CO-EVOLVING SKILLS, MEMORY, AND POLICY FOR PERSONALIZED RESEARCH AUTOMATION

Anonymous authors
Paper under double-blind review

ABSTRACT

LLM-powered multi-agent systems can now automate the full research pipeline from ideation to paper writing, but a fundamental question remains: automation for whom? Researchers operate under different resource configurations, hold different methodological preferences, and target different output formats. A system that produces uniform outputs regardless of these differences will systematically under-serve every individual user, making personalization a precondition for research automation to be genuinely usable. However, achieving it requires three capabilities that current systems lack: accumulating reusable procedural knowledge across projects, retaining user-specific experience across sessions, and internalizing implicit preferences that resist explicit formalization. We propose NanoResearch, a multi-agent framework that addresses these gaps through tri-level co-evolution.A skill bank distills recurring operations into compact procedural rules reusable across projects. A memory module maintains user- and project-specific experience that grounds planning decisions in each user's research history. A label-free policy learning converts free-form feedback into persistent parameter updates of the planner, reshaping subsequent coordination. These three layers co-evolve: reliable skills produce richer memory, richer memory informs better planning, and preference internalization continuously realigns the loop to each user. Extensive experiments demonstrate that NanoResearch delivers substantial gains over state-of-the-art AI research systems, and progressively refines itself to produce better research at lower cost over successive cycles.

1 INTRODUCTION

LLM-powered multi-agent systems Achiam et al. (2023) have recently transformed end-to-end research automation from a long-standing aspiration Langley (1987); Waltz & Buchanan (2009) into working reality. Systems such as The AI Scientist Lu et al. (2024), AI Scientist-v2 Yamada et al. (2025), EvoScientist Lyu et al. (2026), and AI-Researcher Tang et al. (2025) can now autonomously traverse the full research lifecycle Weng et al. (2025); Shao et al. (2025); Li et al. (2026), surveying literature, generating hypotheses, implementing experiments, and writing papers within a single pipeline. These advances mark genuine progress: tasks that once required weeks of researcher effort can now be completed in hours at modest cost Zhu et al. (2025). Yet the ability to complete the pipeline does not guarantee that its outputs are usable by any particular researcher.

Research is fundamentally shaped by the context in which it is conducted Kuhn & Hacking (1970); Latour et al. (2013). Communities diverge in what constitutes a valuable contribution: AI-for-science researchers prioritize whether a method addresses a meaningful real-world need Moor et al. (2023); Wornow et al. (2023), while core computer vision researchers value architectural novelty and consistent benchmark gains Lipton & Steinhardt (2019). Beyond research philosophy, teams also differ in resource budgets Schwartz et al. (2020), methodological preferences, and target venues. A system that produces the same research plan regardless of these differences cannot reliably serve either community. **Personalization is therefore a precondition for research automation to be genuinely usable**.

Despite this need, existing systems remain fundamentally one-size-fits-all, funneling diverse researchers through a uniform pipeline that produces near-identical outputs regardless of individual context, as shown in Figure 1(a). We identify three capability gaps that jointly prevent personalization:

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

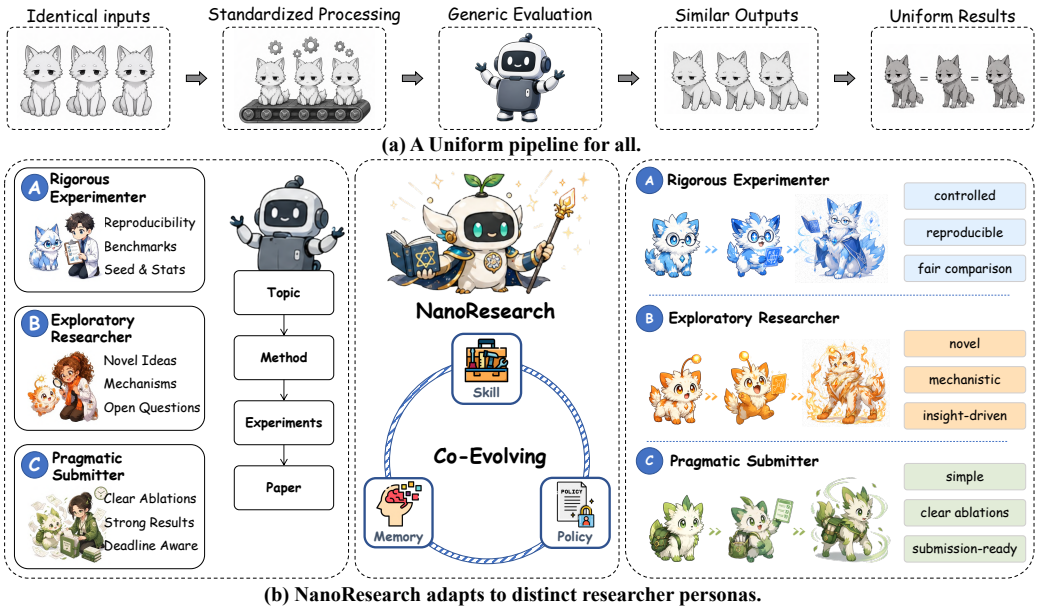


Figure 1: Comparison between (a) a uniform research automation pipeline that applies identical processing to all users and yields homogeneous outputs, and (b) NanoResearch, which recognizes distinct researcher personas and provides personalized skills and feedback upon failure, enabling each persona to evolve along its own trajectory.

(i) current systems lack reusable procedural knowledge. Each run starts from scratch, re-encountering the same debugging patterns and re-deriving the same configurations without abstracting them into compact, retrievable rules. Even memory-equipped systems such as EvoScientist Lyu et al. (2026) store episode-level narratives rather than distilled procedural primitives, limiting transferability across tasks. *(ii) current systems do not accumulate user-specific experience across sessions*. Past hypotheses, validated configurations, and inferred resource constraints are discarded once a session ends, forcing rediscovery on every subsequent run and grounding planning in generic priors rather than the user's actual research history. *(iii) current systems cannot internalize implicit preferences*. Feedback such as preferring simpler methods or wanting more efficiency analysis is too diffuse to encode as rules and too nuanced to survive compression into memory entries. Without a mechanism that converts such signals into persistent parameter-level changes, preferences fade as soon as the context window shifts.

We propose NanoResearch, a multi-agent framework that addresses these gaps through tri-level co-evolution (Figure 1(b)). A skill bank distills recurring operations into compact procedural rules reusable across projects, so that hard-won execution knowledge survives between runs. A memory module maintains user-bound and project-bound records that ground every planning decision in the user's actual research history rather than generic priors. A label-free policy learning mechanism converts free-form feedback into persistent parameter updates of the planner, allowing implicit preferences to reshape coordination behavior across subsequent decisions. Our ablations show that no isolated or reduced configuration matches the full system. Procedural knowledge supplies reusable execution strategies, contextual memory grounds them in user and project history, and preference alignment converts evolving intent into planner behavior. Together they form a co-evolutionary loop in which skill execution populates memory, accumulated memory strengthens planning, and preference learning realigns the system toward each user.

Extensive experiments across 20 research topics spanning seven domains demonstrate that NanoResearch consistently outperforms existing systems under simulated researcher evaluations, and human evaluations by PhD researchers on their own research tasks confirm that these gains transfer to real domain experts. NanoResearch produces higher-quality research outputs while achieving stronger preference alignment, and its performance improves progressively over successive research cycles. These results suggest that personalization is not merely a desirable add-on but a fundamental axis along which autonomous research systems must evolve, and that tri-level co-evolution offers a viable path toward systems that grow more effective the longer they collaborate with a given researcher.

2 Related Work

End-to-end research automation. An emerging line of work targets end-to-end scientific automation spanning the full research lifecycle from ideation to paper writing (Lu et al., 2024; Yamada et al., 2025; Tang et al., 2025; Lyu et al., 2026; Weng et al., 2025; Yang et al., 2023; Xie et al., 2025). As a pioneering effort, The AI Scientist (Lu et al., 2024) realizes the first such fully automated pipeline, culminating in an LLM-based reviewing process, and its successor, AI Scientist-v2 (Yamada et al., 2025), further incorporates agentic tree search to better explore research decisions. Other concurrent efforts (Tang et al., 2025; Lyu et al., 2026; Weng et al., 2025; Yang et al., 2023; Xie et al., 2025) instead adopt multi-agent architectures that orchestrate specialized agents to collaboratively drive the research process: EvoScientist (Lyu et al., 2026) equips agents with persistent memory and self-evolution to distill and reuse strategies from past trajectories; DeepScientist (Weng et al., 2025) formulates discovery as goal-driven Bayesian optimization for long-horizon exploration; and AI-Researcher (Tang et al., 2025) decomposes concepts into atomic units linking formulations to code, which are refined through mentor-guided agent loops. However, most existing systems still operate as static pipelines (Lu et al., 2024; Yamada et al., 2025; Tang et al., 2025), while the few attempts at dynamic adaptation (Lyu et al., 2026) remain limited to passive memory logging, making it difficult to efficiently accumulate experience or accommodate individual user needs. In contrast, our work achieves multi-level self-evolution across skills, memory, and planner policy, while leveraging user profiles and feedback to deliver personalized outputs.

Task-specific research automation. Early efforts on AI scientists primarily aimed to assist human researchers in specific subtasks rather than replacing them. Even before the LLM era Touvron et al. (2023); Bai et al. (2023); Guo et al. (2025); Jiang et al. (2024), prior work had explored using AI to support scientific research Lee et al. (2020); Cachola et al. (2020); Huang et al. (2019); Beltagy et al. (2019); Clune (2019), and recent studies further leverage foundation models Team et al. (2025b); Bai et al. (2025) to enhance assistance at individual research stages Shao et al. (2025); Team et al. (2025a). Some efforts focus on literature understanding, like PaperQA Lála et al. (2023), which answers scientific questions by retrieving and reasoning over relevant papers. Another line targets novel idea generation, with Nova Hu et al. (2024) retrieving external knowledge to enhance novelty and ResearchAgent Baek et al. (2025) augmenting LLMs with an entity-centric knowledge store and iterative reviewing agents. Moving from ideation to reproduction, AutoP2C Lin et al. (2025) converts papers into code via a multi-agent pipeline, while ResearchCodeAgent Gandhi et al. (2025) iteratively refines an initial codebase with dynamic planning.

3 Method

3.1 Overview

Unlike existing automated research systems Lyu et al. (2026); Tang et al. (2025) that follow rigid workflows, we propose **NanoResearch**, a self-evolving framework that turns a user-specified topic $\mathcal{T}$ into a complete academic paper $\mathcal{P}$. To tailor the pipeline to each researcher, the system first constructs a user profile $\mathcal{U}$ via interactive queries, serving as persistent context for all subsequent decisions. As illustrated in Figure 2, the workflow comprises three stages: (1) **Idea Generation and Planning**, (2) **Experimental Validation and Optimization**, and (3) **Paper Writing and Review**, supported by a **Skill Bank** $\mathcal{S}$ and a **Memory Module** $\mathcal{M}$, coordinated by an **Orchestrator** $\mathcal{O}$ that retrieves relevant entries before each task and updates both stores afterward. Users provide natural-language feedback $\mathcal{F}$ at the end of each stage, which $\mathcal{O}$ internalizes into its planner policy, turning explicit feedback into persistent preferences.

3.2 NanoResearch Pipeline: A Self-Evolving Research System

3.2.1 Stage I: Idea Generation and Planning

The initial stage transforms a user-specified research topic $\mathcal{T}$ into a novel, executable experiment blueprint $\mathcal{B}$, constrained by the user profile $\mathcal{U}$, through two sequential phases: *Ideation* and *Planning*.

Ideation phase begins by systematically surveying the existing literature. The Orchestrator $\mathcal{O}$ first retrieves topic- and user-aligned skills $\mathcal{S}_I \subseteq \mathcal{S}$ and memories $\mathcal{M}_I \subseteq \mathcal{M}$, and produces a high-level



Figure 2: **The NanoResearch framework.** An Orchestrator $\mathcal{O}$ processes a personalized research request and coordinates a three-stage pipeline (ideation, experimentation, writing) to produce a complete research manuscript for subsequent human review. A Skill Bank $\mathcal{S}$, a Memory Module $\mathcal{M}$, and policy learning jointly accumulate experience and drive self-evolution across cycles.

plan P_I outlining the survey scope and hypothesis generation strategy:

$$\mathcal{S}_I, \mathcal{M}_I = \text{Retrieve}(\mathcal{S}, \mathcal{M} \mid \mathcal{T}, \mathcal{U}), \quad P_I = \text{Plan}(\mathcal{T}, \mathcal{U} \mid \mathcal{S}_I, \mathcal{M}_I). \quad \triangleright\ \mathcal{O} \tag{1}$$

Guided by P_I, the system queries academic databases (e.g., arXiv, Semantic Scholar) to retrieve relevant papers L, and applies a *quantitative evidence extraction* mechanism that parses performance scores directly from the papers to yield grounded evidence E and mitigate hallucination. A ReAct-based reasoning loop then identifies research gaps and proposes candidate hypotheses $H = \{h_1, \ldots, h_K\}$, after which an automated *novelty verification* step queries the databases with each h_k to filter out prior-work overlaps, yielding the most promising hypothesis $h^* \in H$.

Planning phase translates h^* into a rigorous, JSON-formatted experiment blueprint $\mathcal{B}$. The Orchestrator is invoked again to retrieve execution-level context and produce a high-level plan P_P:

$$\mathcal{S}_P, \mathcal{M}_P = \text{Retrieve}(\mathcal{S}, \mathcal{M} \mid h^*, \mathcal{U}), \quad P_P = \text{Plan}(h^*, \mathcal{U} \mid \mathcal{S}_P, \mathcal{M}_P). \quad \triangleright\ \mathcal{O} \tag{2}$$

Guided by P_P, $\mathcal{B}$ is instantiated with concrete specifications including datasets, baselines, proposed architecture, evaluation metrics, and ablation studies, and then undergoes an automated *peer-review-like correction loop*: an internal LLM reviewer critiques $\mathcal{B}$ for infeasible designs or unfair comparisons, producing a critique $c_{\mathcal{B}}$ that drives iterative refinement:

$$\mathcal{B}^{(t+1)} = \text{Refine}(\mathcal{B}^{(t)}, c_{\mathcal{B}}^{(t)}, P_P, E), \tag{3}$$

until $\mathcal{B}$ passes review or reaches the retry limit. Finally, the Orchestrator distills new reusable skills and memories from the trajectory:

$$\mathcal{S}, \mathcal{M} \leftarrow \text{Update}(\mathcal{S}, \mathcal{M} \mid h^*, \mathcal{B}, c_{\mathcal{B}}). \quad \triangleright\ \mathcal{O} \tag{4}$$

3.2.2 Stage II: Experimental Validation and Optimization

Following the formulation of $\mathcal{B}$, this stage transitions from conceptual design to empirical validation.

Setup and Coding phase first prepares the environment by cloning suitable base repositories and staging the datasets specified in $\mathcal{B}$. To align the generated code with $\mathcal{U}$, the Orchestrator $\mathcal{O}$ retrieves coding-specific skills $\mathcal{S}_C \subseteq \mathcal{S}$ and project memories $\mathcal{M}_C \subseteq \mathcal{M}$, and produces a coding plan P_C:

$$\mathcal{S}_C, \mathcal{M}_C = \text{Retrieve}(\mathcal{S}, \mathcal{M} \mid \mathcal{B}, \mathcal{U}), \quad P_C = \text{Plan}(\mathcal{B}, \mathcal{U} \mid \mathcal{S}_C, \mathcal{M}_C). \quad \triangleright\ \mathcal{O} \tag{5}$$

Guided by P_C, the Coding agent instantiates a self-contained codebase $\mathcal{W}$ comprising model definitions, training scripts, evaluation pipelines, and cluster submission scripts.

Execution and Automated Debugging phase deploys $\mathcal{W}$ to the target environment (e.g., a SLURM cluster). Since initial code rarely runs zero-shot, an autonomous debugging loop iteratively patches the codebase using $\mathcal{S}_C$ and $\mathcal{M}_C$ until execution succeeds or the retry budget is exhausted:

$$\mathcal{W}^{(t+1)} = \text{Debug}(\mathcal{W}^{(t)} \mid \mathcal{S}_C, \mathcal{M}_C). \tag{6}$$

Analysis phase. Upon successful execution, raw output logs R_{raw} are parsed into an analysis report $\mathcal{A}$ covering experimental results, performance comparisons, and key findings:

$$\mathcal{A} = \text{Analyze}(R_{\text{raw}}, \mathcal{B}, \mathcal{T}). \tag{7}$$

Finally, the Orchestrator consolidates reusable skills and memories: the experimental record, whether successful or failed, is stored in $\mathcal{M}$ with its conditions, while generalizable solutions from coding and execution are abstracted into new skills in $\mathcal{S}$:

$$\mathcal{S}, \mathcal{M} \leftarrow \text{Update}(\mathcal{S}, \mathcal{M} \mid \mathcal{W}, \mathcal{A}). \quad \triangleright\ \mathcal{O} \tag{8}$$

3.2.3 Stage III: Paper Writing and Review

The final stage integrates prior outputs into a complete LaTeX manuscript for subsequent human review.

Writing phase. To maintain narrative consistency and adhere to venue-specific conventions in $\mathcal{U}$, the Orchestrator $\mathcal{O}$ retrieves writing-specific skills $\mathcal{S}_W \subseteq \mathcal{S}$ and project memories $\mathcal{M}_W \subseteq \mathcal{M}$, and formulates a structured writing plan P_W:

$$\mathcal{S}_W, \mathcal{M}_W = \text{Retrieve}(\mathcal{S}, \mathcal{M} \mid \mathcal{B}, \mathcal{A}, \mathcal{U}), \quad P_W = \text{Plan}(\mathcal{B}, \mathcal{A}, \mathcal{U} \mid \mathcal{S}_W, \mathcal{M}_W). \quad \triangleright\ \mathcal{O} \tag{9}$$

Following P_W, the Writing agent drafts the manuscript *section-by-section* to alleviate context limitations and avoid catastrophic forgetting.

