A Scientific Human-Agent Reproduction Pipeline

Joschka Birk
Universität Hamburg
joschka.birk@uni-hamburg.de

Gregor Kasieczka
Universität Hamburg
gregor.kasieczka@uni-hamburg.de

Siddharth Mishra-Sharma*
Boston University
smishras@bu.edu

Benjamin Nachman
Stanford University
SLAC National Accelerator Laboratory
nachman@stanford.edu

Dennis Noll
Stanford University
SLAC National Accelerator Laboratory
nollde@stanford.edu

Tanvi Wamorkar
Stanford University
SLAC National Accelerator Laboratory
twamorka@stanford.edu

Abstract

Reproducing scientific analyses is essential for preserving knowledge, building extensible codebases, and deepening researcher understanding – yet the effort often outweighs its academic recognition. We argue that the reproduction of scientific data analyses is fundamentally a translation task: converting human-readable knowledge (papers, documentation) into machine-readable analysis code. This makes it uniquely well-suited for AI agents. We present SHARP (Scientific Human-Agent Reproduction Pipeline), a structured framework for reproducing scientific analyses through human-agent collaboration. SHARP decomposes a reproduction task into discrete steps, which an AI agent executes autonomously using specialized subagents for code generation, testing, and quality assurance. At defined checkpoints, the researcher reviews progress, provides feedback, and steers the analysis - keeping the human firmly in control of scientific judgment while the agent handles implementation. We demonstrate SHARP by reproducing a jet classification task in particle physics from a published paper. We evaluate the reproduction along three axes: analysis performance against the original results, code quality and faithfulness, and the nature of the human-agent conversation. The latter is evaluated with a novel framework for characterizing human-agent interactions. Our work highlights a practical model for AI-assisted scientific reproduction where the researcher's role shifts from writing code to understanding, evaluating, and directing – elevating human understanding rather than replacing it.

1 Introduction

Reproducing a published scientific data analysis yields two things of lasting value: an extensible codebase that preserves the analysis and can be adapted to explore related problems, and a deep researcher understanding of the processes, assumptions, and pitfalls that lie between the analysis methods and the numerical results. Despite these benefits, the effort required to reproduce an analysis often outweighs its academic recognition, and, as a consequence, published results are rarely

*Also at Anthropic

Conference on Physics and AI at Stanford University (PAI 2026).

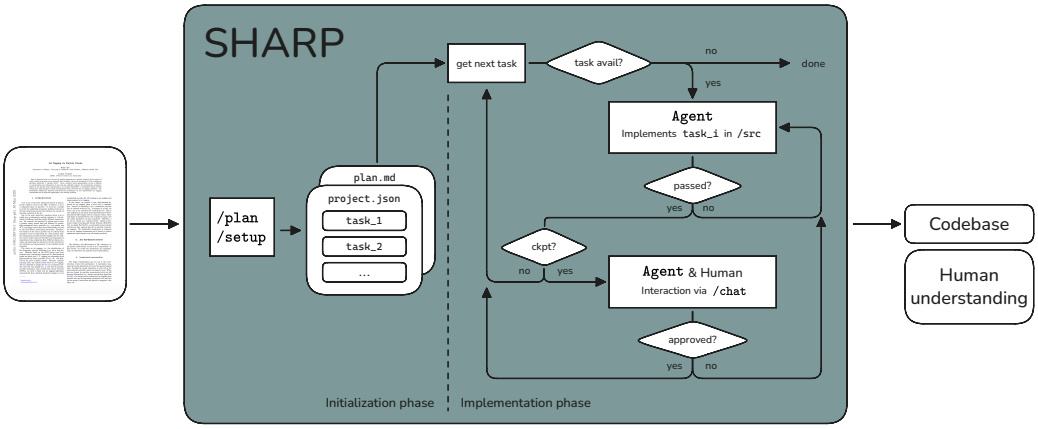

Figure 1: Starting from an initial input – including the paper to be reproduced – SHARP first produces a reproduction plan, then iteratively solves the individual tasks of the plan, with the agent working autonomously and the human intervening at checkpoints. The final output is a reproduced codebase and a deepened human understanding of the analysis.

independently reproduced. Modern AI coding agents offer a path to make this effort substantially easier.

