. For the transitions I1→I2 and I2→I3, the learner receives the prior shared Skill together with the permitted accumulated history, performs an additional three successful evaluations per training task, and synthesizes the next shared Skill from the resulting experience. The learning rule, TP organization, task interface, execution environment, and evaluator remain unchanged across stages. Each stage's Skill is frozen before evaluation on the same four RPF cases. Figure 4(c) shows the stage means. Relative improvements use the pooled Base reference of 0.031909 from Appendix C.4.

Results and analysis. Mean normalized CRPS decreases from 0.02802 at I1 to 0.02750 at I2 and 0.02717 at I3, corresponding to a 3.02% reduction from I1 to I3. Relative to the common Base mean, the reductions are 12.18%, 13.80%, and 14.84%, respectively, computed using the unrounded pooled reference. These results show continued improvement in mean workflow performance across successive Skill-refinement stages. Because later stages also include additional exploration and accumulated history, the observed trend should be interpreted as the outcome of the full refinement process rather than as an isolated effect of prior-Skill feedback.

Summarized Answer to *RQ3*. Across five TP learning lineages, mean workflow performance improves over three successive refinement stages under a fixed learning rule. This trend reflects the combined process of Skill reuse, accumulated history, and additional exploration.

6 Conclusion and Limitations

AtmoSkillRSI studies persistent Skill formation and recursive refinement in atmospheric workflows. Both Sequential and Task-Parallel learning improve mean performance on RPF, while effectiveness varies across task families. Broader training coverage does not consistently improve performance, and trace-enabled synthesis performs better in the available comparison. Mean performance also improves across successive refinement stages under a fixed learning rule. These findings are descriptive: evaluation cases informed system development, and later stages combine prior Skills with additional exploration and accumulated history. The observed trend therefore does not isolate the effect of prior-Skill feedback or establish generalization to unseen events. Fresh held-out evaluation and matched ablations remain necessary (Wang et al., 2026b). Appendix D provides the full discussion and limitations.

AI USE STATEMENT

Writing assistance. GPT-6 Astra was used to assist with improving the clarity, grammar, wording, conciseness, and organization of parts of the manuscript. AI-generated suggestions were reviewed and revised by the authors before inclusion in the final manuscript.

Retrieval and discovery. GPT-6 Astra was used to assist with literature retrieval and discovery, including generating search terms, identifying potentially relevant related work, and summarizing candidate references for further inspection. All cited works and bibliographic information were independently checked by the authors against the original sources.

Research ideation and execution. GPT-6 Astra and OpenAI Codex were used to support aspects of experimental design refinement, software implementation, debugging, and experimental workflow development and execution. Codex was primarily used as an AI coding agent for implementation and experimentation. AI-generated code and experimental artifacts were inspected and tested by the authors. AI systems also constitute part of the experimental methodology studied in this work, as described in the main text.

Drafting. GPT-6 Astra was used to assist with generating candidate wording, outlines, and early drafts of parts of the manuscript. These materials were subsequently reviewed, revised, and integrated by the authors, who determined the final structure, arguments, and scientific content.

All AI-assisted outputs, including manuscript text, code, and experimental artifacts, were reviewed and verified by the authors. The authors take full responsibility for the scientific claims, experimental design, analyses, interpretation of results, citations, and final content of the paper.

REFERENCES

Daniil A. Boiko, Robert MacKnight, Ben Kline, and Gabe Gomes. Autonomous chemical research with large language models. *Nature*, 624:570–578, 2023. doi: 10.1038/s41586-023-06792-0. URL https://www.nature.com/articles/s41586-023-06792-0.

Boris Bonev, Thorsten Kurth, Christian Hundt, Jaideep Pathak, Maximilian Baust, Karthik Kashinath, and Anima Anandkumar. Spherical Fourier neural operators: Learning stable dynamics on the sphere. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 2806–2823. PMLR, 2023. URL https://proceedings.mlr.press/v202/bonev23a.html.

Kaiyuan Chen, Shuangyu Xie, Letian Fu, Justin Yu, William Pacini, Sandeep Bajamahal, Hudson Kim, Jaimyn Drake, Daehwa Kim, Haoru Xue, Jonathan Francis, Christian Juette, Peter Schaldenbrand, Muhammet Yunus Seker, Ruwan Wickramarachchi, Uksang Yoo, Guanzhi Wang, Adithyavairavan Murali, Balakumar Sundaralingam, S. Shankar Sastry, Spencer Huang, Yuke Zhu, Linxi "Jim" Fan, and Ken Goldberg. GaP: A graph-as-policy multi-agent self-learning harness for variational automation tasks. *arXiv preprint arXiv:2607.05369*, 2026. doi: 10.48550/arXiv.2607.05369. URL https://arxiv.org/abs/2607.05369.

DeepSeek-AI. DeepSeek-V4: Towards highly efficient million-token context intelligence. *arXiv preprint arXiv:2606.19348*, 2026. doi: 10.48550/arXiv.2606.19348. URL https://arxiv.org/abs/2606.19348.

Yi Duan, Ying Liu, Zirui Tang, Haodong Chen, Jun Zhou, Yumou Liu, Bangrui Xu, Yukai Wu, Sidi Chen, Yuhan Zhou, Haoyu Wang, Xiaoyou Yu, Shaokun Han, Xuzhou Zhu, Le Zhou, Bolin Lu, Wei Zhou, Jiachen Liu, Nuozhou Fang, Jiaxin Tian, Ruoyu Chen, Yuxuan Li, Kai Zuo, Kaiyan Zhang, Qianyu Yang, Zijie Wang, Jiantao Qiu, Conghui He, Guoliang Li, Bowen Zhou, Zhiyuan Liu, Zhoufutu Wen, Jihua Kang, Xuanhe Zhou, and Fan Wu. The last AI built by humans: Toward genuine recursive self-improvement. *arXiv preprint arXiv:2609.11873*, 2026. doi: 10.48550/arXiv.2609.11873. URL https://arxiv.org/abs/2609.11873.