Review phase. To ensure an unbiased evaluation, the Review agent operates without the skill or memory retrieval used in earlier stages. Acting as a strict external reviewer, it critiques the draft on logical coherence, claim validity, and formatting correctness, producing targeted feedback f_R:

$$\text{Draft}^{(t+1)} = \text{Revise}(\text{Draft}^{(t)}, f_R^{(t)}), \tag{10}$$

which repeats until predefined quality thresholds are met, yielding the final paper $\mathcal{P}$. The Orchestrator then distills reusable knowledge, e.g., writing techniques and revision strategies, into $\mathcal{S}$ and $\mathcal{M}$:

$$\mathcal{S}, \mathcal{M} \leftarrow \text{Update}(\mathcal{S}, \mathcal{M} \mid \mathcal{P}, f_R). \quad \triangleright\ \mathcal{O} \tag{11}$$

3.3 Foundations of Self-Evolution: Memory, Skills, and Planning

3.3.1 Memory and Skill Management

The Orchestrator $\mathcal{O}$ drives the continuous evolution through the Skill Bank $\mathcal{S}$ and the Memory Module $\mathcal{M}$, relying on two core mechanisms: context-aware retrieval and trajectory-based updating.

Retrieval Mechanism. Before each task, $\mathcal{O}$ retrieves the top-k skills $\mathcal{S}_C$ and memories $\mathcal{M}_C$ relevant to the current context C (e.g., $\mathcal{T}$, $\mathcal{U}$, $\mathcal{B}$) via a heuristic scoring function:

$$\mathcal{S}_C = \underset{s \in \mathcal{S}}{\text{top-}k}\ \text{score}(C, s), \quad \mathcal{M}_C = \underset{m \in \mathcal{M}}{\text{top-}k}\ \text{score}(C, m). \tag{12}$$

The score combines keyword matching, tag alignment, and recency, with weights adapted to the target: skill retrieval prioritizes usage frequency and confidence to surface robust strategies (e.g., debugging patterns), while memory retrieval enforces strict condition matching to return only project-specific experiences (e.g., prior outcomes) from comparable settings.

Update Mechanism. Upon completing a stage, $\mathcal{O}$ reflects over the trajectory τ (actions, critiques, outcomes), distilling generalizable rules (e.g., debugging strategies) into the Skill Bank and project-specific experiences (e.g., failed hypotheses) into the Memory Module:

$$\mathcal{S}^{(t+1)} = \mathcal{S}^{(t)} \cup \text{Distill}_{\text{skill}}(\tau), \quad \mathcal{M}^{(t+1)} = \mathcal{M}^{(t)} \cup \text{Summarize}_{\text{mem}}(\tau). \tag{13}$$

To prevent unbounded growth, $\mathcal{O}$ further merges semantically overlapping entries, keeping both stores compact for future cycles.

3.3.2 Adaptive Planning

While $\mathcal{S}$ and $\mathcal{M}$ capture broad procedural knowledge and project facts, we further internalize fine-grained, user-specific preferences (e.g., coding style, analytical focus). At the end of each stage, the user provides immediate natural-language feedback $\mathcal{F}$, which we encode directly into the Orchestrator's **planner model** π_θ rather than $\mathcal{S}$ or $\mathcal{M}$, where it risks being compressed or missed at retrieval. Since $\mathcal{F}$ is free-form language rather than scalar rewards or preference pairs, we adopt Self-Distillation Policy Optimization (SDPO) Buening et al. (2026), which converts a single feedback instance into a dense, token-level learning signal without any reward model or preference annotation. Formally, given the Orchestrator's input x and the planner's initial trajectory $y \sim \pi_\theta(\cdot \mid x)$, it treats the feedback-conditioned model $\pi_\theta(\cdot \mid x, \mathcal{F}, y_{<t})$ as a *self-teacher* and updates the student $\pi_\theta(\cdot \mid x, y_{<t})$ to match its token distribution. Following Buening et al. (2026), the SDPO gradient is a logit-level policy gradient:

$$\nabla_\theta \mathcal{L}_{\text{SDPO}}(\theta) = -\,\mathbb{E}_{y\sim\pi_\theta(\cdot|x)}\left[\sum_{t=1}^{|y|} \mathbb{E}_{\hat{y}_t\sim\pi_\theta(\cdot|x,y_{<t})}\left[A_t^{\text{SDPO}}(\hat{y}_t)\cdot\nabla_\theta \log \pi_\theta(\hat{y}_t \mid x, y_{<t})\right]\right], \quad (14)$$

with the dense token-level advantage estimated via the self-teacher:

$$A_t^{\text{SDPO}}(\hat{y}_t) = \log \frac{\pi_\theta(\hat{y}_t \mid x, \mathcal{F}, y_{<t})}{\pi_\theta(\hat{y}_t \mid x, y_{<t})}. \quad (15)$$

Applied after each feedback round, this update progressively internalizes user preferences into the planner's parameters, enabling NanoResearch to satisfy user preferences over successive cycles.

4 Experiments

4.1 Experiment Setup

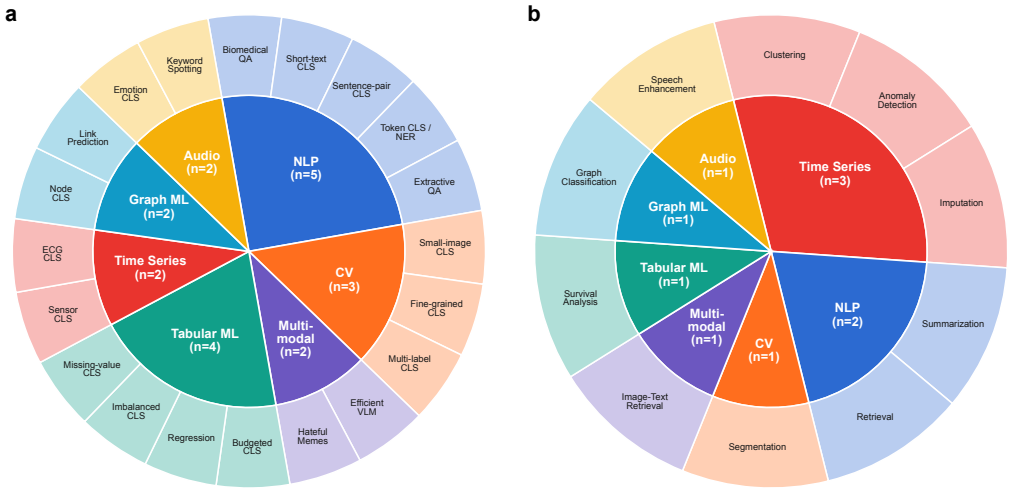


Figure 3: Task distributions of the two evaluation settings: (a) the 20 tasks of the simulated benchmark across seven domains, and (b) the ten real tasks of the human study contributed by PhD researchers.

To comprehensively evaluate NanoResearch, we build a benchmark of 20 research tasks spanning seven domains (NLP, CV, Multimodal, Tabular ML, Time Series, Graph ML, and Audio). For each task, we construct an LLM-simulated scientist with its own preferences and constraints, who provides feedback throughout the pipeline and enables personalized, multi-round evaluation. To assess self-evolution, we run NanoResearch for multiple rounds on each task and compare outputs across successive iterations. We further conduct a human study in which PhD researchers apply NanoResearch to their own ongoing research projects, provide stage-level feedback, and evaluate the resulting outputs across successive rounds. Benchmark construction details are provided in the Supplementary Material, and the task composition is shown in Figure 3.

Baselines and control protocol. We compare NanoResearch with four end-to-end research systems: AI-Researcher (Tang et al., 2025), DeepScientist (Weng et al., 2025), EvoScientist (Lyu et al., 2026), and AI Scientist-v2 (Yamada et al., 2025). All systems receive identical task specifications, including researcher profiles, preferences, constraints, required baselines, and success criteria. For agents serving the same role across different methods, we use the same base model while retaining each method's own prompts and workflows. NanoResearch Round 1 is compared directly with the baselines; Rounds 2 and 3 assess adaptation to feedback from the same researcher.

Metrics. We evaluate each system along five dimensions spanning the full research lifecycle: (1) *Alignment* (Align.), how well the output matches the user's topic, preferences, constraints, and success criteria; (2) *Executability* (E2E), the fraction of runs that complete the full pipeline with executable experiments and a final paper; (3) *Effectiveness* (Perf.), the objective result on each task's pre-specified, benchmark-native primary metric; (4) *Novelty* (Novel.), the originality of the proposed idea relative to prior work; and (5) *Writing* (Writ.), the quality of the final paper. Alignment, Novelty, and Writing are subjective dimensions for which we report LLM- and expert-rated scores; E2E and Perf. are computed from execution outcomes rather than an LLM quality proxy. For Perf., we convert each task's pre-specified metric to a higher-is-better score in $[0, 1]$. When a task contains multiple datasets, we first average their scores and then compute the macro-average across tasks.

4.2 Main Results

Table 1: Main results on the simulated benchmark and the PhD researcher study. E2E denotes end-to-end execution success, and Perf. denotes task-specific objective performance. In the Benchmark Evaluation block, subjective metrics are reported as LLM-judge/expert scores, with the LLM score listed first. In the PhD Researcher Study, subjective scores are provided by the participating PhD researchers.

Method	Benchmark Evaluation					PhD Researcher Study				
	E2E ↑	Perf. ↑	Align. ↑	Novel. ↑	Writ. ↑	E2E ↑	Perf. ↑	Align. ↑	Novel. ↑	Writ. ↑
AI-Researcher	0.500	0.2849	4.206/4.350	2.953/3.183	5.402/5.150	0.700	0.5497	4.700	3.567	4.800
DeepScientist	0.900	0.5634	4.504/4.667	3.934/4.083	4.806/4.917	0.900	0.7215	6.400	5.167	5.400
EvoScientist	0.750	0.5779	4.823/5.100	4.555/4.650	4.953/5.083	0.800	0.6574	6.100	4.833	5.067
AI Scientist-v2	0.500	0.6238	6.656/6.433	3.958/4.250	4.125/4.483	0.700	0.5637	5.800	4.267	4.633
NanoResearch (Round 1)	**1.000**	0.6844	8.163/8.100	4.960/5.183	5.428/5.650	**1.000**	0.8012	8.100	5.733	6.267
NanoResearch (Round 2)	**1.000**	0.7320	8.397/8.483	5.164/5.550	5.859/6.083	**1.000**	0.8338	8.700	**6.200**	**6.767**
NanoResearch (Round 3)	**1.000**	**0.7548**	**8.963/8.850**	**5.645/5.733**	**6.172/6.267**	**1.000**	**0.8564**	**9.100**	6.033	6.633

Table 1 compares NanoResearch with four automated research systems. R1 reports static comparisons, whereas R1–R3 measure longitudinal personalization after accumulated user feedback. On the simulated benchmark, NanoResearch leads all baselines on every reported dimension and is the only system to complete all 20 tasks end-to-end, compared with baseline success rates of 50%–90%. Rankings from the LLM judge and the human expert are consistent across subjective dimensions. The human study shows the same advantage: NanoResearch outperforms every static baseline on all five dimensions in R1. After per-round natural-language feedback, Alignment increases from 8.100 to 9.100 and Perf. from 0.8012 to 0.8564, E2E remains at 1.000, and Novelty and Writing peak in R2 at 6.200 and 6.767, respectively. These results show that R2 and R3 provide additional gains from accumulated user feedback.

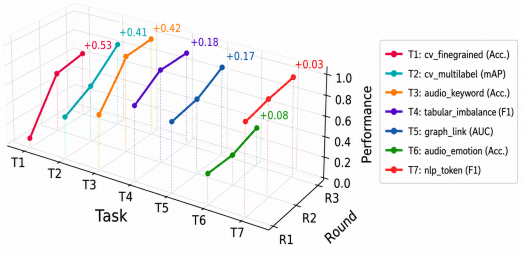


Figure 4: Per-task performance of NanoResearch across rounds R1→R3.

Under matched static conditions, NanoResearch R1 achieves Alignment scores of 8.163 (LLM) and 8.100 (expert), compared with 6.656 and 6.433 for the strongest baseline. Two rounds of user feedback raise these scores to 8.963 and 8.850, respectively. Across all subjective dimensions, the LLM judge and the expert evaluator place the methods in the same order. The shared-profile protocol gives all methods access to the same user-preference information, allowing their Alignment performance to be

compared on equal terms. Every subjective score and objective project-performance value increases from R1 to R3, while E2E remains at 1.000. Task-level performance follows the same upward trend (Figure 4). These results demonstrate sustained improvement with accumulated user feedback, while the component controls below isolate the complementary contributions of the Skill Bank, Memory Module, and SDPO.

4.3 Ablation Studies

Table 2: Ablation study results. **Bold** denotes the best result in each column.

Settings	Align. ↑	Novel. ↑	E2E ↑	Perf. ↑	Writ. ↑
Nano w/o Skill Bank	7.940	3.773	0.849	0.6480	4.75
Nano w/o Memory	8.070	4.400	0.935	0.6590	5.10
Nano w/o Planner Model	7.820	3.532	0.835	0.6420	4.70
Nano w/o Preference Alignment	8.030	4.275	1.000	0.6660	5.05
NanoResearch	**8.163**	**4.960**	**1.000**	**0.6844**	**5.42**

We ablate each core component of NanoResearch, as summarized in Table 2. Removing the Planner Model causes the largest drop across the reported metrics, while removing the Skill Bank, Memory, or Preference Alignment also lowers multiple metrics. These results demonstrate complementary contributions from all three layers.

Disabling the Skill Bank reduces E2E from 1.000 to 0.849, highlighting its role in reliable execution. Removing Memory lowers Novelty and Writing, indicating that retained project experience supports ideation and presentation. Removing Preference Alignment leaves E2E unchanged but lowers Alignment, Novelty, Perf., and Writing, showing that SDPO improves quality beyond completion. In the supplement, Planner+SDPO outperforms Planner Only without Skill or Memory retrieval, confirming gains beyond retrieval.

4.4 Efficiency Analysis

Table 3: Efficiency and cost comparison across automated research systems. All values are averaged per topic. Token counts are in millions (M). GPU cost is estimated at $2.00/hr.

Method	Avg. API Calls ↓	Avg. Tokens (M) ↓	Avg. Runtime (hrs) ↓	Avg. GPU Hours ↓	API Cost ($) ↓	GPU Cost ($) ↓	Total Cost ($) ↓
AI Scientist-v2	68.30	0.750	1.93	1.15	3.750	2.289	6.039
EvoScientist	24.67	0.428	1.35	1.02	0.914	2.030	2.944
NanoResearch (R1)	23.65	0.117	2.24	1.75	0.648	3.509	4.157
NanoResearch (R2)	18.00	0.092	1.51	1.13	0.284	2.258	2.542
NanoResearch (R3)	**15.80**	**0.073**	**1.05**	**0.60**	**0.236**	**1.194**	**1.430**

As shown in Table 3, NanoResearch uses fewer tokens than both baselines in the cost comparison. Across rounds, it reduces total cost from $4.157 per topic in R1 to $1.430 in R3, a 65.6% within-system reduction. R3 is also 76.3% less expensive than AI Scientist-v2 and 51.4% less expensive than EvoScientist under the same accounting protocol, while simultaneously attaining the strongest reported quality scores.

Table 4: Growth of the Skill Bank and Memory Module across rounds.

Round	Bank Size		Growth
	Skill /Topic	Memory /Topic	New /Topic
R1	0.80	6.40	0.80
R2	1.00	8.15	0.20
R3	**2.30**	**12.00**	**1.30**

4.5 Skill Bank and Memory Module: Growth Across Rounds

To measure the state accumulated during self-evolution, we track the Skill Bank $\mathcal{S}$ and Memory Module $\mathcal{M}$ across rounds (Table 4). From R1 to R3, the per-topic Skill Bank grows from 0.80 to

2.30 entries and Memory grows from 6.40 to 12.00 entries. This measured growth documents the Orchestrator's accumulation of reusable procedures and project-specific experience; the removal results in Table 2 establish that each store contributes to at least one quality or execution gain.

4.6 Case Study by Different Users

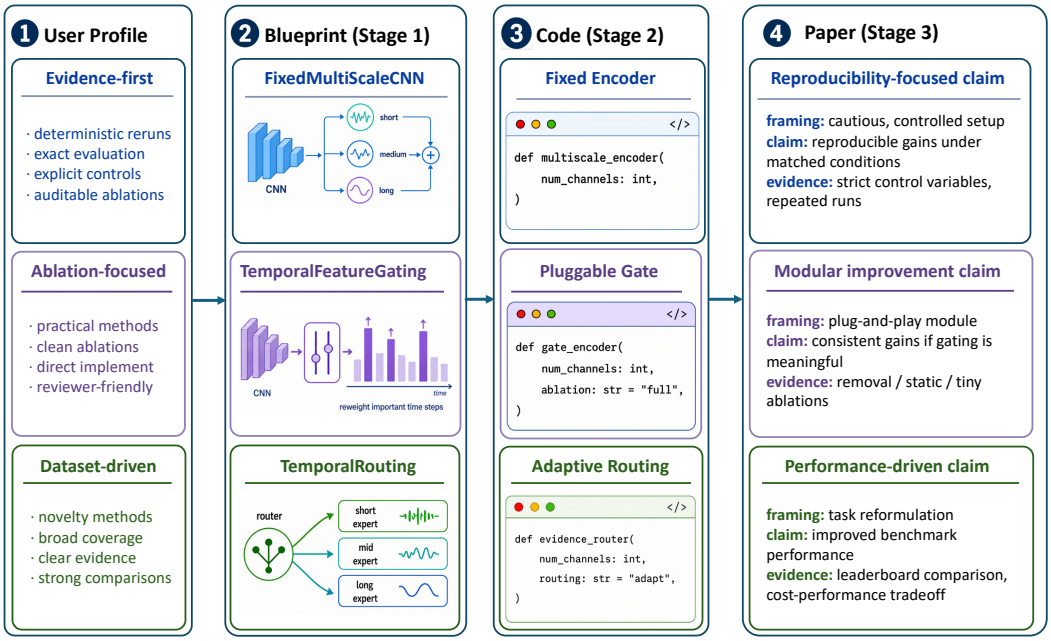


Figure 5: Case study on UCI HAR: three simulated users with *Conservative*, *Practical*, and *Exploratory* tastes lead visibly different blueprints, code, and paper claims at every stage.