The reproduction of data analyses is particularly well suited to AI agents because it is a *translation* task, not an *extrapolation* task. A paper describing such an analysis is ideally a self-sufficient specification of the process. Reproduction then amounts to converting this human-readable specification into machine-readable code. This framing casts the agent as a meticulous translator rather than a creative problem-solver. With the agent handling the translation, reproduction becomes faster and easier, and the researcher can devote their time to scientific judgment – reading, testing, and understanding the resulting analysis. An agent-driven workflow can also enforce consistent code standards and quality across analysis reproductions, without requiring different researchers to internalize best practices from scratch.

Recent work explores a variety of different approaches to agents in scientific research [1–11]. We present Scientific Human-Agent Reproduction Pipeline (SHARP), a specialized agent harness and collaborative workflow for the guided reproduction of scientific analyses, jointly run by a human and an agent. During the reproduction process, the agent, which is built on Claude Code, has access to the paper, its supporting documents, and selected online resources. The agent and the human first agree on a reproduction plan; the agent then works on subtasks autonomously, interacting with the human at predefined checkpoints. With this structure, SHARP occupies a deliberate middle ground between fixed, task-specific pipelines such as Agents of Discovery [12] or `CoLLM` [13] and more free-form agent workflows such as the Just Furnish Context [14]: structured enough to enforce consistency and quality, yet flexible enough to tackle non-trivial, real-world scientific analyses.

2 The SHARP Framework

The SHARP workflow, illustrated in Figure 1, is iterative and largely follows Geoffrey Huntley's Ralph pattern [15]. A user starts from a SHARP template [16] that provides the basic project structure, software environment, and agent setup. The workflow is built on Claude Code `v2.1.92` [17] and uses `claude-opus-4.6` as the underlying LLM.

Invoked via the `/plan` skill, the agent first studies the paper in question and asks clarifying questions to understand the task and the user's expectations. Then, it proposes a decomposition of the reproduction into a set of individual tasks (ten by default), some of which are marked as checkpoints, stored in `plan.md`. The user can refine or redirect the plan, including the checkpoints, through further discussion with the agent. Once the plan is approved, the agent compiles it into a machine-readable project specification, `project.json`, via the `/setup` skill. This project specification drives the subsequent execution phase.

In each iteration, the agent solves a single task from the plan, which reduces the needed context window and allows the process to be paused and resumed at any task boundary. State is handed off between iterations via `git` version control, a small set of project files (`project.json`, `progress.txt`) that record the current plan and the progress so far, and the state of the codebase. Within each iteration, the agent delegates to specialized subagents in parallel: a *Paper Analyst* extracts information from the paper, *Code* and *Test* subagents implement the task in a test-driven style, a *Statistician* handles statistical analysis, and a *Critic* enforces the *FlexCAST* principles of modularity, testability, and robustness [18]. Modularity is implemented through the `law` [19] workflow engine, and the generated code runs in a self-contained Conda environment set up, along with other project configuration, by a single `setup.sh` script. The agent explicitly evaluates whether the task is complete based on the tests and the project specification.

If the completed task is marked as a checkpoint, the agent pauses execution and hands control back to the user for review. To streamline this step, the agent summarizes the work it has just completed and advises the user on how to run the code, execute the relevant tests, and spot likely issues. Through the `/chat` interface, the user can discuss the solution in more detail before either approving the implementation or requesting revisions. Once the checkpoint is cleared, the agent proceeds to the next task and the process continues until all tasks are completed.

3 Experimental Setup

At the Large Hadron Collider at CERN, high-energy proton collisions produce unstable particles whose decay products can form *jets* – collimated sprays of detectable particles. *Jet tagging* is the task of classifying jets by the type of particle that initiated them, and is a central ML task in particle physics. We focus on the reproduction of the ParticleNet [20] paper, which introduces a graph neural network that treats jets as point clouds, evaluated on top quark vs. QCD jet classification using a public dataset [21, 22]. The reproduction targets four metrics for the ParticleNet-Lite model reported in [20]: accuracy, area under the ROC curve (AUC), and background rejection at 50% and 30% signal efficiency (R_{50} and R_{30}). We choose the ParticleNet-Lite architecture as it is a non-trivial architecture for jet tagging while requiring only moderate resources. To obtain the target metrics, the agent is expected to implement multiple non-trivial stages including dataset download and preparation, model training and evaluation, and uncertainty estimation. The reproduction starts with a prompt from the human (Prompt 1), specifying the paper, target metrics, and engineering requirements.