Earth2Studio Contributors. NVIDIA Earth2Studio. Open-source software, 2024. URL https://github.com/NVIDIA/earth2studio.

Tilmann Gneiting and Adrian E. Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007. doi: 10.1198/016214506000001437. URL https://doi.org/10.1198/016214506000001437.

Zijie Guo, Jiong Wang, Fenghua Ling, Wangxu Wei, Xiaoyu Yue, Zhe Jiang, Wanghan Xu, Jing-Jia Luo, Lijing Cheng, Yoo-Geun Ham, Fengfei Song, Pierre Gentine, Toshio Yamagata, Ben Fei, Wenlong Zhang, Xinyu Gu, Chao Li, Yaqiang Wang, Tao Chen, Wanli Ouyang, Bowen Zhou, and Lei Bai. A self-evolving ai agent system for climate science. *arXiv preprint arXiv:2507.17311*, 2025.

Hans Hersbach, Bill Bell, Paul Berrisford, Shoji Hirahara, András Horányi, Joaquín Muñoz-Sabater, Julien Nicolas, Carole Peubey, Raluca Radu, Dinand Schepers, Adrian Simmons, Cornel Soci, Saleh Abdalla, Xavier Abellan, Gianpaolo Balsamo, Peter Bechtold, Gionata Biavati, Jean Bidlot, Massimo Bonavita, Giovanna De Chiara, Per Dahlgren, Dick Dee, Michail Diamantakis, Rossana Dragani, Johannes Flemming, Richard Forbes, Manuel Fuentes, Alan Geer, Leo Haimberger, Sean Healy, Robin J. Hogan, Elías Hólm, Marta Janisková, Sarah Keeley, Patrick Laloyaux, Philippe Lopez, Cristina Lupu, Gabor Radnoti, Patricia de Rosnay, Iryna Rozum, Freja Vamborg, Sebastien Villaume, and Jean-Noël Thépaut. The ERA5 global reanalysis. *Quarterly Journal of the Royal Meteorological Society*, 146(730):1999–2049, 2020. doi: 10.1002/qj.3803. URL https://doi.org/10.1002/qj.3803.

Shengran Hu, Cong Lu, and Jeff Clune. Automated design of agentic systems. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://arxiv.org/abs/2408.08435.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. DSPy: Compiling declarative language model calls into self-improving pipelines. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=sY5N0zY5Od.

Yoonho Lee, Roshen Nair, Qizheng Zhang, Kangwook Lee, Omar Khattab, and Chelsea Finn. Meta-Harness: End-to-end optimization of model harnesses. *arXiv preprint arXiv:2603.28052*, 2026. doi: 10.48550/arXiv.2603.28052. URL https://arxiv.org/abs/2603.28052.

Chenyue Li, Wen Deng, Mengqian Lu, and Binhang Yuan. Atmossci-bench: Evaluating the recent advance of large language model for atmospheric science. *arXiv preprint arXiv:2502.01159*, 2025a.

Chenyue Li, Hyeonjae Kim, Wen Deng, Mengxi Jin, Wen Huang, Mengqian Lu, and Binhang Yuan. CLIMATEAGENT: Multi-agent orchestration for complex climate data science workflows. *arXiv preprint arXiv:2511.20109*, 2025b.

Shuaiqi Liu, Zhengkai Lin, Yuxiang Zhang, Yuanyi Ren, Yue Wu, Yongbin Li, Zheng Wang, Zhihang Fu, and Jieping Ye. The path to recursive self-improving agents: Foundation, framework, and future directions. *Preprints*, 2026. doi: 10.20944/preprints202608.0051.v1. URL https://www.preprints.org/manuscript/202608.0051.

Andres M. Bran, Sam Cox, Oliver Schilter, Carlo Baldassari, Andrew D. White, and Philippe Schwaller. Augmenting large language models with chemistry tools. *Nature Machine Intelligence*, 6:525–535, 2024. doi: 10.1038/s42256-024-00832-8. URL https://www.nature.com/articles/s42256-024-00832-8.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. CLIN: A continually learning language agent for rapid task adaptation and generalization. In *First Conference on Language Modeling*, 2024. URL https://www.cis.upenn.edu/~ccb/publications/clin-continual-learning-from-interactions.pdf.

Jingwei Ni, Yihao Liu, Xinpeng Liu, Yutao Sun, Mengyu Zhou, Pengyu Cheng, Dexin Wang, Xiaoxi Jiang, and Guanjun Jiang. Trace2Skill: Distill trajectory-local lessons into transferable agent skills. *arXiv preprint arXiv:2603.25158*, 2026.

Jaideep Pathak, Shashank Subramanian, Peter Harrington, Sanjeev Raja, Ashesh Chattopadhyay, Morteza Mardani, Thorsten Kurth, David Hall, Zongyi Li, Kamyar Azizzadenesheli, Pedram Hassanzadeh, Karthik Kashinath, and Animashree Anandkumar. FourCastNet: A global data-driven high-resolution weather model using adaptive Fourier neural operators. *arXiv preprint arXiv:2202.11214*, 2022. doi: 10.48550/arXiv.2202.11214. URL https://arxiv.org/abs/2202.11214.

Ziqing Qian, Jiaying Lei, Yifang Wang, and Nan Cao. HypoForge: A self-improving multi-agent framework for automated hypothesis generation and testing via scientific skill learning. *arXiv preprint arXiv:2608.25770*, 2026. doi: 10.48550/arXiv.2608.25770. URL https://arxiv.org/abs/2608.25770.

Zhe Ren, Yimeng Chen, Dandan Guo, Guowei Rong, Tonghui Li, R. B. Xiong, Qingfeng Lan, Wenyi Wang, Li Nanbo, Yibo Yang, Mingchen Zhuge, and Jürgen Schmidhuber. Self-improvements in modern agentic systems: A survey. *arXiv preprint arXiv:2607.13104*, 2026. doi: 10.48550/arXiv.2607.13104. URL https://arxiv.org/abs/2607.13104.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 36, pp. 8634–8652. Curran Associates, Inc., 2023. doi: 10.52202/075280-0377. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/1b44b878bb782e6954cd888628510e90-Abstract-Conference.html.