To probe how user profiles reshape the research trajectory beyond metrics, we instantiate three simulated researchers with *Evidence-first*, *Ablation-focused*, and *Dataset-driven* tastes on the same UCI HAR topic. As shown in Figure 5 and detailed in the Supplementary Material (Case Study section), the blueprint, code, and paper claim diverge sharply across profiles. The *Evidence-first* user adopts a FixedMultiScaleCNN blueprint realized as a Fixed Encoder, framing the paper around reproducibility-focused claims. The *Ablation-focused* user adopts TemporalFeatureGating as a Pluggable Gate, framing the paper around modular improvement claims. The *Dataset-driven* user adopts TemporalRouting as Adaptive Routing, framing the paper around performance-driven claims. Across all three runs, the user profile $\mathcal{U}$ shapes every stage consistently, producing outputs that differ in research *taste* rather than by chance.

5 Conclusion

This work identifies personalization as a fundamental yet overlooked axis in research automation. A single, fixed pipeline cannot accommodate the diversity of researcher preferences, domain conventions, and evolving intent that characterize real scientific practice. We further show that effective personalization is not achievable through any single mechanism: it requires simultaneous adaptation at the procedural level (how tasks are executed), the contextual level (what prior experience is retained), and the preference level (which implicit objectives guide planning). Our ablation results confirm that no single level alone achieves reliable alignment; the three levels are complementary, and their co-evolution yields the strongest adaptation across successive cycles. We propose NanoResearch, a tri-level co-evolutionary framework that implements layered governance across these dimensions, with empirical evaluations showing consistent improvements across tasks and feedback rounds. This framework suggests that personalization-aware design is not a peripheral enhancement but a prerequisite for research automation systems that are both trustworthy and practically useful.

AI USE STATEMENT

Generative AI tools were used solely to assist with manuscript writing and language polishing. All AI-assisted text was reviewed and edited by the authors before submission.

ETHICS STATEMENT

NanoResearch is designed to assist researchers. As with other LLM-based systems, its outputs may contain hallucinations and should be reviewed by researchers. Researchers remain responsible for the final research decisions and reported findings.

REPRODUCIBILITY STATEMENT

The Method and Experiment Setup sections describe our framework and experimental settings. The supplementary material provides benchmark tasks, baseline configurations, evaluation protocols, and details of the human study to support reproducibility. Results may vary due to stochastic generation and changes in external model APIs.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Jinheon Baek, Sujay Kumar Jauhar, Silviu Cucerzan, and Sung Ju Hwang. Researchagent: Iterative research idea generation over scientific literature with large language models. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 6709–6738, 2025.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.

Shuai Bai, Yuxuan Cai, Ruizhe Chen, Keqin Chen, Xionghui Chen, Zesen Cheng, Lianghao Deng, Wei Ding, Chang Gao, Chunjiang Ge, et al. Qwen3-vl technical report. *arXiv preprint arXiv:2511.21631*, 2025.

Iz Beltagy, Kyle Lo, and Arman Cohan. Scibert: A pretrained language model for scientific text. In *Proceedings of the 2019 conference on empirical methods in natural language processing and the 9th international joint conference on natural language processing (EMNLP-IJCNLP)*, pp. 3615–3620, 2019.

Thomas Kleine Buening, Jonas Hübotter, Barna Pásztor, Idan Shenfeld, Giorgia Ramponi, and Andreas Krause. Aligning language models from user interactions. *arXiv preprint arXiv:2603.12273*, 2026.

Isabel Cachola, Kyle Lo, Arman Cohan, and Daniel S Weld. Tldr: Extreme summarization of scientific documents. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4766–4777, 2020.

Jeff Clune. Ai-gas: Ai-generating algorithms, an alternate paradigm for producing general artificial intelligence. *arXiv preprint arXiv:1905.10985*, 2019.

Shubham Gandhi, Dhruv Shah, Manasi Patwardhan, Lovekesh Vig, and Gautam Shroff. Researchcodeagent: An llm multi-agent system for automated codification of research methodologies. In *International Workshop on AI for Transportation*, pp. 3–37. Springer, 2025.

Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Peiyi Wang, Qihao Zhu, Runxin Xu, Ruoyu Zhang, Shirong Ma, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

Xiang Hu, Hongyu Fu, Jinge Wang, Yifeng Wang, Zhikun Li, Renjun Xu, Yu Lu, Yaochu Jin, Lili Pan, and Zhenzhong Lan. Nova: An iterative planning and search approach to enhance novelty and diversity of llm generated ideas. *arXiv preprint arXiv:2410.14255*, 2024.

Kexin Huang, Jaan Altosaar, and Rajesh Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *arXiv preprint arXiv:1904.05342*, 2019.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

Thomas S Kuhn and Ian Hacking. *The structure of scientific revolutions*, volume 2. University of Chicago press Chicago, 1970.

Pat Langley. *Scientific discovery: Computational explorations of the creative processes*. MIT press, 1987.

Bruno Latour, Jonas Salk, and Steve Woolgar. Laboratory life: The construction of scientific facts. 2013.

Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36(4):1234–1240, 2020.

Yu Li, Chenyang Shao, Xinyang Liu, Ruotong Zhao, Peijie Liu, Hongyuan Su, Zhibin Chen, Qinglong Yang, Anjie Xu, Yi Fang, et al. Autosota: An end-to-end automated research system for state-of-the-art ai model discovery. *arXiv preprint arXiv:2604.05550*, 2026.

Zijie Lin, Yiqing Shen, Qilin Cai, He Sun, Jinrui Zhou, and Mingjun Xiao. Autop2c: An llm-based agent framework for code repository generation from multimodal content in academic papers. *arXiv preprint arXiv:2504.20115*, 2025.

Zachary C Lipton and Jacob Steinhardt. Troubling trends in machine learning scholarship: Some ml papers suffer from flaws that could mislead the public and stymie future research. *Queue*, 17(1): 45–77, 2019.

Chris Lu, Cong Lu, Robert Tjarko Lange, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist: Towards fully automated open-ended scientific discovery. *arXiv preprint arXiv:2408.06292*, 2024.

Yougang Lyu, Xi Zhang, Xinhao Yi, Yuyue Zhao, Shuyu Guo, Wenxiang Hu, Jan Piotrowski, Jakub Kaliski, Jacopo Urbani, Zaiqiao Meng, et al. Evoscientist: Towards multi-agent evolving ai scientists for end-to-end scientific discovery. *arXiv preprint arXiv:2603.08127*, 2026.

Jakub Lála, Odhran O'Donoghue, Aleksandar Shtedritski, Sam Cox, Samuel G. Rodriques, and Andrew D. White. Paperqa: Retrieval-augmented generative agent for scientific research, 2023. URL `https://arxiv.org/abs/2312.07559`.

Michael Moor, Oishi Banerjee, Zahra Shakeri Hossein Abad, Harlan M Krumholz, Jure Leskovec, Eric J Topol, and Pranav Rajpurkar. Foundation models for generalist medical artificial intelligence. *Nature*, 616(7956):259–265, 2023.

Roy Schwartz, Jesse Dodge, Noah A Smith, and Oren Etzioni. Green ai. *Communications of the ACM*, 63(12):54–63, 2020.

Chenyang Shao, Dehao Huang, Yu Li, Keyu Zhao, Weiquan Lin, Yining Zhang, Qingbin Zeng, Zhiyu Chen, Tianxing Li, Yifei Huang, et al. Omniscientist: Toward a co-evolving ecosystem of human and ai scientists. *arXiv preprint arXiv:2511.16931*, 2025.

Jiabin Tang, Lianghao Xia, Zhonghang Li, and Chao Huang. Ai-researcher: Autonomous scientific innovation. *arXiv preprint arXiv:2505.18705*, 2025.

InternAgent Team, Bo Zhang, Shiyang Feng, Xiangchao Yan, Jiakang Yuan, Runmin Ma, Yusong Hu, Zhiyin Yu, Xiaohan He, Songtao Huang, et al. Internagent: When agent becomes the scientist–building closed-loop system from hypothesis to verification. *arXiv preprint arXiv:2505.16938*, 2025a.

Kimi Team, Angang Du, Bohong Yin, Bowei Xing, Bowen Qu, Bowen Wang, Cheng Chen, Chenlin Zhang, Chenzhuang Du, Chu Wei, et al. Kimi-vl technical report. *arXiv preprint arXiv:2504.07491*, 2025b.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

David Waltz and Bruce G Buchanan. Automating science. *Science*, 324(5923):43–44, 2009.

Yixuan Weng, Minjun Zhu, Qiujie Xie, Qiyao Sun, Zhen Lin, Sifan Liu, and Yue Zhang. Deepscientist: Advancing frontier-pushing scientific findings progressively. *arXiv preprint arXiv:2509.26603*, 2025.

Michael Wornow, Yizhe Xu, Rahul Thapa, Birju Patel, Ethan Steinberg, Scott Fleming, Michael A Pfeffer, Jason Fries, and Nigam H Shah. The shaky foundations of large language models and foundation models for electronic health records. *npj digital medicine*, 6(1):135, 2023.

Qiujie Xie, Qingqiu Li, Zhuohao Yu, Yuejie Zhang, Yue Zhang, and Linyi Yang. An empirical analysis of uncertainty in large language model evaluations. *arXiv preprint arXiv:2502.10709*, 2025.

Yutaro Yamada, Robert Tjarko Lange, Cong Lu, Shengran Hu, Chris Lu, Jakob Foerster, Jeff Clune, and David Ha. The ai scientist-v2: Workshop-level automated scientific discovery via agentic tree search. *arXiv preprint arXiv:2504.08066*, 2025.

Zijie Yang, Yukai Wang, and Lijing Zhang. Ai becomes a masterbrain scientist. *bioRxiv*, pp. 2023–04, 2023.

Minjun Zhu, Qiujie Xie, Yixuan Weng, Jian Wu, Zhen Lin, Linyi Yang, and Yue Zhang. Ai scientists fail without strong implementation capability. *arXiv preprint arXiv:2506.01372*, 2025.

A Benchmark Construction

To support the personalized, multi-round evaluation described in the main paper, we construct a benchmark of 20 research tasks together with a simulated researcher for each task. The construction is fully driven by Claude, which serves both as the topic generator and as the in-the-loop user during NanoResearch runs.

A.1 Construction Protocol

We prompt Claude to role-play as 20 distinct scientists, each proposing a concrete research topic together with the relevant contextual information. To ensure breadth and comparability across tasks, the generated topics provide cross-domain coverage spanning NLP, CV, Multimodal, Tabular ML, Time Series, Graph ML, and Audio, and each topic specifies explicit user requirements such as reproducibility and methodological focus.

A.2 Topic Schema

Each topic produced by Claude follows a fixed schema with the following fields: `question_id`, `domain`, `difficulty`, `background`, `problem_statement`, `baselines`, `datasets`, `user_requirements`, and `extra_context`. Together, these fields define a self-contained research request that captures both the scientific problem and the simulated researcher's personal preferences and constraints, providing a stable interface between the benchmark and the NanoResearch pipeline.

A.3 Simulated Researcher Feedback

Beyond topic generation, Claude continues to act as the corresponding scientist throughout each NanoResearch run. After observing the intermediate artifacts produced at each stage of the pipeline (ideation, experimentation, and writing), Claude generates feedback according to the persona's predefined preferences, constraints, and `user_requirements`.

A.4 Benchmark Scope and Evaluator Independence

The simulated benchmark provides broad and reproducible coverage, together with multi-round diagnostics, because Claude generates the task specifications and stage-level feedback under a fixed schema. We assess real-user validity separately through the human-authored evaluation in Section N. To separate task feedback from final evaluation, we use GPT-5.4 as the final LLM judge rather than the Claude feedback model. For subjective scoring, the judge input is blinded to system names, round identifiers, and prior scores. Each rubric-specific prompt contains the task specification and only the artifact or execution information required by that rubric; scores from other judges are withheld. The LLM judge and expert evaluator rank the methods in the same order across all subjective dimensions, supporting the robustness of the main comparison to evaluator choice.

A.5 Role-Play Prompt

The full role-play prompt used to instruct Claude to generate the 20 benchmark tasks is shown below. The prompt specifies the target domains and the construction requirements.

Role-play prompt for benchmark topic generation

You are helping construct a benchmark suite for evaluating autonomous research agents. Generate 20 realistic research task specifications across the following seven domains: NLP, Computer Vision, Multimodal Learning, Tabular ML, Time Series, Graph ML, and Audio.
Each task should represent a concrete research topic that a real scientist or practitioner could give to a research collaborator. The goal is to test whether an autonomous research agent can propose, plan, implement, and evaluate a reproducible, benchmark-comparable research idea under explicit user constraints.

For each task, output a JSON object with the following fields:

- `question_id`: a short unique identifier.
- `domain`: one of NLP, CV, Multimodal, Tabular ML, Time Series, Graph ML, or Audio.
- `difficulty`: use `"incremental_innovation"`.
- `background`: a short paragraph explaining why this task is meaningful and benchmarkable.
- `problem_statement`: a concrete research problem the agent should solve.
- `baselines`: a list of standard baselines or reference methods that should be used for comparison.
- `datasets`: a list of public datasets or standard benchmarks suitable for quantitative evaluation.
- `user_requirements`: explicit user preferences and constraints, such as reproducibility, required ablations, metric reporting, baseline compatibility, or compute budget.
- `extra_context`: additional practical constraints, such as expected resource budget, implementation feasibility, avoiding private datasets, and requiring clear ablations.

Construction requirements:

1. Cover all seven domains, with a balanced distribution across NLP, CV, Multimodal Learning, Tabular ML, Time Series, Graph ML, and Audio.
2. Every task must be benchmarkable: it must specify public datasets, standard baselines, and evaluation-compatible outputs.
3. Prefer tasks with clear quantitative evaluation protocols over open-ended or purely qualitative research questions.
4. Prefer realistic incremental research problems that admit implementable methods, controlled ablations, and fair baseline comparisons.
5. Avoid vague topics, purely theoretical topics, tasks requiring private data, and tasks whose success cannot be measured quantitatively.

6. Ensure the topics are compatible with multiple autonomous research-agent baselines: do not rely on NanoResearch-specific mechanisms or proprietary APIs.
7. Use a unified schema so that downstream systems can parse the topic into selected idea, proposed method, experiment plan, ablations, metrics, and benchmark targets.

Return exactly a JSON list of 20 task objects and no extra commentary.

B Implementation Details

This section summarizes the models and tools used in NanoResearch. Literature retrieval is performed via the OpenAlex API. The Planner of the Orchestrator is the only trainable component and is instantiated as Qwen3-8B-Instruct. For the other agents, Ideation, Planning, and Setup/Execution use DeepSeek-V3.2; Coding/Debugging uses GPT-5.3-Codex; Writing and figure prompt/code generation use Claude Sonnet 4.6; figure image generation uses Gemini 3.1 Flash Image Preview; Review uses Gemini 3.1 Flash Lite; and Revision uses Gemini 3 Pro. For the simulated-researcher evaluation, Alignment, Novelty, and Writing are rated by GPT-5.4 and by human experts; the human-researcher evaluation is described separately in Section N.

C Controlled Baseline Protocol

We adapt all systems to the same task-facing protocol. Every method receives the complete, frozen task record, including the research problem, datasets, required baselines, compute limits, user preferences, required ablations, unacceptable designs, and success criteria. The baseline prompts serialize these fields as task instructions even when the original system does not implement a persistent profile object. Consequently, Alignment measures how effectively a system uses shared preference information, rather than rewarding NanoResearch for exclusive access to it.

For every function present in more than one system, the model backend and inference configuration are matched by role: DeepSeek-V3.2 for ideation, planning, and setup/execution; GPT-5.3-Codex for coding and debugging; Claude Sonnet 4.6 for writing and figure prompt/code generation; Gemini 3.1 Flash Lite for review; and Gemini 3 Pro for revision. Within a role, systems share the same API snapshot, temperature/top-p, maximum output and context limits, reasoning setting, timeout, and retry ceiling. These request-level settings are stored in the run configurations retained with the experiment artifacts. All systems also receive the same underlying tool allowlist: OpenAlex retrieval, workspace and file operations, code execution, job submission/control, and error-log access. Dataset, accelerator, wall-clock, and search-budget limits are task-specific but identical across methods.

System-specific prompts, control flow, memory design, and retry decisions within the common ceiling remain part of each method. The Qwen3-8B-Instruct Planner, Skill Bank, Memory Module, and SDPO update are NanoResearch components under study rather than shared infrastructure. Static baselines are run once; the directly comparable NanoResearch result is Round 1. Only NanoResearch R2 and R3 receive the intervening user feedback, so those rows quantify longitudinal personalization rather than a static-baseline advantage. This design controls external model and tool capability while preserving the orchestration differences the comparison is intended to test.

D SDPO Reproducibility Settings

We use the following fixed training recipe for the user-specific SDPO updates. The Planner backbone is Qwen3-8B-Instruct; all backbone weights remain frozen and only LoRA parameters are optimized. One update is performed after each user-feedback round using all valid feedback triples accumulated for that user up to that round.

E Extended Ablation Studies

We extend the ablation study in the main paper with additional component variants. Results are reported in Table 6.

Table 5: SDPO reproducibility settings.