Prompt 1: Prompt used to initiate the replication of the ParticleNet-Lite results.

```
Load the plan skill and create a PRD for replication (incl reimplementation) of arXiv:1902.08570.
- Running in a container with 1 GPU. Ensure that the code runs on GPU, but also add CPU support as fallback.
- Only ParticleNet-Lite, only top tagging. Reproduce accuracy, AUC, and 1/eps_b results.
- Provide smoke-test mode (‘--n-events N‘) that runs train->evaluate->plot on a tiny subset in a short time.
- Use PyTorch + PyTorch Lightning. Create validation plots of input feature distributions vs physical expectations.
- Versioned output (‘--version i‘) creating separate results folders. Plots in PDF.
- Ensure that uncertainties on the final results are handled exactly as in the paper.
```

Both the agent and the human operate inside `claude-hpc`, a sandboxed Claude Code environment specifically developed for this project [23]. `claude-hpc` executes Claude Code in an isolated container on HPC systems – in our case NERSC's Perlmutter. Inside the container, file system access is restricted to a set of explicitly allowed directories, and network access is limited to a curated allowlist of approved services including GitHub, arXiv, and Conda. In particular, the agent has access to public code repositories associated with the target paper, and has been observed to occasionally consult them. Furthermore, the container has access to a single NVIDIA A100 GPU on which all training runs are performed.

4 Results

Table 1 shows the results from three independent SHARP runs of the reproduction task compared to the values reported in the original paper. All three runs were performed under a flat-rate $200/month Claude Max subscription, with no per-token API charges.

For each SHARP run we report the median and spread of 9 classifier trainings. In all runs, the performance agrees with the published ParticleNet-Lite result, except for slight deviations in the R_{30}

metric, which is particularly sensitive. This shows that most of the results are well reproduced with hints to minor differences to the official publication.

Table 1: Reproduced ParticleNet-Lite [20] performance compared to the original paper. We report the performance of the training with the median-accuracy. Uncertainties on the accuracy and AUC are not compared as they are not reported in the original paper. Uncertainties on the background rejection at signal efficiency equal to 50% (R_{50}) and 30% (R_{30}) are obtained from the standard deviation over all classifier trainings.

	Accuracy	**AUC**	$\boldsymbol{R_{50}}$	$\boldsymbol{R_{30}}$
ParticleNet-Lite (paper)	0.937	0.9844	325 ± 5	1262 ± 49
SHARP Run 1	0.938	0.9844	318 ± 7	1278 ± 41
SHARP Run 2	0.938	0.9844	317 ± 7	1188 ± 41
SHARP Run 3	0.938	0.9845	323 ± 6	1147 ± 50

The agent produces a well-structured `law`-based workflow, giving the human a legible entry point into the generated codebase. The generated codebase contains $\mathcal{O}(100)$ unit tests with 99% `pytest` line coverage of production code (not including `law` or plotting code). In addition, individual model checkpoints from the SHARP runs were independently validated by an external evaluation script written by a human expert, using only the plain PyTorch model definition and the corresponding weights.

To characterize the human-agent conversation, we use `claude-parser` [24], a tool specifically developed for this work. For each human message, the tool calls `claude-haiku-4-5` with the full conversation as context, and asks it to assign a *complexity* and a *type*. *Complexity* captures how much valuable information the message contributes to the reproduction – that is, how much worse off the agent would be without it: *easy* (approvals, error reports), *medium* (a concrete parameter, dataset, or correction), or *hard* (dense guidance, methodological insight, or scope-defining input). *Type* captures the role of the intervention: *essential* (required for faithfulness to the paper), *optional* (extra scope beyond the paper), or *meta* (workflow steering, e.g. "commit now").

Table 2 reports the resulting distribution of human messages across the three reproduction runs. In every run, the first human message – specifying the reproduction target and engineering requirements – is classified as a *hard essential* intervention, providing the agent with the scope and evaluation targets of the reproduction. Run 1 proceeded smoothly with only nine total human messages, while runs 2 and 3 required additional steering that is dominated by medium-complexity essential interventions (concrete corrections and clarifications) and easy meta messages for workflow actions such as commits. Each reproduction run took roughly one working day from the initial prompt to the final results, with the user operating two runs in parallel at medium intensity – attending to the agent between other unrelated tasks throughout the day.