Sumanth Varambally, Marshall Fisher, Jas Thakker, Yiwei Chen, Zhirui Xia, Yasaman Jafari, Ruijia Niu, Manas Jain, Veeramakali Vignesh Manivannan, Zachary Novack, Luyu Han, Srikar Eranky, Salva Rühling Cachay, Taylor Berg-Kirkpatrick, Duncan Watson-Parris, Yi-An Ma, and Rose Yu. Zephyrus: An agentic framework for weather science. *arXiv preprint arXiv:2510.04017*, 2025.

Guanzhi Wang, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar. Voyager: An open-ended embodied agent with large language models. *Transactions on Machine Learning Research*, 2024a. URL https://openreview.net/forum?id=ehfRiF0R3a.

Hao Wang, Jindong Han, Wei Fan, and Hao Liu. Climagent: Llm as agents for autonomous open-ended climate science analysis. *arXiv preprint arXiv:2604.16922*, 2026a.

Kaiqi Wang, Wenjin Hou, Yuchen Yan, Hongrui Jia, Zhisheng Zhong, Botao Ren, Yifei Chen, Songyang Zhang, Yongliang Shen, Jun Xiao, Yi Yang, Yueting Zhuang, and Hehe Fan. Diving into reliable self-evolving agents: A survey. *Preprints*, 2026b. URL https://www.preprints.org/manuscript/202609.0913.

Zhiruo Wang, Graham Neubig, and Daniel Fried. TroVE: Inducing verifiable and efficient toolboxes for solving programmatic tasks. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pp. 51177–51191. PMLR, 2024b. URL https://proceedings.mlr.press/v235/wang24az.html.

Mert Yuksekgonul, Federico Bianchi, Joseph Boen, Sheng Liu, Pan Lu, Zhi Huang, Carlos Guestrin, and James Zou. Optimizing generative AI by backpropagating language model feedback. *Nature*, 639:609–616, 2025. doi: 10.1038/s41586-025-08661-4. URL https://www.nature.com/articles/s41586-025-08661-4.

Eric Zelikman, Eliana Lorch, Lester Mackey, and Adam Tauman Kalai. Self-taught optimizer (STOP): Recursively self-improving code generation. In *First Conference on Language Modeling*, 2024. URL https://arxiv.org/abs/2310.02304.

Jenny Zhang, Shengran Hu, Cong Lu, Robert Lange, and Jeff Clune. Darwin Gödel machine: Open-ended evolution of self-improving agents. *arXiv preprint arXiv:2505.22954*, 2025. doi: 10.48550/arXiv.2505.22954. URL https://arxiv.org/abs/2505.22954.

Andrew Zhao, Daniel Huang, Quentin Xu, Matthieu Lin, Yong-Jin Liu, and Gao Huang. ExpeL: LLM agents are experiential learners. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 19632–19642, 2024. doi: 10.1609/aaai.v38i17.29936. URL https://ojs.aaai.org/index.php/AAAI/article/view/29936.

Tong Zheng, Xidong Wu, Zheng Zhang, Zhankui He, Chaoyi Zhang, Benjamin Coleman, Ruoqiao Wei, Di Bai, Haolin Liu, Rui Liu, Xue Wang, Yue Zhuan, Wang-Cheng Kang, Renkai Xiang, Heng Huang, Xinwu Cheng, and Yunsong Guo. Dream-RSI: Recursive self-improvement through evolving worlds. *arXiv preprint arXiv:2609.14858*, 2026. doi: 10.48550/arXiv.2609.14858. URL https://arxiv.org/abs/2609.14858.

Leijun Zhou, Zhihao Liu, Xiang Qu, Chenxu Liu, Yifei Liu, Yanke Yu, Jingzhe Xu, Xuejun Wu, Buyue Qian, Xi Chen, Yaowei Zheng, and Junhao Hu. GDPevo: Evaluating agent self-evolution on real business tasks. *arXiv preprint arXiv:2608.03764*, 2026.

A ATMOSPHERIC TASK DEFINITIONS

This appendix details the atmospheric tasks; execution safeguards appear in Appendix B.

A.1 TASK DESIGN PRINCIPLES

Our task design follows four principles. **Scientific relevance and meaningful decisions:** each task specifies an atmospheric objective and exposes composable workflow operations through which the agent can influence the outcome. **Learnable variation:** cases should offer observable conditions that motivate different workflow choices, with training-side checks assessing whether these choices produce distinguishable outputs and scores. **Transfer and information separation:** Skills should capture reusable decision rules; evaluating their transfer requires independent weather cases, with future verification data kept private and final-test feedback excluded from development. **Reproducible evaluation:** task definitions, metrics, admissible operations, and resource limits are fixed for each experiment, while execution enforces physical and data constraints and records provenance, failures, and actual resource use. These principles guide task construction; evidence of execution validity, numerical correctness, and scientific improvement is assessed separately.

A.2 ATMOSPHERIC WORKFLOW TASKS

We instantiate the framework with four atmospheric tasks: Regional Probabilistic Forecasting (RPF), Forecasting from Incomplete Initial Fields (FIIF), Wind Power Event Probability Prediction (WPEP), and Precipitation Spatial Structure Correction (PSSC). Together, they expose workflow decisions involving forecast-model and ensemble construction, initial-field repair, wind-power event probabilities, and precipitation spatial correction. All four use the common task interface, execution mechanism, scientific evaluation, and Skill lifecycle defined above.

Table 1: **Atmospheric task families.** Each task specifies its required outputs and scientific evaluator. Arrows indicate the direction of better performance.