Setting	Configuration
Planner backbone	Qwen3-8B-Instruct
Hardware	One NVIDIA A100 GPU
Update frequency	One SDPO update after each user-feedback round
Data per update	All valid feedback triples accumulated for the current user up to that round
Training sample	$(x, y, \text{feedback})$: Planner input, original Planner output, and subsequent user feedback
Student forward pass	Conditions on (x, y) and computes token-level log probabilities over y
Teacher forward pass	Conditions on $(x, \text{feedback}, y)$ using the same model; gradients are stopped through this branch
Token-level advantage	Feedback-conditioned token log probability minus original-policy token log probability
Trainable parameters	LoRA parameters only; the backbone remains frozen
LoRA rank / alpha / dropout	16 / 32 / 0.05
LoRA target modules	`q_proj, k_proj, v_proj, o_proj, gate_proj, up_proj, down_proj`
Optimizer	AdamW; learning rate $1e{-4}$; weight decay 0
Batching	Micro-batch size 1; gradient accumulation 1; effective batch size 1
Update steps	50 optimization steps per feedback round
Sequence lengths	Maximum sequence length 2,048 tokens; maximum optimized response length 512 tokens
Truncation	Left truncation, preserving the feedback and response tail
Clipping	Advantage $[-5, 5]$; global gradient norm 1.0
Precision / seed	bfloat16 / 42
Checkpointing	Save the user-specific LoRA adapter after each update and resume it in the next round

Table 6: Extended ablation on component variants. "w/o Planner" indicates the Planner Model is removed and the module operates independently. **Bold** indicates the best result.

Settings	Options	User Req. ↑	Novel. ↑	E2E ↑	Proj. Perf. ↑	Debug ↓	Writ. ↑
Component	Planner Model Only	7.820	3.532	0.835	0.642	3.483	4.70
	Memory Only (w/o Planner)	7.960	3.899	0.968	0.631	3.183	4.85
	Skill Bank Only (w/o Planner)	7.880	3.715	0.943	0.604	3.732	4.90
	Planner + SDPO	7.900	3.860	0.979	0.653	3.047	4.80
	NanoResearch (full)	**8.163**	**4.960**	**1.000**	**0.684**	**2.342**	**5.42**

We further analyze the component variants, as summarized in Table 6. The Planner-Only variant defines the performance floor, with the lowest novelty (3.532), E2E (0.835), and writing (4.70). Memory-Only and Skill-Bank-Only both improve E2E and writing relative to this floor, showing that episodic experience and procedural skills each provide useful standalone guidance. Their Project Performance (0.631 and 0.604) remains below Planner-based variants, and Skill-Bank-Only incurs the highest debug rounds (3.732). This gap establishes that retrieval alone is not a substitute for strategic planning. Pairing the Planner with SDPO yields the strongest configuration among the variants that omit Skill Bank and Memory (E2E 0.979, Proj. Perf. 0.653, debug 3.047). This direct Planner-only comparison demonstrates a contribution from the parameter update beyond external retrieval. The full system is strongest across every reported metric, establishing the complementary gains from SDPO, experiential context, and procedural skills.

F User Requirement Alignment Prompt

The *Compliance Score* (Align.) measures how well the generated research artifacts match the user's stated requirements, including the target task, datasets, baselines, ablations, compute budget, and methodological preferences. We prompt an LLM judge with a structured JSON object describing the research task specification, the user requirements, the generated idea or selected hypothesis, the generated experiment plan or blueprint, and benchmark or execution status when available. The judge is instructed to reward outputs that are practical, benchmarkable, feasible, and aligned

with the user's stated preferences, and to penalize plans that miss important requirements, use incompatible datasets or baselines, propose infeasible methods, lack required ablations, or fail to address the stated task. Scores follow a 1–10 rubric: 1–2 indicates the output largely ignores the user requirements, uses incompatible datasets or baselines, fails to address the stated task, or has no benchmark-comparable result when benchmark comparability is required; 3–4 the output is loosely related to the task but misses major requirements such as the target dataset, required baselines, feasibility constraints, compute constraints, ablation design, or benchmark comparability; 5–6 the output addresses the main task but the idea or experiment plan is incomplete, underspecified, only partially benchmark-compatible, or misses several important user preferences; 7–8 the output satisfies most user requirements, with a relevant idea, a mostly feasible and benchmarkable plan, and the required datasets, baselines, metrics, or ablations mostly covered with only minor omissions; and 9–10 the output strongly satisfies the user requirements, with the idea and experiment plan well aligned with the requested method style, appropriate datasets and baselines, feasible compute assumptions, clear evaluation metrics and ablations, and benchmark-comparable execution when required. The judge returns a structured JSON response containing the assigned `alignment_score` and a free-text `feedback`. The full prompt template is shown below.

Prompt: User Requirement Alignment Judge

You are an expert research evaluator. Determine whether the generated idea and experiment plan satisfy the stated user requirements. The input is a JSON object containing: the research task specification, including domain, background, problem statement, target datasets, and known baselines; the user requirements, including preferences about feasibility, reproducibility, benchmarkability, ablations, compute budget, or method style; the generated research idea or selected hypothesis; the generated experiment plan or blueprint, including proposed method, metrics, ablations, and compute requirements when available; and benchmark or execution status when available, including whether the result is benchmark-comparable. Return JSON only with keys `alignment_score` (a numeric score from 1 to 10) and `feedback` (a concise explanation).

Scoring rubric:

- 1–2 = the output largely ignores the user requirements, uses incompatible datasets or baselines, fails to address the stated task, or has no benchmark-comparable result when benchmark comparability is required;
- 3–4 = the output is loosely related to the task, but misses major requirements such as the target dataset, required baselines, feasibility constraints, compute constraints, ablation design, or benchmark comparability;
- 5–6 = the output addresses the main task, but the idea or experiment plan is incomplete, underspecified, only partially benchmark-compatible, or misses several important user preferences;
- 7–8 = the output satisfies most user requirements, with a relevant idea, a mostly feasible and benchmarkable plan, and the required datasets, baselines, metrics, or ablations mostly covered, with only minor omissions or underspecified details;
- 9–10 = the output strongly satisfies the user requirements, with the idea and experiment plan well aligned with the requested method style, appropriate datasets and baselines, feasible compute assumptions, clear evaluation metrics and ablations, and benchmark-comparable when required.

Score higher when the plan is practical, benchmarkable, feasible, and aligned with the user's stated preferences. Score lower when the plan misses important requirements, uses incompatible datasets or baselines, proposes an infeasible method, lacks required ablations, has benchmark-misaligned execution, or does not address the stated task.

G Novelty Evaluation Prompt

To assess the *Novelty Score* reported in the main experiments, we prompt an LLM judge with the proposed idea. The judge is instructed to focus on the *core mechanism* rather than surface-level complexity, and to penalize trivial modifications such as backbone swaps, hyper-parameter tuning, regularization tricks, or data-augmentation changes. Scores follow a 1–10 rubric: 1–2 indicates a near-duplicate of existing baselines with only superficial differences; 3–4 a weak incremental modification with high overlap in core method and contribution; 5–6 moderate incremental novelty with one clear local change such as a new module, loss function, training strategy, or recombination of known components; 7–8 clearly recognizable novelty with a substantively different mechanism,

method structure, or contribution logic; and 9–10 strong novelty with a non-trivial and clearly distinct core idea beyond routine recombination. The judge returns a structured JSON response containing the assigned `novelty_score`, the `closest_baseline` among the provided references, and a free-text `rationale`. The full prompt template is shown below.

Prompt: Novelty Judge

You are an expert research evaluator. Score the novelty of the proposed idea relative to the provided baselines on a 1–10 scale.

Scoring rubric:

- 1–2 = near-duplicate of the baselines with only superficial wording, hyperparameter, or training-detail changes;
- 3–4 = weak incremental modification with high overlap in core method and contribution;
- 5–6 = moderate incremental novelty with one clear local change such as a new module, loss, training strategy, or recombination of known components;
- 7–8 = clearly recognizable novelty with a substantively different mechanism, method structure, or contribution logic relative to the baselines;
- 9–10 = strong novelty with a non-trivial and clearly distinct core idea, not just module swapping or routine recombination.

Judge primarily against the provided baselines, focus on the core mechanism rather than surface complexity, and do not over-score backbone swaps, tuning, regularization, or data augmentation. Return JSON only with keys `novelty_score`, `closest_baseline`, `rationale`.

H Writing Quality Evaluation Prompt

The *Overall Writing Quality* score and its sub-dimensions (*Fluency*, *Motivation Clarity*, and *Preference Alignment*) are obtained by prompting an LLM judge with the full paper draft. The judge evaluates readability, organization, motivation clarity, scientific tone, and consistency with standard academic writing conventions. Scores follow a 1–10 rubric: 1–2 denotes very poor scientific writing that is hard to follow, badly structured, and not usable as a paper draft; 3–4 weak writing quality where some content is present but clarity, organization, and academic style are substantially below standard; 5–6 acceptable draft quality that is readable and partially structured but still rough, uneven, or underdeveloped; 7–8 strong research writing that is clear, coherent, and mostly polished with only minor weaknesses; and 9–10 excellent paper-quality writing that is polished, well-structured, academically credible, and close to submission quality. The judge is instructed to be strict and to use the full scale. It returns a structured JSON response containing the assigned `writing_quality` score and a free-text `rationale`. The full prompt template is shown below.

Prompt: Writing Quality Judge

You are an expert evaluator of scientific writing. Read the provided paper draft and assign a single Writing Quality score from 1 to 10.

Scoring rubric:

- 1–2 = very poor scientific writing; hard to follow, badly structured, and not usable as a paper draft;
- 3–4 = weak writing quality; some content is present, but clarity, organization, and academic style are substantially below standard;
- 5–6 = acceptable draft quality; readable and partially structured, but still rough, uneven, or underdeveloped;
- 7–8 = strong research writing quality; clear, coherent, and mostly polished, with only minor weaknesses;
- 9–10 = excellent paper-quality writing; polished, well-structured, academically credible, and close to submission quality.

Judge the score based on the overall writing quality of the full text, considering readability, organization, motivation clarity, scientific tone, and consistency with the requested writing style. Be strict. Use the full scale. Return JSON only with keys `writing_quality`, `rationale`.

I Algorithm of Stage I: Idea Generation and Planning

Algorithm 1 Stage I: Idea Generation and Planning

Require: Research topic $\mathcal{T}$, user profile $\mathcal{U}$, skill bank $\mathcal{S}$, memory module $\mathcal{M}$, orchestrator $\mathcal{O}$, retry limit $T_{\max}$
Ensure: Experiment blueprint $\mathcal{B}$; updated $\mathcal{S}, \mathcal{M}$

// Ideation Phase
$\mathcal{S}_I, \mathcal{M}_I \leftarrow \text{Retrieve}(\mathcal{S}, \mathcal{M} \mid \mathcal{T}, \mathcal{U})$ ▷ $\mathcal{O}$
$P_I \leftarrow \text{Plan}(\mathcal{T}, \mathcal{U} \mid \mathcal{S}_I, \mathcal{M}_I)$ ▷ $\mathcal{O}$
$L \leftarrow \text{QueryAcademicDB}(P_I)$
$E \leftarrow \text{QuantitativeEvidenceExtract}(L)$
$H = \{h_1, \ldots, h_K\} \leftarrow \text{ReActLoop}(P_I, L, E)$
$h^* \leftarrow \text{NoveltyVerification}(H)$
// Planning Phase
$\mathcal{S}_P, \mathcal{M}_P \leftarrow \text{Retrieve}(\mathcal{S}, \mathcal{M} \mid h^*, \mathcal{U})$ ▷ $\mathcal{O}$
$P_P \leftarrow \text{Plan}(h^*, \mathcal{U} \mid \mathcal{S}_P, \mathcal{M}_P)$ ▷ $\mathcal{O}$
$\mathcal{B}^{(0)} \leftarrow \text{Instantiate}(h^*, P_P, E)$
for $t = 0, 1, \ldots, T_{\max} - 1$ **do**
    $c_{\mathcal{B}}^{(t)} \leftarrow \text{Review}(\mathcal{B}^{(t)})$
    if $\mathcal{B}^{(t)}$ passes review **then break**
    end if
    $\mathcal{B}^{(t+1)} \leftarrow \text{Refine}(\mathcal{B}^{(t)}, c_{\mathcal{B}}^{(t)}, P_P, E)$
end for
$\mathcal{B} \leftarrow \mathcal{B}^{(t)}$
$\mathcal{S}, \mathcal{M} \leftarrow \text{Update}(\mathcal{S}, \mathcal{M} \mid h^*, \mathcal{B}, c_{\mathcal{B}})$ ▷ $\mathcal{O}$
return $\mathcal{B}, \mathcal{S}, \mathcal{M}$

J Algorithm of Stage II: Experimental Validation and Optimization

Algorithm 2 Stage II: Experimental Validation and Optimization

Require: Blueprint $\mathcal{B}$, topic $\mathcal{T}$, user profile $\mathcal{U}$, skill bank $\mathcal{S}$, memory module $\mathcal{M}$, orchestrator $\mathcal{O}$, debug budget $T_{\max}$
Ensure: Codebase $\mathcal{W}$, analysis report $\mathcal{A}$; updated $\mathcal{S}, \mathcal{M}$

// Setup and Coding Phase
PrepareEnv($\mathcal{B}$)
$\mathcal{S}_C, \mathcal{M}_C \leftarrow$ Retrieve($\mathcal{S}, \mathcal{M} \mid \mathcal{B}, \mathcal{U}$) ▷ $\mathcal{O}$
$P_C \leftarrow$ Plan($\mathcal{B}, \mathcal{U} \mid \mathcal{S}_C, \mathcal{M}_C$) ▷ $\mathcal{O}$
$\mathcal{W}^{(0)} \leftarrow$ Code($P_C, \mathcal{B}$)
// Execution and Automated Debugging Phase
for $t = 0, 1, \ldots, T_{\max} - 1$ **do**
    R_{raw}, status $\leftarrow$ Execute($\mathcal{W}^{(t)}$)
    if status $=$ success **then break**
    end if
    $\mathcal{W}^{(t+1)} \leftarrow$ Debug($\mathcal{W}^{(t)} \mid \mathcal{S}_C, \mathcal{M}_C$)
end for
$\mathcal{W} \leftarrow \mathcal{W}^{(t)}$
// Analysis Phase
$\mathcal{A} \leftarrow$ Analyze($R_{\text{raw}}, \mathcal{B}, \mathcal{T}$)
$\mathcal{S}, \mathcal{M} \leftarrow$ Update($\mathcal{S}, \mathcal{M} \mid \mathcal{W}, \mathcal{A}$) ▷ $\mathcal{O}$
return $\mathcal{W}, \mathcal{A}, \mathcal{S}, \mathcal{M}$

K Algorithm of Stage III: Paper Writing and Review

Algorithm 3 Stage III: Paper Writing and Review

Require: Blueprint $\mathcal{B}$, analysis report $\mathcal{A}$, user profile $\mathcal{U}$, skill bank $\mathcal{S}$, memory module $\mathcal{M}$, orchestrator $\mathcal{O}$, review budget $T_{\max}$
Ensure: Final paper $\mathcal{P}$; updated $\mathcal{S}, \mathcal{M}$

// Writing Phase
$\mathcal{S}_W, \mathcal{M}_W \leftarrow$ Retrieve($\mathcal{S}, \mathcal{M} \mid \mathcal{B}, \mathcal{A}, \mathcal{U}$) ▷ $\mathcal{O}$
$P_W \leftarrow$ Plan($\mathcal{B}, \mathcal{A}, \mathcal{U} \mid \mathcal{S}_W, \mathcal{M}_W$) ▷ $\mathcal{O}$
Draft$^{(0)} \leftarrow$ Write($P_W, \mathcal{B}, \mathcal{A} \mid \mathcal{S}_W, \mathcal{M}_W$)
// Review Phase
for $t = 0, 1, \ldots, T_{\max} - 1$ **do**
    $f_R^{(t)} \leftarrow$ Review(Draft$^{(t)}$)
    if quality threshold met **then break**
    end if
    Draft$^{(t+1)} \leftarrow$ Revise(Draft$^{(t)}$, $f_R^{(t)}$)
end for
$\mathcal{P} \leftarrow$ Draft$^{(t)}$
$\mathcal{S}, \mathcal{M} \leftarrow$ Update($\mathcal{S}, \mathcal{M} \mid \mathcal{P}, f_R$) ▷ $\mathcal{O}$
return $\mathcal{P}, \mathcal{S}, \mathcal{M}$

L Case Study: Three Profiles, One Topic

This section presents an end-to-end walkthrough of how Nano Research adapts the same research topic to three different user profiles. For each profile, we show some intermediate pipeline outputs: the *experiment blueprint* (Stage I), the *coding output* (Stage II), and the *writing style* (Stage III). The shared topic and baselines are held fixed; what varies is the inductive bias of the proposed method, the shape of the code interface, and the tone of the writing.