Table 2: Distribution of human messages across the three reproduction runs, classified by *type* (*essential*, *optional*, *meta*) and *complexity* (H: hard, M: medium, E: easy). The first human message – specifying the reproduction target and engineering requirements (Prompt 1) – is hard essential by construction in every run, as it defines the scope and evaluation targets of the reproduction. Remaining messages reflect organic human-agent interaction during execution.

Run	**Essential (H/M/E)**	**Optional (H/M/E)**	**Meta (H/M/E)**
SHARP Run 1 ($n = 9$)	1 / 1 / 1	1 / 0 / 0	0 / 0 / 5
SHARP Run 2 ($n = 24$)	3 / 6 / 1	0 / 1 / 1	0 / 0 / 12
SHARP Run 3 ($n = 31$)	2 / 13 / 0	0 / 2 / 1	0 / 1 / 12

Limitations The agent occasionally introduces subtle implementation differences from the paper – such as the learning rate schedule, activation functions, or aggregation choice – which typically have only minor impact on the final metrics and are usually caught by repeated cross-checking prompts. A more severe failure mode concerns dataset-specific pitfalls that require domain knowledge: the public top tagging dataset, for instance, stores a truth-label particle that trivially invalidates the classification task if inadvertently loaded. This failure did not occur in our reported runs but was

encountered during development and went undetected by automated tests. The three reported runs comprise all runs performed with the published setup; no run was abandoned or failed to complete with human-in-the-loop steering. The degree of human input required in each run is reflected in the conversation analysis (Table 2). Finally, our evaluation rests on a single reproduction; how well SHARP generalizes across analyses and domains is left for future work.

5 Discussion and Conclusion

This work has introduced SHARP, a structured framework for reproducing scientific analyses through human-agent collaboration, and demonstrated its effectiveness via the reproduction of a non-trivial result from particle physics: a graph-based neural network for jet classification on point clouds. We report on the resulting performance, code quality, and human-agent interaction.

The reproduced results match the original paper with high precision (within 0.1 percentage points in accuracy). Human-agent interaction varied across runs, from highly efficient exchanges to runs requiring more substantive methodological clarification. SHARP handles codebase preparation and workflow setup with high fidelity. Correctly reproducing finer implementation details requires targeted prompting and iterative cross-checking, and remains an important limitation for scientific contexts. A key latent risk is domain-specific failure modes that automated tests cannot catch, such as truth-label leakage, underscoring that human oversight remains essential.

We see analysis reproduction as a high-value target for agentic AI. Benchmarks are abundant, and reproduction both deepens researcher understanding and provides a foundation for downstream innovation. Promising directions for future work include improved agent self-consistency and a more structured test design.

Code Availability

The SHARP template is available at: `https://github.com/stanford-ai4physics/sharp`. The `claude-parser` tool for the conversation analysis is available at: `https://github.com/nollde/claude-parser`. The `claude-hpc` containment tool is available at: `https://github.com/nollde/claude-hpc`.

Acknowledgments

This project originated at the "Human Meets AI in Scientific Research Replication" hackathon at the Center for Decoding the Universe at Stanford University – we would like to thank the organizers, especially Ioana Ciucă. B.N., D.N., and T.W. are supported by the Department of Energy (DOE), Office of Science under contract DE-AC02-76SF00515. T.W. is supported by the National Science Foundation under Grant No. 2311666. This research used resources of the National Energy Research Scientific Computing Center, a DOE Office of Science User Facility supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231 using NERSC award HEP-ERCAP0035546. J.B. and G.K. are supported by the DFG under the German Excellence Initiative – EXC 2121 Quantum Universe – 390833306. J.B. is supported by a scholarship of the German Academic Exchange Service (DAAD). J.B. also acknowledges support via the Hamburg VISTA/VISOR – Virtual Initiative for Science & Technology in AI – network.

References

[1] Matthew D. Schwartz. Resummation of the C-Parameter Sudakov Shoulder Using Effective Field Theory, 2026. URL https://arxiv.org/abs/2601.02484.

[2] David Shih. Learning to Unscramble: Simplifying Symbolic Expressions via Self-Supervised Oracle Trajectories, 2026. URL https://arxiv.org/abs/2603.11164.

[3] David Shih. Learning to Unscramble Feynman Loop Integrals with SAILIR, 2026. URL https://arxiv.org/abs/2604.05034.