Task	Scientific objective	Task-specific workflow decisions	Primary metric
RPF	Regional probabilistic forecasting of `z500` and `t2m`	Model selection, perturbations, ensemble allocation, regional mixing, historical calibration	Normalized CRPS ↓
FIIF	72-h forecasting from incomplete initial fields	Spatial or historical repair, background residual interpolation, smoothing and shrinkage; observed cells preserved	Normalized CRPS ↓
WPEP	Probabilities of wind-power ramps and low-power events	Historical residual windows, forecast conditioning, site pooling, scenario count and event aggregation	Brier score ↓
PSSC	Precipitation spatial structure and amount correction	Displacement, conservative smoothing, amount scaling and operator composition	Composite precipitation score ↑

RPF and FIIF use normalized continuous ranked probability score (CRPS), and WPEP uses Brier score; these losses are lower-is-better, and the training Judge returns their negatives. PSSC uses a higher-is-better composite precipitation score, returned directly by the Judge. RPF provides the main setting for the experience-organization, training-coverage, synthesis-evidence, and recursive-refinement experiments. All four tasks enter the task-family comparison. The following subsections specify their inputs, workflow decisions, outputs, and scoring rules. Each task includes a concrete example DAG; its parameter values illustrate admissible workflow choices. Appendix C.4 gives the learning configurations, repeated runs, and evaluation aggregation.

A.3 REGIONAL PROBABILISTIC FORECASTING (RPF)

RPF asks an agent to construct a probabilistic forecast from an ERA5 initial condition, with ERA5 fields providing the verifying reference (Hersbach et al., 2020). The required variables are 500-hPa

geopotential (`z500`, $\mathrm{m^2\,s^{-2}}$) and 2-m temperature (`t2m`, K). Registered models include FourCastNet (FCN) and the spherical Fourier neural operator (SFNO) (Pathak et al., 2022; Bonev et al., 2023). The agent chooses model branches, initial perturbations, ensemble sizes, regional mixture weights, and operations using a registered historical calibration package. Forecasts retain their ensemble members and probability weights on a common global 720×1440 grid at 0.25° spacing; scoring uses the task's prescribed regions and lead times. Mixture operators combine predictive distributions while preserving their members.

Figure 5 illustrates a candidate workflow.

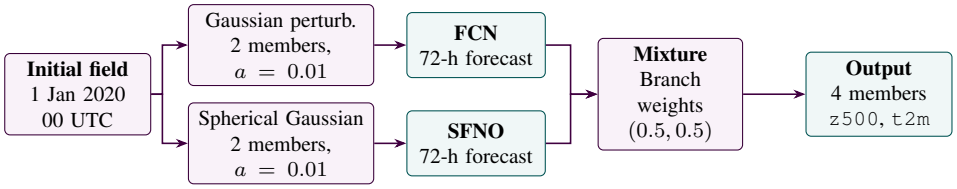

Figure 5: **An illustrative RPF workflow.** For the 1 January 2020 case, two perturbation branches supply two members each to FCN and SFNO. The coefficient $a = 0.01$ scales each input channel's registered checkpoint standard deviation. Equal branch weights yield four forecast members, each with probability $1/4$. The seven-node DAG combines the two model branches into one predictive distribution.

We evaluate predictive distributions using the continuous ranked probability score (CRPS) (Gneiting & Raftery, 2007), expressed as a loss. For a weighted empirical distribution $\widehat{P} = \sum_{m=1}^{M} w_m \delta_{x_m}$ and verifying value z,

$$C(\widehat{P}, z) = \sum_{m=1}^{M} w_m |x_m - z| - \frac{1}{2} \sum_{m,n=1}^{M} w_m w_n |x_m - x_n|, \qquad w_m \geq 0, \quad \sum_m w_m = 1. \tag{3}$$

Weights may vary spatially. Let $\mathcal{C}_\tau$ contain the scored region–lead–variable combinations. The nonnegative normalized task loss is

$$L_\tau(g) = \frac{1}{|\mathcal{C}_\tau|} \sum_{(r,\ell,v)\in\mathcal{C}_\tau} \frac{1}{\sigma_v} \sum_{p\in r} a_{r,p}\, C(\widehat{P}_{p,\ell,v}, z_{p,\ell,v}), \qquad q = -L_\tau(g), \tag{4}$$

where $a_{r,p}$ is proportional to the cosine of latitude and sums to one within region r. Each σ_v is a fixed population standard deviation computed with equal grid-cell weights over 2018–2019 historical reference fields: 24 initializations at 00 UTC on the first day of each month and three lead times (72, 120, and 168 hours). Thus each component has equal weight after variable normalization, lower L_τ is better, and the Judge returns its negative as q in Equation 1.

A.4 Forecasting from Incomplete Initial Fields (FIIF)

FIIF asks the agent to construct a 72-hour forecast from an initial atmospheric field with missing values. Public inputs contain the damaged current field and registered past fields or monthly reference fields. The agent chooses how to repair missing values using spatial neighbors, previous fields, or reference-based anomaly reconstruction, then constructs a forecast using the registered models and workflow operators. A background-repair operator interpolates visible current-minus-background residuals into missing regions, with selectable smoothing and shrinkage toward the historical background. The repair operation preserves the initially observed cells and must supply finite values for the model's required input channels. Subsequent permitted perturbations may change the repaired field.

Every forecast branch must descend from the authorized damaged current field through a repair operation. Historical and reference fields can support repair but cannot replace that current-field lineage; an intact current field is not exposed as a public input. The evaluator scores 72-hour predictive distributions of `z500` and `t2m` over the Global and Eurasia regions using normalized CRPS and

fixed historical scales, following Equation 4. The metric therefore evaluates forecast quality after repair and model execution.

Figure 6 illustrates repair followed by ensemble forecasting.

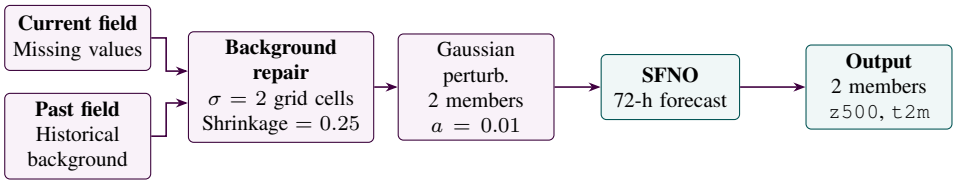

Figure 6: **An illustrative FIIF workflow.** The six-node DAG repairs the damaged current field using a past background, preserving observed cells. Two Gaussian-perturbed members enter SFNO for a 72-hour forecast. The output distribution for `z500` and `t2m` is evaluated by normalized CRPS; the diagram specifies the repair and perturbation parameters.