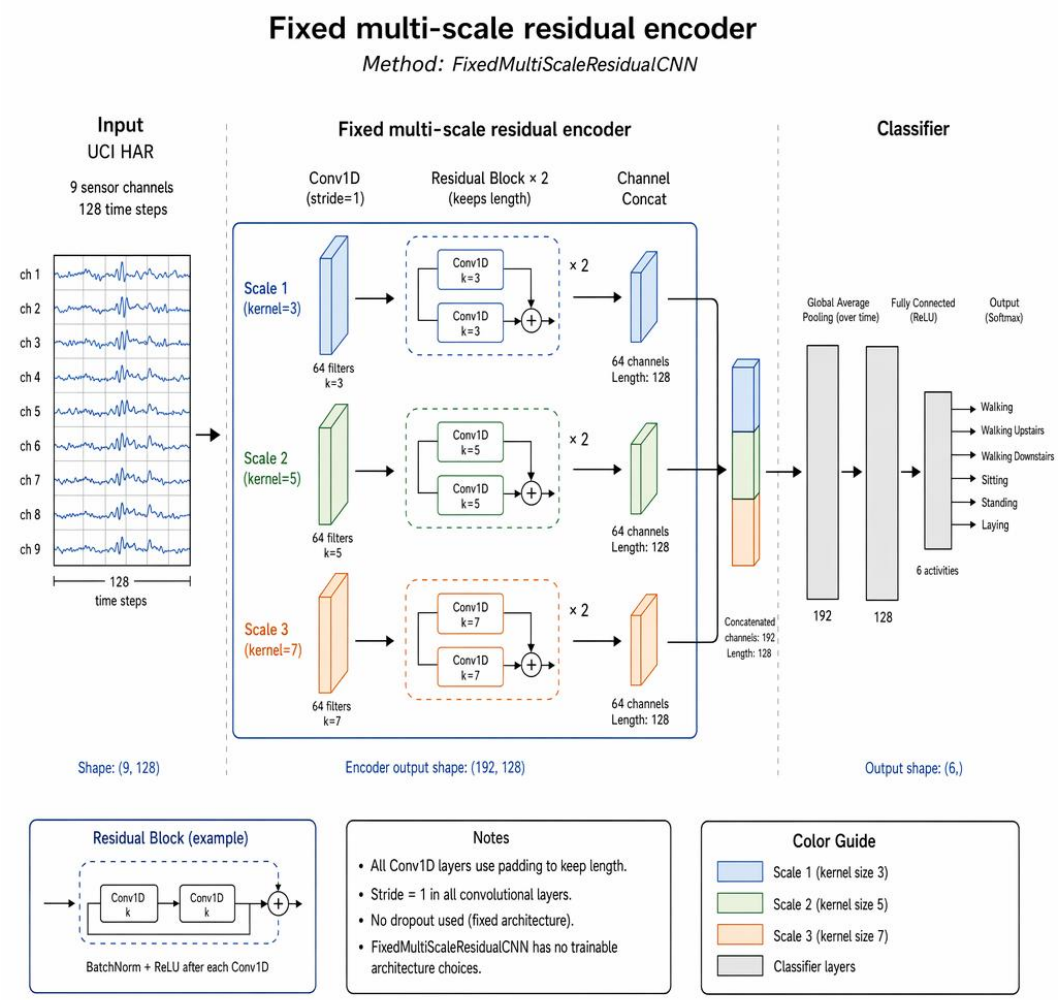


Figure 6: Architecture diagram for Profile A (Evidence-First Scientist).

L.1 Shared Topic and Three Inductive Biases

Shared research topic. The shared topic is *lightweight time-series sensor classification on UCI HAR*. The goal is to design a compact model for wearable sensor activity recognition, compare it with a 1D CNN, a GRU, and InceptionTime-small, and keep the full experiment feasible on a single GPU.

Three inductive biases. At a high level, the three profiles lead to three different inductive biases over the same UCI HAR input, summarized in Table 7.

L.2 Profile A: Evidence-First Scientist

Stage I: Experiment blueprint.

- **title:** Lightweight Fixed Multi-Scale Residual CNN for Controlled Sensor Classification on UCI HAR.
- **proposed_method.name:** FixedMultiScaleResidualCNN.
- **proposed_method.description:** A standard 1D CNN is augmented with parallel fixed-kernel temporal branches at scales 3, 5, and 7, followed by feature fusion and a residual

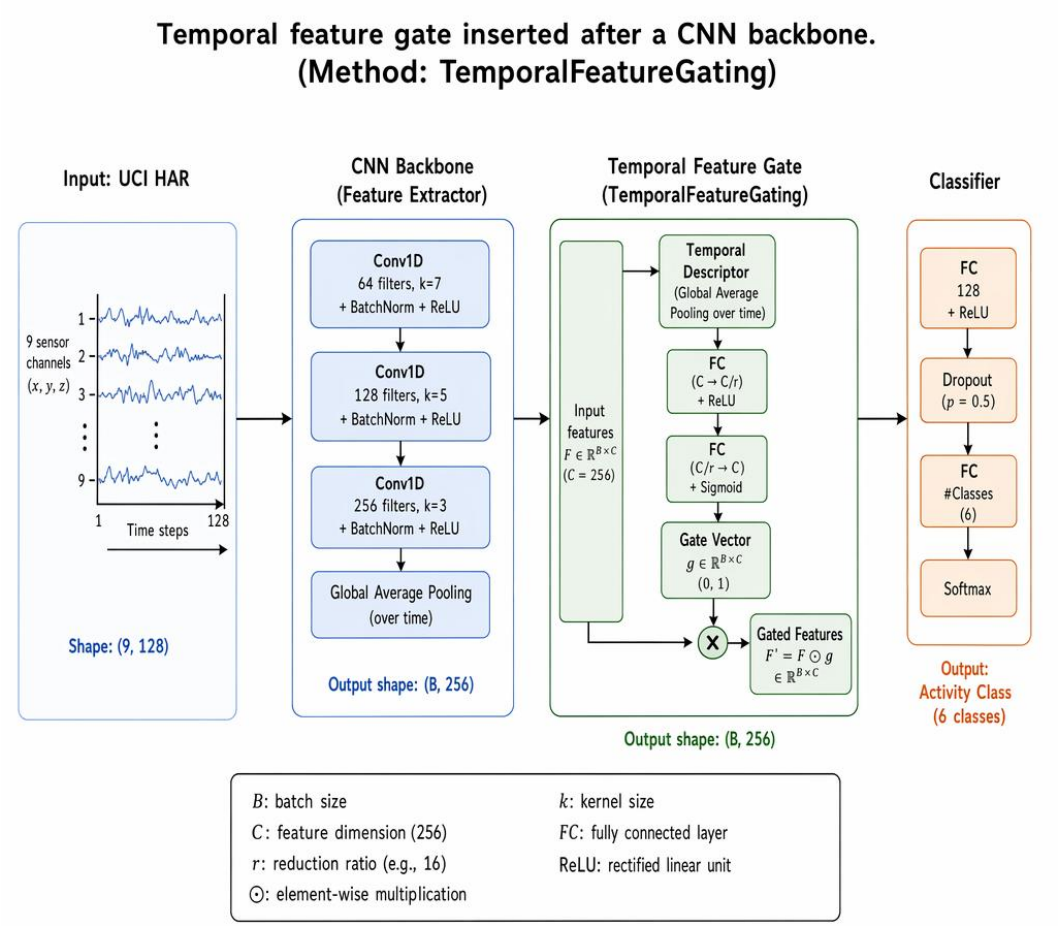


Figure 7: Architecture diagram for Profile B (Ablation-Focused Researcher).

Profile	Core modeling bias	Intuition
A: Evidence-First Scientist	Fixed multi-scale temporal features	Use a stable, low-variance model to test whether fixed temporal receptive fields help.
B: Ablation-Focused Researcher	Small temporal gating module	Add one compact, inspectable contribution that can be directly removed or simplified.
C: Benchmark-Driven Exploratory Researcher	Sample-adaptive evidence routing	Use a stronger dynamic mechanism that can support broader benchmark-facing claims.

Table 7: Three inductive biases derived from the same shared topic. The scientific contract is fixed; the modeling bias, code interface, and writing style adapt to the user profile.

projection. The design keeps the model close to the baseline while testing whether broader fixed receptive fields improve wearable sensor classification.

- **proposed_method.key_components:** fixed multi-scale Conv1d branches; residual feature fusion; compact classification head.

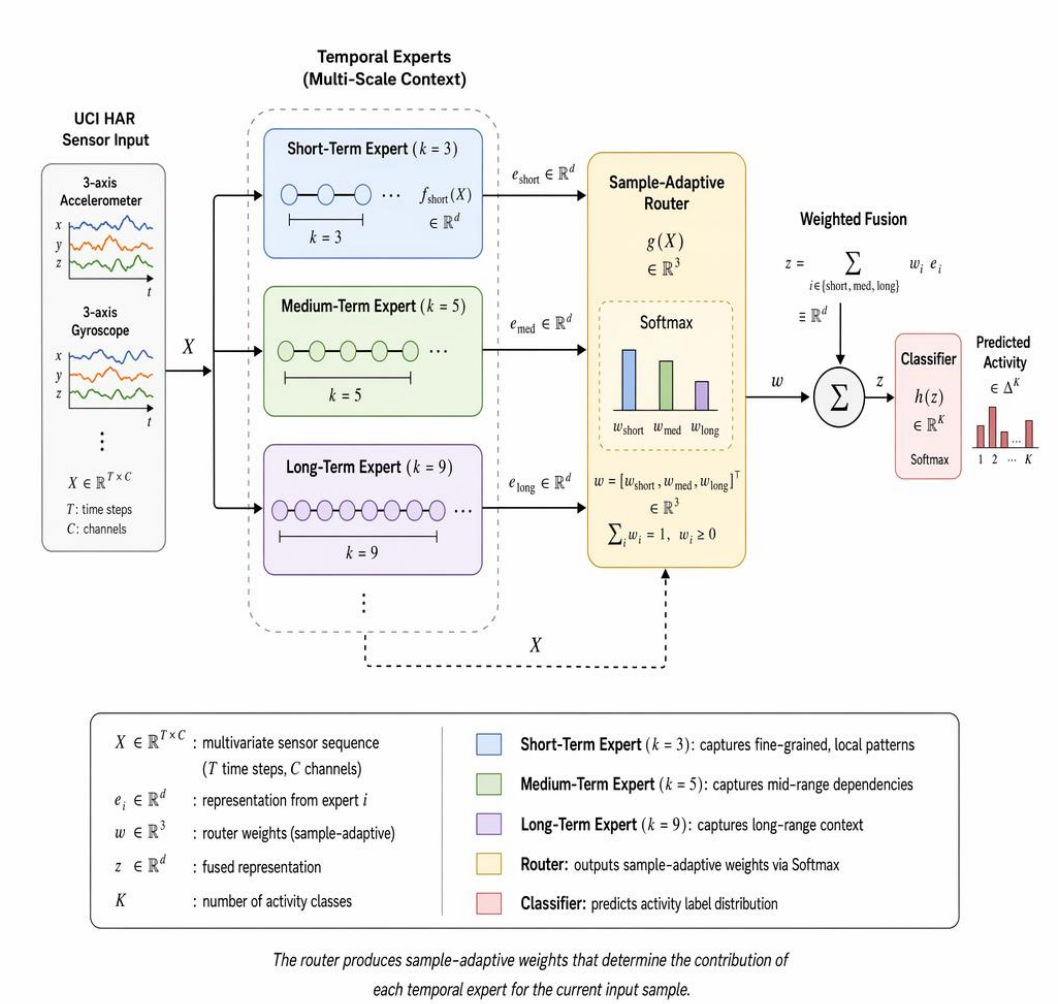


Figure 8: Architecture diagram for Profile C (Benchmark-Driven Exploratory Researcher).

- **proposed_method.architecture:** Input (9 channels × 128 time steps) → parallel Conv1d branches with kernel sizes [3, 5, 7] → concatenation → 1×1 projection → residual addition → global average pooling → linear classifier.
- **ablation_groups:** kernel-scale ablation with `kernel_sizes=[5]` versus `[3,5,7]`; residual ablation with the skip connection removed; parameter-control ablation with matched channel width.

Stage II: Coding output. The code is as follows.

```
def build_fixed_multiscale_encoder(num_channels: int):
    """
    Stable encoder for a controlled UCI HAR study.
    The design uses fixed temporal scales rather than sample-adaptive
    routing.
    """
    return FixedScaleTemporalEncoder(
        input_channels=num_channels,
        kernel_sizes=(3, 5, 7),
        merge="concat_then_1x1",
        residual=True,
```

```
        learned_gate=False,
        learned_router=False,
    )
```

Stage III: Writing style.

> *Rather than claiming a new general architecture, we ask a narrower question: do fixed multi-scale temporal features yield reproducible gains under matched training conditions? The proposed encoder changes only the temporal receptive-field structure of a standard 1D CNN, allowing us to test whether short-, medium-, and longer-range convolutional features provide consistent gains without introducing sample-adaptive routing.*

L.3 Profile B: Ablation-Focused Researcher

Stage I: Experiment blueprint.

- **title:** Temporal Feature Gating for Lightweight UCI HAR Classification.
- **proposed_method.name:** TemporalFeatureGating.
- **proposed_method.description:** A lightweight temporal gate is inserted after the 1D CNN feature extractor to predict importance weights over time steps before global pooling. The module is separated from the backbone so that its contribution can be isolated without changing the data loader, optimizer, or classifier.
- **proposed_method.key_components:** standard 1D CNN backbone; temporal gate with small bottleneck; weighted temporal pooling; compact classification head.
- **proposed_method.architecture:** Input (9 channels $\times$ 128 time steps) $\rightarrow$ 1D CNN feature extractor $\rightarrow$ temporal gate MLP $\rightarrow$ feature reweighting over time $\rightarrow$ global average pooling $\rightarrow$ linear classifier.
- **ablation_groups:** gate removal ablation with `gate=None`; static-gate ablation with uniform temporal weights; bottleneck-size ablation with full versus tiny temporal gate.

Stage II: Coding output. The code is as follows.

```
def build_temporal_gate_encoder(
    num_channels: int,
    ablation: str = "full",
):
    """
    Compact encoder with a single reviewer-facing contribution module.
    The gate can be removed or simplified without changing the backbone.
    """
    gate = {
        "full":        TemporalGate(bottleneck=8, mode="learned"),
        "static_gate": TemporalGate(mode="uniform"),
        "tiny_gate":   TemporalGate(bottleneck=2, mode="learned"),
        "no_gate":     None,
    }[ablation]

    return TemporalGateEncoder(
        input_channels=num_channels,
        backbone="standard_1d_cnn",
        gate=gate,
    )
```

Stage III: Writing style.

> *The central claim is intentionally simple: a removable temporal gate improves a 1D CNN only if its learned weighting survives direct removal, static-gate, and tiny-gate ablations. Because the gate can be removed, frozen to uniform weights, or*

compressed to a smaller bottleneck, the same implementation directly tests whether learned temporal weighting explains the observed accuracy–cost tradeoff.

L.4 Profile C: Benchmark-Driven Exploratory Researcher

Stage I: Experiment blueprint.

- **title:** Temporal Evidence Routing for Benchmark-Oriented Sensor Classification.
- **proposed_method.name:** TemporalEvidenceRouting.
- **proposed_method.description:** A multi-expert temporal encoder uses short-, medium-, and long-range convolutional experts selected by a lightweight sample-adaptive router. The design turns multi-scale feature extraction into an input-dependent routing problem and supports broader benchmark-facing comparisons.
- **proposed_method.key_components:** short-range Conv1d expert; medium-range Conv1d expert; long-range Conv1d expert; sample-adaptive evidence router; shared classification head.
- **proposed_method.architecture:** Input (9 channels $\times$ 128 time steps) $\rightarrow$ temporal experts with kernel sizes [3, 5, 9] $\rightarrow$ router predicts expert weights per sample $\rightarrow$ weighted expert fusion $\rightarrow$ global pooling $\rightarrow$ linear classifier.
- **ablation_groups:** single-expert ablation; uniform-routing ablation; no-router feature-fusion ablation; routing-statistics analysis across activity classes.

Stage II: Coding output. The code is as follows.

```
def build_temporal_evidence_router(
    num_channels: int,
    routing: str = "sample_adaptive",
):
    """
    Benchmark-facing encoder with a named dynamic mechanism.
    The model routes each sample across temporal experts.
    """
    return TemporalEvidenceRouter(
        input_channels=num_channels,
        experts={
            "short_range": ConvExpert(kernel_size=3),
            "mid_range":   ConvExpert(kernel_size=5),
            "long_range":  ConvExpert(kernel_size=9),
        },
        router=EvidenceRouter(mode=routing),
        shared_classifier=True,
        export_routing_statistics=True,
    )
```

Stage III: Writing style.

We cast lightweight sensor classification as temporal evidence routing, where each input dynamically selects the short-, medium-, or long-range expert needed for strong benchmark performance. This design turns multi-scale temporal modeling into a sample-adaptive decision, enabling a stronger benchmark-facing comparison against fixed CNN, recurrent, and Inception-style compact baselines.

M Example of User Profile, Memory, and Skill

To make the abstract notions of *User Profile*, *Memory*, and *Skill* concrete, we provide three representative examples produced by our system.

Profile A: Evidence-First Journal Scientist

User Profile.

- **Research preference:** exact reruns, explicit controls, reproducible ablations, and conservative methods built from standard PyTorch components.
- **Archetype:** ai4science_journal.
- **Domain:** Time Series.
- **Method preference:** prefer exact reruns, explicit controls, and reproducible ablations over speculative novelty.
- **Risk preference:** low.
- **Baseline/ablation strictness:** very high.
- **Resource budget:** 1×A100 80GB, 5 days.
- **Feasibility bias:** prefer explicit reproducibility steps, deterministic settings, and auditable experiment plans.
- **Writing tone:** highly restrained.
- **Claim strength:** conservative.
- **Section organization:** journal-style with dense evidence and careful limitations.
- **Venue style:** Nature/Springer journal.
- **LaTeX template preference:** nature_springer.
- **Figure style:** composite scientific figure.
- **Caption style:** self-contained dense.
- **Priority feedback:** missing controls, weak reproducibility details, or hidden implementation variance.
- **Unacceptable errors:** overclaiming biological or physical conclusions, or under-specifying data provenance.
- **Router hints:** prefer profile over sparse memory; focus writing on conservative claims and dense evidence; focus planning on reproducibility and controlled compute.
- **Persona brief:** AI4Science persona that prioritizes reproducibility, exact reruns, and explicit experimental controls over flashy novelty.

Profile summary. The user prioritizes exact reruns, deterministic evaluation, explicit controls, and auditable ablations. They prefer conservative methods built from standard PyTorch components over speculative architectural novelty. This profile tends to favor stable model designs with few dynamic decisions, and a restrained writing style that states narrow, evidence-grounded claims before broader interpretation.

Memory.