[4] Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The AI Scientist: Towards Fully Automated Open-Ended Scientific Discovery, 2024. URL https://arxiv.org/abs/2408.06292.

[5] Eli Gendreau-Distler, Joshua Ho, Dongwon Kim, Luc Tomas Le Pottier, Haichen Wang, and Chengxi Yang. Automating High Energy Physics Data Analysis with LLM-Powered Agents, 2025. URL https://arxiv.org/abs/2512.07785.

[6] Andrej Karpathy. karpathy/autoresearch, April 2026. URL https://github.com/karpathy/autoresearch. original-date: 2026-03-06T22:00:43Z.

[7] Anthony Badea, Yi Chen, Marcello Maggi, Yen-Jie Lee, and Electron-Positron Alliance. Agentic AI – Physicist Collaboration in Experimental Particle Physics: A Proof-of-Concept Measurement with LEP Open Data, 2026. URL https://arxiv.org/abs/2603.05735.

[8] Shi Qiu et al. PRBench: End-to-end Paper Reproduction in Physics Research, 2026. URL https://arxiv.org/abs/2603.27646.

[9] Tilman Plehn, Daniel Schiller, and Nikita Schmal. MadAgents, 2026. URL https://arxiv.org/abs/2601.21015.

[10] Shi Qiu, Zeyu Cai, Jiashen Wei, Zeyu Li, Yixuan Yin, Qing-Hong Cao, Chang Liu, Ming-xing Luo, Xing-Bo Yuan, and Hua Xing Zhu. An End-to-end Architecture for Collider Physics and Beyond, 2026. URL https://arxiv.org/abs/2603.14553.

[11] Qibin Liu and Julia Gonski. Scifi: A safe, lightweight, user-friendly, and fully autonomous agentic ai workflow for scientific applications, 2026. URL https://arxiv.org/abs/2604.13180.

[12] Sascha Diefenbacher, Anna Hallin, Gregor Kasieczka, Michael Krämer, Anne Lauscher, and Tim Lukas. Agents of Discovery, 2026. URL https://arxiv.org/abs/2509.08535.

[13] W. Esmail, A. Hammad, and M. Nojiri. CoLLM: AI engineering toolbox for end-to-end deep learning in collider analyses, 2026. URL https://arxiv.org/abs/2602.06496.

[14] Eric A. Moreno, Samuel Bright-Thonney, Andrzej Novak, Dolores Garcia, and Philip Harris. AI Agents Can Already Autonomously Perform Experimental High Energy Physics, 2026. URL https://arxiv.org/abs/2603.20179.

[15] Geoffrey Huntley. Ralph Wiggum as a "software engineer", July 2025. URL https://ghuntley.com/ralph/.

[16] Dennis Noll and Joschka Birk. SHARP: Template to reproduce scientific analyses with a coding agent., April 2026. URL https://github.com/stanford-ai4physics/sharp.

[17] Anthropic. Claude Code Docs, 2026. URL https://code.claude.com/docs/en/overview.

[18] Benjamin Nachman and Dennis Noll. FlexCAST: Enabling Flexible Scientific Data Analyses, 2025. URL https://arxiv.org/abs/2507.11528.

[19] Marcel Rieger. End-to-End Analysis Automation over Distributed Resources with Luigi Analysis Workflows. *EPJ Web Conf.*, 295:05012, 2024. doi: 10.1051/epjconf/202429505012.

[20] Huilin Qu and Loukas Gouskos. ParticleNet: Jet Tagging via Particle Clouds. *Phys. Rev. D*, 101 (5):056019, 2020. doi: 10.1103/PhysRevD.101.056019.

[21] Gregor Kasieczka, Tilman Plehn, Jennifer Thompson, and Michael Russel. Top Quark Tagging Reference Dataset, March 2019. URL `https://doi.org/10.5281/zenodo.2603256`.

[22] Anja Butter et al. The Machine Learning landscape of top taggers. *SciPost Phys.*, 7:014, 2019. doi: 10.21468/SciPostPhys.7.1.014.

[23] Dennis Noll and Joschka Birk. claude-hpc: Sandboxed Claude Code environment for HPC and local Docker, April 2026. URL `https://github.com/nollde/claude-hpc`.

[24] Dennis Noll. claude-parser: Display and Analyze Conversations with Claude, April 2026. URL `https://github.com/nollde/claude-parser`.