A.5 Wind Power Event Probability Prediction (WPEP)

WPEP asks the agent to predict upward ramps, downward ramps, and low-power events from 100-m wind forecasts at the prescribed sites. Public inputs include forecast wind components and paired historical forecast–reference trajectories whose valid times precede the current initialization. The agent composes wind-speed conversion, historical residual fitting, scenario construction, a fixed turbine power curve, and event-probability aggregation. Its choices include the historical window, conditioning on similar forecast trajectories, site-specific or shared residual pooling, and the number of scenarios. Each scenario retains a complete historical residual trajectory, preserving temporal dependence when computing changes in power.

The fixed power curve maps wind speed to normalized power in $[0, 1]$. For each scenario, an upward or downward ramp occurs when the change between adjacent output times is at least the prescribed positive threshold or at most its negative, respectively; a low-power event occurs when power is at or below the prescribed low-power threshold. The output contains mean normalized power and the fraction of scenarios satisfying each event. Future reference trajectories and their event indicators remain private to the evaluator.

The primary loss is the Brier score averaged uniformly over all required time–site–event combinations:

$$L_\tau(g) = \frac{1}{|\mathcal{I}_\tau|} \sum_{i \in \mathcal{I}_\tau} (p_i - o_i)^2, \qquad q = -L_\tau(g), \tag{5}$$

where p_i is the predicted probability, $o_i \in \{0, 1\}$ is the verifying indicator, and $\mathcal{I}_\tau$ indexes upward-ramp, downward-ramp, and low-power events across sites and output times. Mean absolute error in normalized power is auxiliary and does not enter the primary loss.

Figure 7 illustrates the historical-scenario construction of event probabilities.

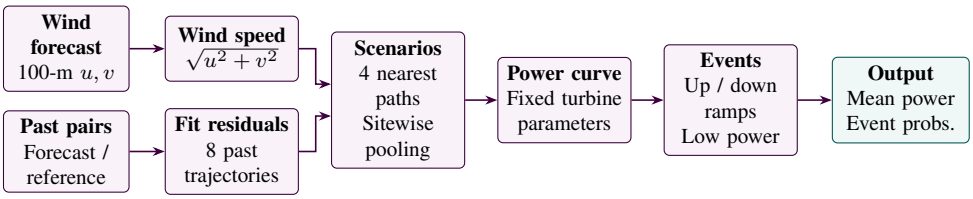

Figure 7: **An illustrative WPEP workflow.** The eight-node DAG selects four complete residual paths from eight past forecast–reference pairs using forecast-speed similarity and applies them sitewise. A fixed turbine curve converts wind scenarios to normalized power. Aggregation yields mean power and upward-ramp, downward-ramp, and low-power probabilities at each site and time; the probabilities are evaluated by Brier loss.

A.6 Precipitation Spatial Structure Correction (PSSC)

PSSC asks the agent to correct the spatial distribution and amount of gridded six-hour precipitation forecasts. The agent composes displacement, spatial smoothing, and multiplicative amount correction. Its choices include northward and eastward displacement, the smoothing scale, the amount factor, and the composition of these operations. The displacement and smoothing operators restore the area-weighted precipitation total at each lead time when the transformed field has positive total precipitation; the amount factor explicitly adjusts that total. Outputs must remain finite and nonnegative, with the prescribed grid, accumulation period, and valid times. The verifying ERA5 precipitation fields remain private to the evaluator.

The evaluator combines spatial agreement and an amount-bias penalty:

$$q = C_{\text{precip}} = 0.8\,\overline{\text{FSS}} + \frac{0.2}{1+v}, \qquad v = \frac{|\langle x\rangle_a - \langle z\rangle_a|}{\max(1\,\text{mm}, \langle z\rangle_a)}, \tag{6}$$

where x and z are forecast and verifying precipitation, and $\langle\cdot\rangle_a$ averages over valid times and grid cells with cosine-latitude weights. The mean Fractions Skill Score (FSS) averages spatial agreement over the registered precipitation thresholds and neighborhood sizes, with each component aggregating over the scored times and grid cells. Higher scores indicate better performance; reported relative improvement is a score gain.

Figure 8 illustrates a composition of the three correction operations.

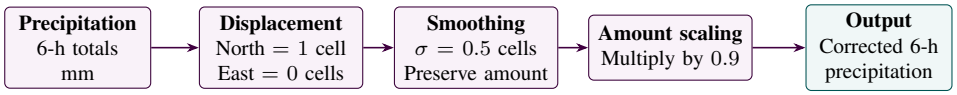

Figure 8: **An illustrative PSSC workflow.** The five-node DAG shifts precipitation one cell north, smooths with a half-cell Gaussian scale, and rescales by 0.9. Displacement and smoothing preserve the amount when the transformed total is positive; scaling adjusts it. The evaluator combines spatial agreement and amount bias.

A.7 Training and Evaluation Cases

For RPF, Table 2 defines the four- and eight-task training configurations. The four-task set samples quarterly initializations in 2020; the eight-task set adds a second date in each quarter while preserving that quarter's lead times and scoring region. Four evaluation cases use the corresponding quarterly dates in 2021. The fixed historical calibration package and scoring scales use 2018–2019 data, and future verifying fields remain private to the evaluator.

Table 2: **RPF training and evaluation cases.** All dates are at 00 UTC. The eight-task set combines the first two columns. Row order corresponds to training IDs 001–004, added IDs 005–008, and evaluation IDs 001–004. All cases require both variables.