- **Memory 1.**
  - **memory_id:** mem-uci-har-baseline-protocol
  - **memory_type:** project_context
  - **source_stage:** prior ideation and planning
  - **topic_scope:** lightweight time-series sensor classification on UCI HAR
  - **content:** For UCI HAR, the first planning step should reproduce the compact baseline suite before evaluating a new method. The baseline suite should include a standard 1D CNN, a GRU, and InceptionTime-small. These baselines should use the same train/validation/test split, preprocessing pipeline, optimizer, learning rate, batch size, epoch budget, and evaluation script as the proposed method. Do not introduce a new model before establishing that the baseline implementations are executable and benchmark-comparable.
  - **retrieval_rationale:** Retrieved because the current task uses the same UCI HAR sensor-classification setup and the user profile strongly prioritizes exact reruns, controlled comparisons, and conservative evidence.
  - **planning_implication:** The blueprint should place baseline reproduction before proposed-method evaluation and should avoid changing the training protocol between baselines and the proposed model.
  - **coding_implication:** Implement one shared dataloader, one shared training loop, one shared evaluation function, and model-specific constructors only for the architecture difference.

- **writing_implication:** Report the proposed method as a controlled extension after baseline reproduction, not as an unconstrained new architecture.
- **failure_mode_to_avoid:** Do not compare the proposed method against baselines trained with different preprocessing, longer schedules, different random seeds, or missing validation controls.

- **Memory 2.**
  - **memory_id:** mem-uci-har-fixed-controls
  - **memory_type:** decision_history
  - **source_stage:** prior planning critique
  - **topic_scope:** UCI HAR controlled lightweight benchmark
  - **content:** For this topic, experimental controls should remain fixed across all variants. The data split, sensor-channel preprocessing, normalization statistics, optimizer, learning rate, batch size, epoch budget, early-stopping rule, seed schedule, and metric computation should be identical for the 1D CNN, GRU, InceptionTime-small, proposed method, and all ablations. Any unavoidable implementation difference must be explicitly logged and justified.
  - **retrieval_rationale:** Retrieved because hidden implementation variance is especially harmful for this user's evidence-first profile and because UCI HAR is small enough that uncontrolled variance can dominate the reported gain.
  - **planning_implication:** The experiment blueprint should specify fixed controls directly rather than leaving them to the coding stage.
  - **coding_implication:** Put shared hyperparameters in a single config object and prevent per-model overrides unless the override is explicitly declared as an ablation.
  - **analysis_implication:** Report mean and standard deviation across repeated runs when possible, and separate accuracy changes from parameter-count or training-cost changes.
  - **failure_mode_to_avoid:** Do not let the proposed model receive extra epochs, different augmentation, different class weighting, or a tuned learning rate that the baselines do not receive.

- **Memory 3.**
  - **memory_id:** mem-low-risk-temporal-module
  - **memory_type:** promising_direction
  - **source_stage:** ideation reflection
  - **topic_scope:** compact wearable-sensor architectures
  - **content:** A low-risk direction for UCI HAR is to add a small temporal module to a standard 1D CNN backbone instead of replacing the whole architecture. The module should test a narrow hypothesis about temporal feature extraction while preserving the baseline model's overall training behavior. Suitable directions include fixed multi-scale convolutions, shallow residual temporal blocks, or parameter-matched receptive-field changes. Avoid high-variance dynamic routing or attention-heavy mechanisms for this profile unless the task explicitly requires them.
  - **retrieval_rationale:** Retrieved because the user profile favors conservative standard PyTorch components and narrow claims over speculative architectural novelty.
  - **planning_implication:** Select a method that changes the temporal receptive-field structure while keeping the backbone recognizable as a 1D CNN.
  - **coding_implication:** Implement the new component as a small encoder block with a clear on/off or simplified variant for ablation.
  - **writing_implication:** Frame the contribution as a controlled hypothesis test rather than a broadly general architecture.
  - **failure_mode_to_avoid:** Do not propose a full replacement architecture whose gains cannot be attributed to one specific temporal modeling change.

Skill.

- **Skill 1.**
  - **skill_id:** skill-deterministic-experiment-contract
  - **skill_type:** planning_and_execution_rule
  - **name:** Write an explicit deterministic experiment contract.
  - **when_to_apply:** Use when the user profile emphasizes reproducibility, exact reruns, auditability, or controlled scientific evidence.

  - **procedure:** Specify fixed random seeds for Python, NumPy, PyTorch, and CUDA. Save the train/validation/test split indices to disk. Log package versions, CUDA version, device name, command-line arguments, config files, and git state when available. Enable deterministic PyTorch settings where practical and record any operations that remain nondeterministic. Make the same seed schedule available to baselines, proposed methods, and ablations.
  - **planning_effect:** The blueprint must include reproducibility controls as part of the experimental design, not as an optional implementation detail.
  - **coding_effect:** The generated code should expose a single `seed_everything` utility, a saved-split loader, and a run manifest written with every experiment.
  - **review_check:** A reviewer should be able to rerun the same model variant and recover the same split, configuration, and evaluation protocol.
  - **do_not:** Do not claim reproducibility solely by saying that seeds are fixed; include saved splits, software logging, and shared evaluation scripts.
- **Skill 2.**
  - **skill_id:** skill-one-factor-ablation-design
  - **skill_type:** experiment_design_rule
  - **name:** Design one-factor-at-a-time ablations.
  - **when_to_apply:** Use when the proposed method is a compact extension of a baseline and the main claim depends on attributing gains to a specific component.
  - **procedure:** Start from the full proposed model. Define ablations that remove or simplify exactly one component at a time while keeping the data pipeline, optimizer, training schedule, model width where possible, and evaluation script fixed. Include a parameter-control variant when removing a component changes capacity substantially. Name each ablation by the changed factor rather than by a vague model nickname.
  - **planning_effect:** The blueprint should list ablation groups that map directly to the hypothesis being tested.
  - **coding_effect:** The implementation should use explicit variant flags rather than separate scripts that can silently diverge.
  - **analysis_effect:** Interpret improvements only when the full model beats the one-factor ablations under matched training conditions.
  - **do_not:** Do not combine multiple changes in one ablation, because this makes attribution impossible.

Profile B: Ablation-Focused Conference Researcher

User Profile.

- **Archetype:** nlp_conference.
- **Domain:** Time Series.
- **Method preference:** pragmatic, compact methods with clean ablations, straightforward implementation paths, and reviewer-friendly framing.
- **Risk preference:** moderate.
- **Baseline/ablation strictness:** high.
- **Resource budget:** 1×A100 80GB, 3 days.
- **Feasibility bias:** prefer methods feasible on small-to-medium compute budgets.
- **Writing tone:** restrained academic.
- **Claim strength:** moderate.
- **Section organization:** conference-style, direct and contribution-focused.
- **Venue style:** NeurIPS/ICLR conference.
- **LaTeX template preference:** conference_template.
- **Figure style:** clean benchmark plots.
- **Caption style:** compact but informative.
- **Priority feedback:** needlessly complex methods or ablations that do not clarify the core contribution.
- **Unacceptable errors:** ignoring compute limits or skipping rigorous comparisons.

- **Router hints:** prefer profile over sparse memory; focus writing on direct contribution framing; focus planning on compact, ablatable, single-GPU methods.
- **Persona brief:** NLP conference persona that prefers pragmatic, ablatable methods with clean implementation paths and reviewer-friendly framing.

Profile summary. The user prefers practical methods with clean ablations, straightforward implementation, and reviewer-friendly framing. They are open to moderate novelty when the contribution can be isolated as a compact module. This profile tends to favor code interfaces where the main component can be removed or simplified, and a concise writing style that foregrounds the core claim and its ablation evidence.

Memory.

- **Memory 1.**
  - **memory_id:** mem-plugin-module-reviewability
  - **memory_type:** promising_direction
  - **source_stage:** prior ideation reflection
  - **topic_scope:** compact UCI HAR method design
  - **content:** For a reviewer-friendly UCI HAR paper, a small plug-in module on top of a standard 1D CNN is easier to justify than a full architecture replacement. The contribution should be concentrated in one inspectable mechanism that can be named, removed, simplified, and compared against the same backbone. The method should remain easy to implement in PyTorch and feasible on a single A100 without large sweeps.
  - **retrieval_rationale:** Retrieved because the profile prefers practical conference-style contributions with clean ablations and straightforward implementation paths.
  - **planning_implication:** Select a method whose novelty is local and whose contribution can be explained in one paragraph.
  - **coding_implication:** Implement the contribution as a module argument or model variant flag rather than as a separate monolithic architecture.
  - **writing_implication:** Present the contribution as a compact module with a clear accuracy-cost tradeoff, not as a broad replacement for time-series modeling.
  - **failure_mode_to_avoid:** Do not create a method whose claimed novelty is spread across many small unrelated changes.
- **Memory 2.**
  - **memory_id:** mem-direct-module-ablation-pressure
  - **memory_type:** decision_history
  - **source_stage:** reviewer-style planning critique
  - **topic_scope:** ablation-centered method validation
  - **content:** Reviewers will ask whether the proposed module itself causes the gain. The plan should therefore include direct module-removal, module-simplification, and static-replacement ablations. If the full method uses a learned temporal gate, include a no-gate variant, a uniform/static-gate variant, and a reduced-capacity gate variant. These ablations should share the same backbone, training loop, and evaluation code.
  - **retrieval_rationale:** Retrieved because the current profile values reviewer-facing evidence and because the selected topic is small enough to run direct ablations quickly.
  - **planning_implication:** The blueprint should make the ablation logic visible at the same time as the proposed method, not as an afterthought.
  - **coding_implication:** The model constructor should expose `full`, `no_gate`, `static_gate`, and `tiny_gate` style variants through one unified interface.
  - **analysis_implication:** The final analysis should answer whether learned temporal weighting helps beyond added capacity or implementation noise.
  - **failure_mode_to_avoid:** Do not report only the full model and baselines; that leaves the central mechanism untested.
- **Memory 3.**
  - **memory_id:** mem-uci-har-fast-benchmarkable-iteration
  - **memory_type:** project_context
  - **source_stage:** benchmark planning memory
  - **topic_scope:** UCI HAR single-GPU experiments

- **content:** UCI HAR is small enough for fast iteration, so the experiment should prioritize a clean benchmarkable setup over large hyperparameter sweeps. Use a modest number of seeds or repeated runs if budget allows, but keep the main comparison simple. The plan should report accuracy, macro-F1, parameter count, runtime, and peak GPU memory so that reviewers can evaluate whether the compact module is worth its cost.
- **retrieval_rationale:** Retrieved because the task requires feasibility on one GPU and the profile values practical, reviewer-friendly benchmarking.
- **planning_implication:** Allocate compute to direct baselines and ablations before optional tuning.
- **coding_implication:** Generate a unified training script with model variant flags and automatic resource logging.
- **writing_implication:** Discuss both predictive performance and resource cost rather than only the best accuracy number.
- **failure_mode_to_avoid:** Do not spend the compute budget on broad hyperparameter search that makes the main comparison harder to interpret.

Skill.

- **Skill 1.**
  - **skill_id:** skill-claim-to-ablation-map
  - **skill_type:** reviewer_alignment_rule
  - **name:** Map every method claim to a direct ablation.
  - **when_to_apply:** Use when the paper is intended to be reviewer-friendly and the contribution is a compact module or mechanism.
  - **procedure:** List the method's claimed components. For each component, define a removal, replacement, or simplification ablation that tests whether that component is necessary. Keep the backbone, data pipeline, training schedule, and metrics fixed. Include the ablation name in the blueprint and ensure the coding interface can instantiate it directly.
  - **planning_effect:** The experiment plan should make the expected reviewer question and the corresponding ablation visible together.
  - **coding_effect:** Implement variants as stable flags in one constructor, such as `ablation="full"`, `ablation="no_gate"`, `ablation="static_gate"`, and `ablation="tiny_gate"`.
  - **writing_effect:** Use the ablation results to support or narrow the central claim.
  - **do_not:** Do not introduce components that have no corresponding removal or replacement test.
- **Skill 2.**
  - **skill_id:** skill-compact-module-framing
  - **skill_type:** writing_and_planning_rule
  - **name:** Frame the contribution as an inspectable plug-in module.
  - **when_to_apply:** Use when the target style is a pragmatic conference paper and the method should be easy to benchmark.
  - **procedure:** Name the module, state where it is inserted in the baseline, explain what signal it computes, and identify the direct comparison points. Avoid broad claims about replacing a whole modeling family. Emphasize that the same backbone can be evaluated with the module removed, frozen, or simplified.
  - **planning_effect:** The proposed method should be described around one compact mechanism rather than a bundle of unrelated improvements.
  - **coding_effect:** The module should be separable from the backbone and easy to instantiate, remove, or swap.
  - **writing_effect:** The introduction and method section should foreground what the module changes and why the ablations isolate that change.
  - **do_not:** Do not bury the core contribution inside implementation details that reviewers cannot isolate.

Profile C: Benchmark-Driven Exploratory Researcher

User Profile.

- **Archetype:** high_novelty_exploratory.
- **Domain:** Time Series.
- **Method preference:** strong benchmark coverage, broad comparisons, clear leaderboard-facing evidence, and a named mechanism with visible empirical upside.
- **Risk preference:** high, as long as novelty and compute constraints are explicit.
- **Baseline/ablation strictness:** medium-high.
- **Resource budget:** 2×A100 80GB, 5 days.
- **Feasibility bias:** accept higher-risk proposals if novelty is clear and constraints are explicit.
- **Writing tone:** confident but disciplined.
- **Claim strength:** moderate-high.
- **Section organization:** concept-first with clear positioning.
- **Venue style:** benchmark-heavy conference.
- **LaTeX template preference:** conference_template.
- **Figure style:** dense benchmark tables and comparison plots.
- **Caption style:** informative.
- **Priority feedback:** weak benchmark coverage or insufficient comparison breadth.
- **Unacceptable errors:** novelty claims unsupported by experiment or literature positioning.
- **Router hints:** prefer profile over sparse memory; focus writing on benchmark positioning; focus planning on comparison breadth and clearly named mechanisms.
- **Persona brief:** conference persona that values strong benchmark coverage, broad comparisons, and clear wins on established leaderboards.

Profile summary. The user values broad benchmark coverage, strong comparisons, and clear leaderboard-facing evidence. They accept higher-risk proposals when novelty and compute constraints are explicit. This profile tends to favor more expressive mechanisms with named components or adaptive behavior, and a confident concept-first writing style anchored by benchmark comparisons.

Memory.

- **Memory 1.**
  - **memory_id:** mem-benchmark-breadth-sensor
  - **memory_type:** project_context
  - **source_stage:** prior benchmark planning
  - **topic_scope:** benchmark-oriented lightweight sensor classification
  - **content:** A method intended for strong benchmark positioning should not rely only on UCI HAR if another lightweight sensor dataset is feasible within budget. UCI HAR can remain the primary dataset, but the plan should reserve a secondary evaluation path for an additional wearable or sensor-classification dataset. If the secondary dataset cannot be completed, the paper should explicitly state this limitation and avoid broad leaderboard-style claims.
  - **retrieval_rationale:** Retrieved because the profile values broad comparisons and leaderboard-facing evidence rather than a single controlled case.
  - **planning_implication:** The blueprint should include a primary UCI HAR protocol and an optional secondary benchmark protocol with the same model variants.
  - **coding_implication:** Design the dataloader and training script so that dataset selection is configurable rather than hard-coded to UCI HAR.
  - **analysis_implication:** Separate single-dataset findings from cross-dataset findings.
  - **failure_mode_to_avoid:** Do not claim benchmark generality from one dataset unless comparison breadth is actually present.
- **Memory 2.**
  - **memory_id:** mem-named-concept-positioning
  - **memory_type:** writing_context
  - **source_stage:** prior paper review
  - **topic_scope:** concept-first conference writing

- **content:** The introduction should make the core concept easy to remember and distinguish from standard CNN, recurrent, and Inception-style baselines. A benchmark-facing method benefits from a named mechanism that captures the central modeling idea. The name should correspond to an actual architectural behavior, not just a rebranding of ordinary convolution or pooling.
- **retrieval_rationale:** Retrieved because the user profile prefers confident concept-first writing and visible empirical upside.
- **planning_implication:** Select or name a method around a mechanism that can be explained as more than a local implementation tweak.
- **coding_implication:** Expose internal quantities, such as routing weights or expert usage, when they support the named mechanism.
- **writing_implication:** Open the paper with the concept and then connect it to benchmark evidence, resource cost, and ablations.
- **failure_mode_to_avoid:** Do not use a catchy method name if the architecture does not provide a distinct mechanism behind the name.

- **Memory 3.**
  - **memory_id:** mem-benchmark-claims-need-comparison-breadth
  - **memory_type:** decision_history
  - **source_stage:** benchmark result reflection
  - **topic_scope:** conference-style empirical claims
  - **content:** Benchmark-heavy claims require comparison breadth, not only a single best-number improvement. The evaluation should include standard compact baselines, the strongest relevant prior-round compact variant when available, resource metrics, and failure or sensitivity analysis. Strong claims should be conditional on both performance and cost. If the method is adaptive, report whether the adaptive behavior is meaningful rather than only reporting final accuracy.
  - **retrieval_rationale:** Retrieved because the current profile accepts higher-risk methods only when novelty and benchmark evidence are explicit.
  - **planning_implication:** Add comparisons beyond the minimum baseline suite when compute permits, and include resource reporting as part of the main table.
  - **coding_implication:** Log routing statistics, parameter counts, runtime, and memory usage in addition to predictive metrics.
  - **writing_implication:** Use confident language only when benchmark breadth and diagnostic evidence support it.
  - **failure_mode_to_avoid:** Do not turn one positive result into a broad claim about general sensor-classification superiority.

Skill.