Train (2020)	Added (2020)	Eval. (2021)	Leads (h)	Scoring region
1 Jan	15 Jan	1 Jan	72	Global
1 Apr	15 Apr	1 Apr	72, 120	Tropics
1 Jul	15 Jul	1 Jul	72, 120, 168	N. Hemisphere
1 Oct	15 Oct	1 Oct	72	East Asia

Tropics: 20°S–20°N; Northern Hemisphere: 0°–90°N; East Asia: 10°–60°N, 100°–150°E. Output fields cover the full global grid.

FIIF uses four quarterly initializations in 2022 for training and the corresponding dates in 2023 for evaluation, with frozen point, patch, and mixed missingness patterns. WPEP and PSSC each use their own four training and four evaluation cases, with task-specific event definitions, historical inputs, and scoring settings fixed within each family. Learning repeats reuse the corresponding weather cases, so additional repeats provide variation in learning runs rather than additional weather coverage. The four evaluation cases have also informed development; they support the fixed-case comparisons reported here, while broader generalization requires fresh evaluation cases.

B EXECUTION SAFEGUARDS AND RESOURCE USE

We impose execution limits to guard against stalled runs, excessive retries, and unintended resource consumption. These limits serve as operational safeguards rather than a protocol for matching computational expenditure across methods. The following sections report GPU occupancy and specify execution-limit units and standard configurations.

B.1 HARDWARE AND GPU TIME PER WORKFLOW

Scientific GPU execution uses a shared server with eight MetaX C550 GPUs, each with 64 GiB of device memory. Each GPU-backed workflow leases one device. Table 3 reports mean GPU occupancy per scientific workflow execution, computed as $\sum_{j=1}^{N}(t_j^{\text{release}} - t_j^{\text{start}})/(3600N)$ GPU-hours. The interval includes worker startup, model loading, data handling, scientific computation, and cleanup while the device is held; queueing and Agent deliberation outside the lease are excluded. It measures device occupancy rather than GPU-kernel active time.

Table 3: **Mean GPU occupancy per scientific workflow.** Each recorded execution holds one GPU. WPEP and PSSC use CPU operators, so their scientific workflows consume no GPU time. Hosted language-model inference is outside this accounting.

Task	Timed executions	GPU-min/workflow	GPU-h/workflow
RPF	32	3.405	0.05676
FIIF	96	2.145	0.03575
WPEP	CPU	0	0
PSSC	CPU	0	0

The timing cohort comprises two completed RPF Sequential runs and the six FIIF runs used in Figure 3 (three Sequential and three TP). Each run contributes 12 successful training executions and four Skill-evaluation executions, weighted equally. The RPF timing sample uses the two later four-training-case runs with complete per-workflow lease records. These measurements describe successful workflow executions in this cohort.

B.2 SAFEGUARD DEFINITIONS

The stage parameter B_k in Equation 2 collects the configured execution limits for learning stage k. Agent-request limits count requests admitted to the model provider and are enforced at the session and, for node-based learning, shared-stage levels. An Agent node denotes a learner or synthesizer session; a DAG node denotes a scientific operation within a candidate workflow. Wall-clock limits bound session, stage, and whole-run duration, while the output-token limit applies to each model response. Training Judge calls are counted separately: a failed started call consumes the call allowance but does not satisfy the successful-feedback requirement. The learning protocol requires three successful evaluations per training task per stage, as defined in Section 4. Evaluation sessions construct their answers without training-Judge feedback and have their own request and time limits. Configured ceilings and measured consumption are recorded separately.

Scientific workflow limits. Each RPF candidate may contain at most 24 DAG nodes, four forecast nodes, and eight member trajectories across all forecast nodes. One *model-step credit* represents one member advanced by six hours with either FCN or SFNO. A forecast node with M_f members and horizon H_f hours costs $M_f H_f/6$ credits; costs sum across forecast nodes, including cached computations. These are logical workflow credits, whose accounting is distinct from measured execution time. Table 4 gives the limits for the case groups in Table 2. The candidate in Figure 5 uses $2 \times 2 \times (72/6) = 48$ of the available 96 credits.

B.3 STANDARD AGENT CONFIGURATIONS

For standard four-task TP, the shared learning limit is 240 requests, with at most 48 for each of four learners and one synthesizer. The recorded TP preparation profile allows 7,200 seconds for learning,

Table 4: **RPF per-candidate workflow limits.** Every candidate allows at most 24 DAG nodes, four forecast nodes, and eight member trajectories. Credit limits apply to all training and evaluation dates in the corresponding case row of Table 2.

Case group	Scoring region	Model-step credits
January	Global	96
April	Tropics	160
July	Northern Hemisphere	224
October	East Asia	96

3,600 seconds per learner or synthesizer, and 900 seconds per evaluation session. Its whole-run limit is 10,800 seconds, and the training Judge-call limit is 24, including failed calls. RPF model responses are capped at 8,192 output tokens each. The TP timing and Judge-call profile describes that configuration; other historical configurations retain their own recorded settings. For example, TP_NI synthesis uses 40 requests and 900 seconds, while standard TP synthesis uses 48 requests and 3,600 seconds. Within each archived paired evaluation, Skill and Base use the same configured evaluation limits.

C Execution and Learning Protocol Details

C.1 Workflow Execution and Scientific Feedback

The Agent proposes DAGs through the public task interface. The Executor validates graph structure, resolves registered scientific assets, executes the requested operations, and checks the resulting artifacts against the output requirements. Scientific computation uses Earth2Studio's model and data interfaces together with task-specific operators (Earth2Studio Contributors, 2024). The evaluator computes the declared native metric m_τ from the generated artifacts and private reference data.

During training, the Judge J wraps submission, execution, and scoring. A successful evaluation returns the higher-is-better score $q_\tau = d_\tau m_\tau$ defined in Equation 1 together with a public feedback summary, while an invalid or failed execution returns an error without a valid scientific score. The learning rule U determines how this feedback influences subsequent workflow proposals and Skill consolidation. Execution safeguards are fixed within a configuration and are detailed in Appendix B.

C.2 Task-Parallel Synthesis Inputs and Reporting

The synthesizer receives local Skills, task inputs, candidate-level trial records, and available public Judge feedback. Trial records contain submitted candidates and their evaluation outcomes, but not the full learner interaction trajectory. The synthesizer uses file tools to inspect these sources and writes an audit report explaining which procedures it retained, combined, revised, or discarded. Scientific evaluation occurs only in the learner sessions; the synthesizer has no Judge access. Evaluation agents later receive only the shared Skill, while the synthesis report remains an audit artifact.

TP additionally provides recorded learner conversations, tool interactions, intermediate diagnostics, and failures associated with the local exploration process.

C.3 Recursive Continuation Inputs

For the transitions I1→I2 and I2→I3, the preceding shared Skill is reused to guide new exploration on the same four training tasks. Each continuation stage starts fresh learner sessions. A learner receives the complete previous shared Skill S_{k-1} together with the permitted history for its own task from earlier stages. Each learner must then collect three *new* successful evaluations; historical trials provide evidence but do not count toward the new-stage quota. The synthesizer receives the current local Skills and trial records, the previous shared Skill, synthesis-visible history accumulated across training tasks, and earlier synthesis reports. It then produces a complete revised shared Skill S_k and records the rationale for the revision.

Writing $s_{k,i}$ for the local Skill produced for task τ_i and $D_{k,i}$ for the newly collected candidate, feedback, and interaction records, stage $k \in \{1, 2, 3\}$ is

$$\begin{aligned}(s_{k,i}, D_{k,i}) &= \mathrm{Explore}_U\left(\tau_i; S_{k-1}, H_{k-1,i}\right), \quad i = 1, \ldots, n, \\ S_k &= \mathrm{Synthesize}_U\left(S_{k-1}, H_{k-1}^{\mathrm{syn}}, \{(s_{k,i}, D_{k,i})\}_{i=1}^{n}\right),\end{aligned} \tag{7}$$

where $n = 4$, $H_{k-1,i}$ denotes the task-specific history visible to learner i, and H_{k-1}^{syn} denotes the synthesis-visible history accumulated from prior stages, including permitted task-level records and previous synthesis reports. Task inputs are also available to synthesis. The learning organization and update rule U remain fixed, while the persistent Skill and accumulated evidence evolve across stages.

A completed lineage contains 12, 24, and 36 cumulative successful training evaluations at I1, I2, and I3, respectively.

C.4 Experimental Setup

Tasks and runs. The four task families are RPF, FIIF, WPEP, and PSSC. They expose workflow decisions involving forecast-model and ensemble construction, initial-field repair, wind-power event probabilities, and precipitation spatial correction, respectively; Appendix A.2 gives the task-family overview. RPF uses four quarterly training cases from 2020, with an eight-case extension that adds a second date in each quarter, and four evaluation cases from 2021 (Appendix A.7). The FIIF, WPEP, and PSSC records used in Section 5.1 each contain four training and four evaluation cases. A repeated run learns a new Skill and evaluates it on the same four cases; repeated runs therefore provide independent learning trajectories but do not add independent weather cases. Archived Skill evaluations have independently executed Base counterparts; each comparison in Section 5 specifies whether it uses the original paired Base or the pooled reference. Equal repeat indices across different strategies do not imply matched random conditions. For FIIF Sequential and both WPEP wind-power strategies, Figure 3 uses the original repeats 1 and 2 and one independently initialized supplementary repeat 3, with the same frozen tasks, budgets, and shared Base within each family. The reported repeat 3 was selected after both the initial and supplementary test results were observed; these are descriptive selected-run estimates, not a prespecified three-run sample. The original executions remain archived. FIIF TP uses the repeat 1 retry and original repeats 2 and 3. The additional WPEP repeat 4 is listed in the complete results but is outside this figure's three-run aggregation. RPF runs are configured with `deepseek-v4-pro` (DeepSeek-AI, 2026) through the DeepSeek provider with minimal reasoning. Execution safeguards are described in Appendix B.

Pooled baseline reference. We use the pooled RPF Base mean $\overline{L}_{\mathrm{Base,pool}} = 0.031909$ as a common descriptive reference across RPF configurations. It averages 124 independently executed Base case losses equally; Appendix C.5 details the accounting. Comparisons explicitly using this pooled reference share that denominator; other reported reductions retain their stated Base references. FIIF, WPEP, and PSSC each use four independently executed Base evaluations, one per evaluation case, shared across the reported learning runs within that family.

Metrics and aggregation. RPF uses positive normalized CRPS from Equation 4; FIIF also uses normalized CRPS. WPEP uses mean Brier score for wind-power event probabilities, converted from the recorded negative score. Lower values indicate better forecasts for these loss metrics. For a group of N complete learning runs, each evaluated on four cases, we average over the $4N$ recorded case–run pairs. Using the same Base reference and aggregation for both scores and native metrics, the unified relative improvement is

$$\Delta = 100\,\frac{\overline{q}_{\mathrm{Skill}} - \overline{q}_{\mathrm{Base}}}{|\overline{q}_{\mathrm{Base}}|}, \qquad \overline{q}_{\mathrm{Base}} \neq 0. \tag{8}$$

This percentage is undefined for a zero Base mean. For RPF, FIIF, and WPEP, substituting $q = -L$ gives the relative loss reduction

$$\Delta = 100\,\frac{\overline{L}_{\mathrm{Base}} - \overline{L}_{\mathrm{Skill}}}{\overline{L}_{\mathrm{Base}}}. \tag{9}$$

This is a ratio of group means. PSSC instead uses the composite precipitation score $C_{\mathrm{precip}} = 0.8\,\overline{\mathrm{FSS}} + 0.2/(1 + v)$, where FSS is averaged over the registered thresholds and neighborhoods and v is the absolute area-weighted mean precipitation bias divided by the larger of 1 mm

and the area-weighted observed mean. With $q = C_{\mathrm{precip}}$, Equation 8 gives $100(\overline{C}_{\mathrm{precip,Skill}} - \overline{C}_{\mathrm{precip,Base}})/|\overline{C}_{\mathrm{precip,Base}}|$, a score gain rather than an error reduction. Wins, losses, and ties compare each Skill-conditioned workflow with its corresponding Base on the same case: $q_{\mathrm{Skill}} > q_{\mathrm{Base}}$, $q_{\mathrm{Skill}} < q_{\mathrm{Base}}$, and equality, respectively. We report learning-run counts and treat the comparisons descriptively.