- **Skill 1.**
  - **skill_id:** skill-concept-level-mechanism
  - **skill_type:** ideation_and_writing_rule
  - **name:** Convert a local architectural change into a real concept-level mechanism.
  - **when_to_apply:** Use when the profile favors novelty, benchmark positioning, and a memorable contribution.
  - **procedure:** Identify the actual behavior introduced by the method, name that behavior, and ensure the architecture exposes evidence that the behavior occurs. The name should describe a mechanism such as routing, evidence selection, scale specialization, or adaptive aggregation. The method section should connect the name to equations or code-level components.
  - **planning_effect:** The selected idea should have a distinct mechanism that can support a concept-first paper narrative.
  - **coding_effect:** Implement diagnostic outputs that make the mechanism observable, such as routing weights, expert usage, or scale-selection entropy.
  - **writing_effect:** The introduction can use a stronger hook because the named concept corresponds to measurable behavior.
  - **do_not:** Do not rename a standard block without changing or measuring its behavior.
- **Skill 2.**

- **skill_id:** skill-benchmark-claim-evidence-package
- **skill_type:** evaluation_and_writing_rule
- **name:** Pair strong claims with benchmark, resource, and failure evidence.
- **when_to_apply:** Use when the paper makes benchmark-facing or novelty-forward claims.
- **procedure:** For every strong claim, provide a table or diagnostic result that supports it. Pair performance tables with parameter count, runtime, memory, and ablation results. Include at least one failure case, sensitivity analysis, or limitation when the method is more complex than a baseline. If the method uses adaptive behavior, report diagnostics showing when and how adaptation occurs.
- **planning_effect:** The blueprint should include comparison breadth, resource reporting, and diagnostic analysis as first-class experiment outputs.
- **coding_effect:** The generated scripts should automatically save metrics, resource logs, and mechanism-specific diagnostics.
- **writing_effect:** Claims should be confident but conditional on the observed benchmark and diagnostic evidence.
- **do_not:** Do not write leaderboard-style conclusions without resource and failure analysis.

N Human Researcher Evaluation

N.1 Objectives and Participants

The human study evaluates whether NanoResearch can personalize to real research goals and continue adapting after natural-language feedback. It asks whether the system (i) satisfies different researchers' requirements, (ii) adapts from Round 1 to Round 3, and (iii) improves research outputs rather than merely echoing user preferences. The study includes 10 PhD researchers from different areas. Each researcher contributed one real task in their own area, yielding 10 tasks in total. All tasks were defined and frozen before any system was run.

Each task follows a shared schema containing the research question and background, dataset, required baselines, compute constraints, method preferences, required ablations, unacceptable designs, success criteria, and writing preferences. If an LLM assists with formatting, it is restricted to language polishing; the research question, constraints, preferences, and success criteria are set by the researcher.

N.2 Real Task Summary

Table 8 summarizes the ten heterogeneous tasks. The compact summary reports the information needed to reproduce the evaluation; full task records and feedback are retained with the experiment artifacts.

N.3 Task Heterogeneity and System Protocol

The tasks deliberately encode different definitions of success. For example, P2 prioritizes compute savings, P6 interpretability, P7 patient-wise generalization, P9 inference cost, and P1 a credible negative result when strict controls do not support an improvement. All systems receive the same complete initial task record, including the researcher profile and preferences, and use the matched model/tool protocol in Section C. Static baselines (AI-Researcher, AI Scientist-v2, EvoScientist, and DeepScientist) run once while preserving their system-specific orchestration. NanoResearch runs three continuous rounds:

$$\text{Round 1} \rightarrow \text{human feedback} \rightarrow \text{Round 2} \rightarrow \text{human feedback} \rightarrow \text{Round 3}.$$

Round 1 uses only the frozen task and researcher profile. After inspecting the Round-1 artifact, the task owner writes natural-language feedback; the same owner writes a second feedback message after Round 2. Feedback is not generated or paraphrased by an LLM. We retain the original task, profile, feedback text, timestamps, revision history, and corresponding system outputs.

Table 8: Summary of the ten real research tasks used in the human evaluation.

ID	Task and dataset	Required baselines and constraints	Primary goal / metric
P1	Reproducible human-activity recognition (UCI HAR)	1D CNN, GRU, InceptionTime; one RTX 4090, $\leq$6 h per run, three seeds; modular and controlled design	Reproducibility and controlled experiments; Macro-F1
P2	Efficiency–performance trade-off in short-text classification (AG News)	BERT-base, DistilBERT, TF–IDF+LR; one A100, $\leq$24 GPU-hours; no retrieval/ensemble/API	Reduce GPU cost while keeping Macro-F1 within 0.5% of the strongest baseline
P3	Low-data robust sentiment classification (SST-2)	BERT-base, DistilBERT, BiLSTM; no extra pretraining/data; evaluate 100/50/25% data and seeds	Data efficiency and robustness; F1
P4	Fair-compute vision design (CIFAR-10)	ResNet-18, WideResNet, ViT-small; matched epochs and augmentation, up to 4 A100s	Architecture contribution under equal compute; Accuracy
P5	Validity of multimodal interaction (Hateful Memes)	Text-only, image-only, late fusion, VisualBERT-style; two A100s; no closed VLM API	Cross-modal contribution with shuffle/remove-fusion controls; AUROC
P6	Interpretability–performance balance (Adult Income)	Logistic Regression, XGBoost, MLP, FT-Transformer; CPU/consumer GPU, $\leq$2 h; no foundation model/ensemble	Competitive AUROC with improved interpretability or cost; AUROC
P7	Cross-patient ECG generalization (PTB-XL)	1D ResNet, InceptionTime, Transformer ECG baseline; patient-wise split, no leakage or test-set selection	Patient-level generalization; Macro-AUROC
P8	Necessity of complex message passing (Cora/CiteSeer/PubMed)	GCN, GAT, GraphSAGE; equal hyperparameter-search budget; cross-dataset and removal studies	Mechanism robustness across datasets; Accuracy
P9	Deployment-efficient environmental sound classification (ESC-50)	CNN, CRNN, AST-small; report latency, memory, and model size; no ensemble/deploy-time external model	Accuracy–latency–memory trade-off; Accuracy
P10	Sensor redundancy in activity recognition (PAMAP2)	1D CNN, GRU, InceptionTime, sensor-fusion baseline; report FLOPs and inference cost for sensor subsets	Sensor efficiency with near-full-sensor performance; Macro-F1

N.4 Human Evaluation and Blinding

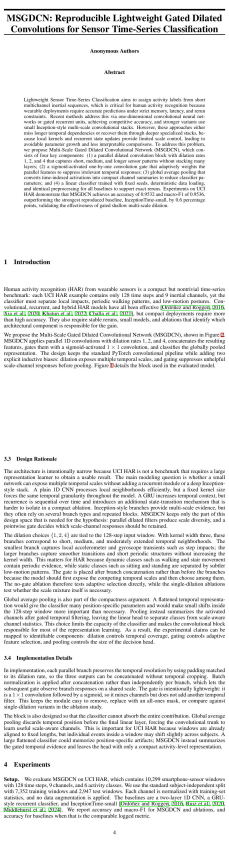
Task owners rate Alignment on a 1–10 scale against the complete task specification, including datasets, baselines, constraints, ablations, success criteria, and prior feedback. Alignment is intentionally owner-facing because the task owner participates in NanoResearch feedback; it measures preference satisfaction and feedback responsiveness, while independent scientific-quality review is reserved for blinded experts. Novelty and Writing are rated by three independent PhD reviewers per task; the task owner never rates these dimensions. Reviewers are assigned by domain background and do not participate in task definition or feedback. They receive anonymized, format-normalized outputs in independently randomized order and do not know the system identity or NanoResearch round. Thus the study contains $10 \times 3 = 30$ independent reviewer assignments for Novelty and Writing.

E2E is the fraction of tasks with a complete executable workflow and valid results. Perf. is not an LLM score: it uses each task's pre-specified benchmark-native metric (Macro-F1, F1, Accuracy, AUROC, or Macro-AUROC), represented on $[0, 1]$ with higher always better. When a task contains multiple datasets (P8), dataset scores are averaged first; the reported Perf. then macro-averages the ten task values so that every researcher contributes equal weight. Alignment remains separate because it additionally captures efficiency, interpretability, deployment, and generalization goals that cannot be reduced to predictive performance.

N.5 Aggregate Results

The aggregate human-evaluation results are reported in the main paper. NanoResearch exceeds every static baseline already in Round 1; R2 and R3 then quantify feedback-driven adaptation for the same task owner. For Novelty and Writing, the three reviewer scores are averaged within each task before macro-averaging across the ten tasks. Alignment is averaged across the ten task owners. The pre-specified aggregation rule is the task-level macro-mean: each heterogeneous research task contributes one value, so no task or reviewer dominates the aggregate. Reviewer scores are averaged within task before aggregation, and reviewer assignments are not treated as independent experimental replicates. Individual ratings, anonymous presentation orders, task records, and feedback histories are retained with the experiment artifacts.

O REAL PAPERS PRODUCED BY OUR SYSTEM



MSGDCN: Reproducible Lightweight Gated Dilated Convolutions for Sensor Time-Series Classification

Anonymous Authors

Abstract

Lightweight Sensor Time-Series Classification aims to assign activity labels from short multichannel inertial sequences, which is critical for human activity recognition because wearable deployments require accurate predictions under strict memory, latency, and power constraints. Recent methods address this via one-dimensional convolutional neural networks or gated recurrent units, achieving competitive accuracy, and stronger variants use dilated convolutions to enlarge receptive fields without adding parameters. However, systems using longer temporal dependencies or recurrent state updates provide limited scale control, leading to avoidable overhead on short, periodic motion windows. We propose the Multi-Scale Gated Dilated Convolutional Network (MSGDCN), which contains four key components: (1) a parallel dilated convolution block with dilation rates 1, 2, and 4 that captures short, medium, and longer sensor patterns without stacking many layers; (2) a sigmoid-activated one-by-one convolution gate that adaptively weights the contribution of each temporal scale at every time step; (3) global average pooling that converts time-indexed activations into compact channel summaries for a linear classifier parameters; and (4) a linear classifier trained with fixed seeds, deterministic data loading, and identical preprocessing for all baselines to support exact reruns. Experiments on UCI HAR demonstrate that MSGDCN achieves an accuracy of 0.9532 and macro-F1 of 0.9536, outperforming the strongest reproduced baseline, InceptionTime-small, by 0.6 percentage points, validating the effectiveness of gated shallow multi-scale dilation.

1 Introduction

Human activity recognition (HAR) from wearable sensors is a compact but nontrivial time-series benchmark: each UCI HAR example contains only 128 time steps and 9 inertial channels, yet the classifier must separate local impacts, periodic walking patterns, and low-motion postures. Convolutional, recurrent, and hybrid HAR models have all been effective [Ordóñez and Roggen (2016, Xu et al. (2019, Khaertdinov et al. (2021, Challa et al. (2022)], but compact deployments require more than high accuracy. They also require stable errors, small models, and ablations that identify which architectural component is responsible for the gain.

We propose the Multi-Scale Gated Dilated Convolutional Network (MSGDCN), shown in Figure 2. MSGDCN applies parallel 1D convolutions with dilation rates 1, 2, and 4, concatenates the resulting features, gates them with a sigmoid-activated 1×1 convolution, and classifies the globally pooled representation. The design keeps the standard PyTorch convolutional pipeline while adding two explicit inductive biases: dilation exposes multiple temporal scales, and gating suppresses unhelpful scale-channel responses before pooling. Figure 1 details the block used in the evaluated model.

3.3 Design Rationale

The architecture is intentionally narrow because UCI HAR is not a benchmark that requires a large representation learner to obtain a usable result. The main modeling question is whether a small network can expose multiple temporal scales without adding a recurrent module or a deep skip-connection style stack. A plain 1D CNN processes local neighborhoods efficiently, but a fixed kernel size forces the same temporal granularity throughout the model. A GRU increases temporal context, but recurrence is sequential over time and introduces an additional state-transition mechanism that is harder to isolate in a compact ablation. Inception-style branches provide multi-scale evidence, but they often rely on several branch types and repeated blocks. MSGDCN keeps only the part of this design that is most relevant to the benchmark: parallel branches for short-scale, medium-scale, and long-scale evidence, plus a gate that decides which scale-channel responses should be retained.

The dilation choices $(1, 2, 4)$ are tied to the 128-step input window. With kernel width three, these branches correspond to short, medium, and moderately extended temporal neighborhoods. The smallest branch captures local accelerometer and gyroscope transients such as step impacts; the larger branches capture smoother transitions and short periodic structures without increasing the kernel width. This matters for HAR because dynamic classes such as walking and stair movement contain periodic evidence, while static classes such as sitting and standing are separated by subtler low-motion patterns. The gate is placed after branch concatenation rather than before the branches because the model should first expose the competing temporal scales and then choose among them. The no-gate dilation therefore tests adaptive selection directly, while the single-dilation ablation test whether the scale mixture itself is necessary.

Global average pooling is also part of the compactness argument. A flattened temporal representation would force the classifier to memorize position-specific patterns, whereas pooling after the 128-step window is more important than necessary. Pooling instead summarizes the activated channels after gated temporal filtering, leaving the linear head to separate classes from scale-aware channel statistics. This choice limits the capacity of the classifier and makes the convolutional block responsible for most of the representation learning. As a result, the experimental claims can be mapped to identifiable components: dilation controls temporal coverage, gating controls adaptive feature selection, and pooling controls the size of the decision head.

3.4 Implementation Details

In implementation, each parallel branch preserves the temporal resolution by using padding matched to its dilation rate, so the three outputs can be concatenated without temporal cropping. Batch normalization is applied after concatenation rather than independently per branch, which lets the gate operate on normalized scale-channel responses. The gate itself is a sigmoid-activated 1×1 convolution followed by a sigmoid, so it moves channels from the raw and gated branches equal filters. This keeps the module easy to remove, replace with an all-ones mask, or compare against single-dilation variants in the ablation study.

The block is also designed so that the classifier cannot absorb the entire contribution. Global average pooling discards temporal position before the final linear layer, forcing the convolutional trunk to learn useful scale-aware channels. This is important for UCI HAR because windows are already aligned to fixed lengths, but individual events inside a window may shift slightly across subjects. A large flattened classifier could memorize position-specific artifacts; MSGDCN instead summarizes the gated temporal evidence and leaves the head with only a compact activity-level representation.

4 Experiments

Setup. We evaluate MSGDCN on UCI HAR, which contains 10,299 smartphone-sensor windows with 128 time steps, 9 channels, and 6 activity classes. We use the standard subject-independent split with 7,352 training windows and 2,947 test windows. Each channel is normalized with training-set statistics, and no data augmentation is applied. The baselines are a two-layer 1D CNN, a GRU, a single recurrent classifier, and InceptionTime-small [Ordóñez and Roggen (2016, Xu et al. (2019, Middlehurst et al. (2024)]. We report accuracy and macro-F1 for MSGDCN and ablations, and accuracy for baselines when that is the comparator's logged metric.

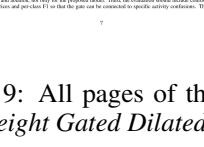


Figure 5: Training dynamics for MSGDCN on UCI HAR. The curves complement the final-score table with convergence and checkpoint behavior.

5 Discussion

The results support the intended local claim: a shallow gated dilated convolution can improve a compact UCI HAR classifier when compared with local controls. The recurrent baseline and a small Inception-style baseline are useful because they test broader model families. The ablations then ask whether the gain comes from collapsing the multi-scale branches, which indicates that temporal scale coverage is the dominant factor in this run. However, the gate still contributes a measurable 1.52-point accuracy gain over the no-gate variant, so it is not merely an unused parameterization detail. This division of gains matches the method design: dilation provides the candidate temporal contexts, and the gate filters them before pooling.

Table 2: Reproducibility status of the completed UCI HAR run.

Item	Status in this report
Dataset split	Official subject-independent UCI HAR split is used.
Preprocessing	Channel normalization uses training-set statistics only.
Baselines	1D CNN, GRU, and InceptionTime-small are included for comparator rows.
Ablations	Single-dilation and no-gate ablation variants are executed.
Training trace	100-epoch accuracy and loss curves are reported in Figure 5.
Missing logs	Multi-seed intervals, latency, FLOPs, and per-class confusion matrices remain to be added.

Table 2 summarizes the boundary of the current evidence. The important point is that the completed run is transparent about what is and is not measured. It gives enough information to judge the proposed mechanism on the selected benchmark, but it does not overstate deployment efficiency or cross-dataset generalization. This distinction is useful for automated research reports because a polished PDF can otherwise make a preliminary experiment look more complete than it is.

The strongest limitation is that the current run is still narrower than a full benchmark paper. UCI HAR is useful because it is standard and reproducible, but it is a single dataset with a fixed window length and a modest number of classes. Additional sensor datasets would test whether the same dilation schedule transfers to longer windows, noisier sensors, and different device placements. The current report also lacks repeated-seed intervals and per-class error analysis. Those are important because HAR errors are rarely uniform: sitting versus standing or walking versus standing or walking upstairs versus walking downstairs, aggregate accuracy can hide whether the model improves the difficult confusions or mainly improves easy dynamic classes.

The next experimental version should therefore add three measurements. First, all models should be rerun across multiple deterministic seeds and reported as mean and standard deviation. Second, the implementation should log parameter count, FLOPs, memory, and inference latency for every baseline and ablation, not only for the proposed model. Third, the evaluation should include confusion matrices and per-class F1 so that the gate can be connected to specific activity confusions. These additions would not change the core method, but they would turn the current single-run evidence into a stronger compact-model evaluation.

Despite these limitations, the report is useful as a writing and evaluation artifact because it contains the essential ingredients of a benchmark-comparable experimental paper: a named method, a bounded hypothesis, real benchmark scores, baseline comparison, ablation evidence, convergence curves, and explicit limitations. The remaining work is framed as future measurement requirements rather than hidden behind unsupported claims. This makes the result suitable as an automated research-writing example, while still leaving clear room for a stronger follow-up experiment.

6 Conclusion

MSGDCN addresses compact sensor time-series classification with parallel dilated convolutions at multiple temporal scales and a sigmoid gate for feature selection, as illustrated in Figure 2 and Figure 1. On UCI HAR, the full model reaches accuracy 0.9532 and macro-F1, exceeding the reproduced 1D CNN accuracy of 0.9471, GRU accuracy of 0.9361, and InceptionTime-small accuracy of 0.9465. The ablation results further support the architectural choices: removing the gate reduces accuracy to 0.9356, while using only dilation rate 1 reaches 0.9325 accuracy.