Execution and review. The score tables contain complete scored pairs, including documented recoveries. They do not estimate the failure rate across all attempted runs. Numerical replay, execution completion, and semantic review of the learned Skill are treated as distinct checks. Because the evaluation cases have informed development, the findings are limited to these development cases.

C.5 Pooled Baseline Accounting

Across the archived RPF comparisons, 31 complete learning-run or refinement-stage groups provide 124 independently executed Base case losses: 31 for each of the same four RPF cases. Counting each Base case execution once, their pooled mean normalized CRPS is $\overline{L}_{\mathrm{Base,pool}} = 0.031909$. We use this value as a common descriptive reference across RPF configurations. It averages the 124 case losses equally. Comparisons explicitly using this pooled reference share that denominator; other reported reductions retain the original group-specific Base means.

C.6 Extension to Learning-Strategy Evolution

Our current formulation intentionally holds the learning strategy U fixed while the persistent Skill S and retained history H evolve. Accordingly, AtmoSkillRSI studies recursive self-improvement at the Skill level rather than modification of the learning mechanism itself.

A natural extension is an outer meta-learning loop that uses outcomes from multiple learning runs to revise U, including task ordering, experience organization, feedback access, and Skill distillation or synthesis rules. Each candidate strategy could govern an inner learning process that produces a frozen Skill, while performance on separate meta-development tasks, together with learning cost and failure records, informs subsequent strategy revisions. Such a setting would require matched resource budgets and separate meta-development and held-out evaluation tasks, extending adaptation from the reusable Skill to the procedure that learns it.

C.7 Frozen-Skill Assessment Protocol

Each stage's Skill is frozen before assessment. The learned-Skill Agent and an independently rerun Base use the same model configuration, public task interface, execution environment, and evaluator; Base receives no learned Skill. Evaluation agents submit their workflow DAGs for external execution and scoring and receive no Judge feedback while solving the evaluation tasks. Evaluation outcomes are retained separately and are not used as continuation inputs.

Workflow scores assess the scientific outcomes produced using a frozen Skill, while separate execution and semantic checks assess whether the retained guidance and code implement their stated procedures and conditions. Freezing the Skill therefore provides a stable object for comparing scientific workflow performance across learning configurations and refinement stages.

C.8 Single-Stage Feedback and Session Protocol

For each training task, the responsible learner revises candidate DAGs using Judge scores and summaries and completes three successful evaluations within the stage. Here, success means that a submission executes and receives a valid scientific score; it need not improve on an earlier candidate. Failed submissions are recorded separately. The learner distills reusable procedures and supporting code into a Skill package. Completion of the feedback quota determines the amount of training interaction but does not itself establish the correctness of the resulting Skill.

Sequential learning. SEQUENTIAL uses one continuous agent session in which training tasks are presented and processed in a prescribed order (Figure 2a). For each task, the learner completes three successful Judge-feedback rounds before proceeding to the next. The conversation state is retained

across tasks, carrying candidate choices, feedback, and observations forward so that earlier experience can influence later exploration. After completing the full sequence, the same learner distills a shared Skill from the accumulated experience. Thus, cross-task reuse occurs through the continuing learner state, while the final Skill serves as the persistent artifact for subsequent evaluation. The four- and eight-task configurations use the corresponding ordered case sets from Appendix A.7.

Task-Parallel learning. TASK-PARALLEL (TP) creates an independent learner session for each training task (Figure 2b). Each learner receives its assigned task, completes three successful evaluations, and produces a local Skill. Once all learners complete, a fresh synthesizer combines their outputs into one shared Skill. The four- and eight-task configurations use four or eight independent learners followed by one synthesizer.

Scientific evaluation occurs only in the learner sessions; the synthesizer has no Judge access. Evaluation agents later receive only the shared Skill. Appendix C.2 details the synthesis inputs and audit report.

D Discussion and Limitations

What do the results suggest? AtmoSkillRSI connects Skill formation, variation in learned-Skill performance, and recursive Skill refinement within a shared atmospheric workflow setting. On RPF, both Sequential and Task-Parallel learning produce Skills that improve workflow performance over Base, showing that useful reusable procedures can emerge under different organizations of exploration experience. Across atmospheric task families, however, learned-Skill performance varies, indicating that procedures learned under one scientific setting are not uniformly effective across tasks. Increasing training-case coverage does not consistently improve workflow performance, while the trace-enabled TP configuration performs better in the available synthesis comparison. These observations suggest that downstream performance depends not only on the amount of training experience, but also on how that experience is organized and what evidence is retained during synthesis. Finally, mean workflow performance improves across successive TP refinement stages, supporting the potential of persistent Skill reuse for cumulative refinement. Taken together, the results suggest that forming reusable Skills and refining them through subsequent interaction are related but distinct aspects of experience-driven adaptation.

Scope of the current evidence. The current study prioritizes controlled comparisons over broad empirical coverage, and the reported results should therefore be interpreted as empirical trends under the studied configurations rather than precise estimates of their expected performance. Additional repeated learning trajectories and broader evaluation would be needed to characterize variability more reliably. In recursive refinement, later stages receive the preceding Skill together with additional exploration and accumulated training history; the I1–I3 trend therefore reflects the full refinement process rather than the isolated contribution of prior-Skill reuse. Likewise, the TP versus TP_NI comparison reflects the available end-to-end configurations rather than a matched intervention on interaction traces alone. Matched ablations and additional repeated trials would be required to separate these effects more precisely.

From Skill refinement to learning-strategy evolution. Appendix C.6 discusses extending the fixed learning rule to an outer strategy-evolution loop.