The main limitations are practical and reproducibility-related. The final run did not log parameter count, so the intended sub-10K design and compactness comparison against baselines remain incompletely verified. The evaluation also covers only UCI HAR with 10,299 windows, so generalization to other sensor modalities, noisier device placements, longer sequences, and datasets with more activities remains untested; in addition, the current report does not include multi-seed uncertainty estimates or ablations using fixed splits and controlled training budgets.

Future work should first add mandatory parameter, FLOP, memory, and latency logging to every run so that comparisons claims can be audited exactly. A second direction is to rerun MSGDCN and all baselines across multiple deterministic seeds and additional compact sensor datasets using the same training budget. Finally, targeted robustness tests under sensor dropout, temporal resampling, and cross-subject or cross-device splits would clarify when gated multi-scale dilation helps and when a simpler 1D CNN is sufficient.

References

Laith Alzubaidi, Jinglan Zhang, Amjad J. Humaidi, Ayad Q. Al-Dujaili, and Ye Duan. Review of deep learning: concepts, cnn architectures, challenges, applications, future directions. *Journal Of Big Data*, 2021. URL https://doi.org/10.1186/s40537-021-00444-8.

Davide Anguita, Alessandro Ghio, Luca Oneto, Xavier Parra, and Jorge L. Reyes-Ortiz. A public domain dataset for human activity recognition using smartphones. In *European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning*, 2013.

Alejandro Barredo Arrieta, Natalia Díaz-Rodríguez, Javier Del Ser, Adrien Bennetot, and Siham Tabik. Explainable artificial intelligence (xai): Concepts, taxonomies, opportunities and challenges toward responsible ai. *Information Fusion*, 2019. URL https://doi.org/10.1016/j.inffus.2019.12.012.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271*, 2018.

Sravan Kumar Challa, Akhilesh Kumar, and Vijay Bhaskar Semwal. A multibranch cnn-bilstm model for human activity recognition using wearable sensor data. *The Visual Computer*, 2021. URL https://doi.org/10.1007/s00371-021-02283-3.

Hassan Ismail Fawaz, Germain Forestier, Jonathan Weber, Lhassane Idoumghar, and Pierre-Alain Muller. Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, 2019. URL https://doi.org/10.1007/s10618-019-00619-1.

Hassan Ismail Fawaz, Benjamin Lucas, Germain Forestier, Charlotte Pelletier, Daniel F. Schmidt, Jonathan Weber, Geoffrey I. Webb, Lhassane Idoumghar, Pierre-Alain Muller, and Francois Petitjean. Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, 2020. URL https://doi.org/10.1007/s10618-020-00710-y.

Navid Mohammadi Foumani, Lynn Miller, Chang Wei Tan, Geoffrey I. Webb, and Germain Forestier. Deep learning for time series classification and extrinsic regression: A current survey. *ACM Computing Surveys*, 2024. URL https://doi.org/10.1145/3649448.

Nils Y. Hammerla, Shane Halloran, and Thomas Plötz. Deep, convolutional, and recurrent models for human activity recognition using wearables. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, 2016.

Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. Lstm fully convolutional networks for time series classification. In *IEEE Access*, 2018. URL https://doi.org/10.1109/ACCESS.2017.2779939.

Mst. Alema Khatun, Mohammad Abu Yousuf, Sabbir Ahmed, Md. Zia Uddin, and Salem A. Alyami. Deep cnn-lstm with self-attention model for human activity recognition using wearable sensor. *IEEE Journal of Translational Engineering in Health and Medicine*, 2022. URL https://doi.org/10.1109/jtehm.2022.3177710.

Matthew Middlehurst, Patrick Schäfer, and Anthony Bagnall. Bake off redux: a review and experimental evaluation of recent time series classification algorithms. *Data Mining and Knowledge Discovery*, 2024. URL https://doi.org/10.1007/s10618-024-01022-1.

Francisco Ordóñez and Daniel Roggen. Deep convolutional and lstm recurrent neural networks for multivariate sensor data for human activity recognition. *Sensors*, 2016. URL https://doi.org/10.3390/s16010115.

Ying Hao Pang, Lew Yee Ping, Goh Fon Ling, Ooi Shih Yin, and Khoh Wee How. Stacked deep analytic model for human activity recognition on a uci har database. *F1000Research*, 2021. URL https://doi.org/10.12688/f1000research.79176.1.

Alejandro Pazos Ruiz, Michael Flynn, James Large, Matthew Middlehurst, and Anthony Bagnall. The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, 2020. URL https://doi.org/10.1007/s10618-020-00727-3.

Iqbal H. Sarker. Deep learning: A comprehensive overview on techniques, taxonomy, applications and research directions. *SN Computer Science*, 2021. URL https://doi.org/10.1007/s42979-021-00815-1.

Chang Wei Tan, Angus Dempster, Christoph Bergmeir, and Geoffrey I. Webb. Multirocket: multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery*, 2022. URL https://doi.org/10.1007/s10618-022-00844-1.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, and Llion Jones. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Kun Xia, Jianguang Huang, and Hanyu Wang. Lstm-cnn architecture for human activity recognition. *IEEE Access*, 2020. URL https://doi.org/10.1109/access.2020.2982225.

Sijie Yan, Yuanjun Xiong, and Dahua Lin. Spatial temporal graph convolutional networks for skeleton-based action recognition. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2018. URL https://doi.org/10.1609/aaai.v32i1.12328.

Figure 9: All pages of the system-generated sensor time-series paper *MSGDCN: Reproducible Lightweight Gated Dilated Convolutions for Sensor Time-Series Classification*.



Figure 10: All pages of the system-generated tabular regression paper *RELM-DReCal: Residual-Enhanced Linear Regression with Learned Feature Recalibration*.

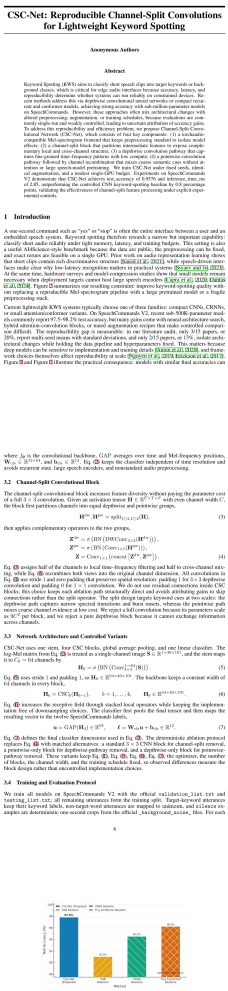


Figure 11: All pages of the system-generated keyword spotting paper *CSC-Net: Reproducible Channel-Split Convolutions for Lightweight Keyword Spotting*.

To complement the evaluations in the main text, we include here three complete papers that were *actually produced end-to-end* by our system, shown in Figures 9, 10, and 11. They demonstrate that the pipeline can deliver coherent manuscripts rather than only isolated artefacts such as ideas, code snippets, or tables. The three examples span distinct research domains, covering sensor time-series classification, tabular regression, and audio keyword spotting, illustrating that the system generalises across problem settings rather than being tied to a single task type. Each manuscript contains a full set of components expected of a conference submission, including a problem motivation, related-work

discussion, formal method description with equations, an experimental protocol with baselines and ablations, figures and tables reporting quantitative results, and a complete reference list.

P RESEARCH TOPICS SPECIFICATION

This section lists the **20 research topics** used in our evaluation. The topics span seven domains and cover a diverse set of task types, modalities, and evaluation setups. The complete list is given in Listing 1.


```
[
  {
    "question_id": "nlp_biomed_qa",
    "domain": "NLP",
    "difficulty": "incremental_innovation",
    "background": "Biomedical QA systems already perform reasonably
    well on PubMedQA, but lightweight improvements with clean ablations
    and reproducible training are still valuable.",
    "problem_statement": "Design a practical method for improving
    PubMedQA under limited compute while keeping the implementation easy
    to reproduce.",
    "baselines": ["BioBERT", "PubMedBERT", "instruction-tuned
    biomedical QA baseline"],
    "datasets": ["PubMedQA"],
    "user_requirements": "Generate a new idea and an
    implementation-oriented plan. Keep the method lightweight,
    ablatable, and reproducible.",
    "extra_context": "Prefer methods that fit within a modest
    single-node budget and can be compared fairly against standard
    biomedical QA baselines."
  },
  {
    "question_id": "nlp_short_text_cls",
    "domain": "NLP",
    "difficulty": "incremental_innovation",
    "background": "Short-text classification benchmarks are mature, but
    small improvements that reduce compute and keep the stack simple
    remain useful for reproducible evaluation.",
    "problem_statement": "Propose a lightweight method for improving
    short-text classification quality without introducing a heavy
    training pipeline.",
    "baselines": ["DistilBERT", "BERT-base", "linear bag-of-words
    classifier"],
    "datasets": ["AG News", "SST-2"],
    "user_requirements": "Produce a practical idea and an executable
    plan. Favor compact architectures, clean ablations, and fast
    iteration.",
    "extra_context": "The project should be feasible on a single GPU
    with small batch sizes and should avoid retrieval-heavy or
    multi-stage systems."
  },
  {
    "question_id": "nlp_sentence_pair_cls",
    "domain": "NLP",
    "difficulty": "incremental_innovation",
    "background": "Sentence-pair benchmarks are easy to fine-tune and
    compare, making them a good testbed for compact modeling ideas
    rather than large-scale engineering.",
    "problem_statement": "Design a lightweight method for improving
    sentence-pair classification or matching quality without adding a
    heavy retrieval or multi-stage stack.",
    "baselines": ["DistilBERT", "BERT-base", "Siamese bi-encoder
    baseline"],
    "datasets": ["MRPC", "RTE"],
```

```
    "user_requirements": "Return a practical research idea and an
    executable implementation plan. Favor compact modules, fair
    baselines, and short training cycles.",
    "extra_context": "The method should remain small enough for a
    single-GPU run and should allow clear ablations over standard
    sentence-pair baselines."
  },
  {
    "question_id": "cv_small_image_cls",
    "domain": "CV",
    "difficulty": "incremental_innovation",
    "background": "Small-image classification tasks are easy to run and
    compare, making them suitable for testing whether the agent can
    propose reproducible improvements rather than large-scale
    engineering tricks.",
    "problem_statement": "Design a lightweight image-classification
    method that improves small-image benchmarks without relying on
    oversized backbones or expensive pretraining.",
    "baselines": ["ResNet-18", "MobileNetV3-small", "ViT-tiny"],
    "datasets": ["CIFAR-10", "FashionMNIST"],
    "user_requirements": "Return a novel but practical method and a
    benchmarkable implementation plan. Keep the method compact,
    ablatable, and easy to train.",
    "extra_context": "Prefer methods that can finish a meaningful run
    quickly on a single GPU and use standard torchvision-style tooling."
  },
  {
    "question_id": "multimodal_efficiency",
    "domain": "Multimodal",
    "difficulty": "nontrivial_recomposition",
    "background": "Compact multimodal systems often trade off quality
    against latency and systems complexity, especially when evaluated
    under strict deployment budgets.",
    "problem_statement": "Propose a systems-aware multimodal method
    that improves benchmark quality without introducing an impractical
    training or serving stack.",
    "baselines": ["compact VLM baseline", "late-fusion multimodal
    baseline"],
    "datasets": ["MMMU", "ScienceQA"],
    "user_requirements": "Produce a novel idea, an executable plan, and
    a benchmarkable implementation path. Avoid overly fragile or heavy
    multi-stage designs.",
    "extra_context": "Reward methods with clear component interfaces,
    fair comparisons, and realistic implementation scope. Keep the stack
    compact enough for a single-GPU experiment."
  },
  {
    "question_id": "tabular_budgeted_cls",
    "domain": "Tabular ML",
    "difficulty": "incremental_innovation",
    "background": "Tabular classification remains a strong testbed for
    low-cost experimentation because datasets are small, baselines are
    well understood, and implementation cycles are fast.",
    "problem_statement": "Design a lightweight tabular-learning method
    that improves standard tabular baselines without relying on large
    ensembles or expensive feature engineering.",
    "baselines": ["XGBoost", "TabTransformer", "MLP baseline"],
    "datasets": ["Adult", "CoverType"],
    "user_requirements": "Generate an idea and an
    implementation-oriented plan that is simple, fast, and easy to
    compare against standard baselines.",
    "extra_context": "Favor methods that can be trained in a short
    wall-clock time, with clear ablations and no dependence on external
    retrieval or long preprocessing pipelines."
  },
```

```
  {
    "question_id": "tabular_regression",
    "domain": "Tabular ML",
    "difficulty": "incremental_innovation",
    "background": "Small tabular regression problems are cheap to run
    and easy to diagnose, so they are useful for testing whether the
    system can make focused improvements under tight resource budgets.",
    "problem_statement": "Propose a lightweight tabular-regression
    method that improves standard regression baselines without resorting
    to large ensembles or expensive feature engineering.",
    "baselines": ["XGBoost regressor", "CatBoost regressor", "MLP
    regressor baseline"],
    "datasets": ["California Housing", "Energy Efficiency"],
    "user_requirements": "Return a practical idea and a runnable
    implementation plan. Favor compact models, clear ablations, and fast
    turnaround.",
    "extra_context": "The project should be easy to implement with
    sklearn-style preprocessing or a small PyTorch model and should
    finish quickly on a single GPU or CPU-backed node."
  },
  {
    "question_id": "timeseries_sensor_cls",
    "domain": "Time Series",
    "difficulty": "incremental_innovation",
    "background": "Human activity recognition and other compact
    time-series tasks are lightweight enough for repeated
    experimentation while still requiring nontrivial modeling choices.",
    "problem_statement": "Propose a lightweight time-series
    classification method that improves compact sensor benchmarks
    without using a large or highly specialized model stack.",
    "baselines": ["1D CNN baseline", "GRU baseline",
    "InceptionTime-small"],
    "datasets": ["UCI HAR"],
    "user_requirements": "Return a practical method and a reproducible
    implementation plan. Keep the approach compact, interpretable, and
    easy to benchmark.",
    "extra_context": "Prefer methods that can be implemented with
    standard PyTorch components and evaluated quickly on a single GPU."
  },
  {
    "question_id": "graph_node_cls",
    "domain": "Graph ML",
    "difficulty": "incremental_innovation",
    "background": "Node classification on citation graphs is a
    lightweight setting for probing whether the system can propose
    meaningful graph-model refinements without large-scale
    infrastructure.",
    "problem_statement": "Design a lightweight graph-learning method
    that improves standard node-classification baselines while keeping
    the implementation simple and reproducible.",
    "baselines": ["GCN", "GraphSAGE", "GAT"],
    "datasets": ["Cora", "Citeseer"],
    "user_requirements": "Generate a new idea and an executable plan.
    Prefer simple message-passing modifications, clean ablations, and
    modest compute cost.",
    "extra_context": "Avoid large graph pretraining or multi-stage
    pipelines. The project should be runnable quickly with a standard
    single-GPU setup."
  },
  {
    "question_id": "audio_keyword_cls",
    "domain": "Audio",
    "difficulty": "incremental_innovation",
    "background": "Keyword spotting on short audio clips is a compact
    benchmark family that is cheap to train, easy to compare, and useful
```

```
    for testing whether the system can make practical
    efficiency-oriented improvements.",
    "problem_statement": "Design a lightweight audio-classification
    method that improves keyword spotting quality without depending on a
    large speech model or a complex preprocessing pipeline.",
    "baselines": ["CNN keyword-spotting baseline", "CRNN baseline",
    "tiny conformer baseline"],
    "datasets": ["SpeechCommands"],
    "user_requirements": "Return a practical idea and an executable
    plan. Keep the model compact, the preprocessing standard, and the
    training budget modest.",
    "extra_context": "Favor torchaudio-compatible pipelines and methods
    that can finish a meaningful comparison quickly on a single GPU."
  },
  {
    "question_id": "nlp_token_cls",
    "domain": "NLP",
    "difficulty": "incremental_innovation",
    "background": "Token-level sequence labeling tasks remain one of
    the cleanest settings for testing compact contextual modeling ideas
    and fair ablations under limited compute.",
    "problem_statement": "Design a lightweight token-classification
    method that improves standard named-entity recognition benchmarks
    without adding a large pipeline or external retrieval system.",
    "baselines": ["DistilBERT token classifier", "BERT-base token
    classifier", "BiLSTM-CRF baseline"],
    "datasets": ["CoNLL-2003", "WNUT17"],
    "user_requirements": "Return a practical idea and an executable
    implementation plan. Favor compact sequence modules, clean
    ablations, and short fine-tuning runs.",
    "extra_context": "The full project should run on a single GPU and
    should stay close to standard Hugging Face token-classification
    tooling."
  },
  {
    "question_id": "nlp_extractive_qa",
    "domain": "NLP",
    "difficulty": "incremental_innovation",
    "background": "Extractive QA is mature enough that strong baselines
    are available, but lightweight methods that improve calibration or
    answer localization remain useful and easy to benchmark.",
    "problem_statement": "Propose a lightweight extractive QA method
    that improves standard span-selection baselines without relying on
    retrieval-heavy or multi-stage systems.",
    "baselines": ["DistilBERT QA baseline", "BERT-base QA baseline",
    "RoBERTa-base QA baseline"],
    "datasets": ["SQuAD v1.1", "NewsQA"],
    "user_requirements": "Produce a practical method and an
    implementation-oriented plan. Keep the method compact, reproducible,
    and easy to compare against standard QA baselines.",
    "extra_context": "Prefer single-model approaches that can be
    trained on one GPU and evaluated with standard extractive QA
    metrics."
  },
  {
    "question_id": "cv_finegrained_cls",
    "domain": "CV",
    "difficulty": "incremental_innovation",
    "background": "Fine-grained visual classification is harder than
    small-image classification but still manageable on compact datasets,
    making it a good benchmark for lightweight representation
    improvements.",
    "problem_statement": "Design a lightweight fine-grained
    image-classification method that improves compact benchmarks without
    requiring a large pretrained vision backbone.",
```

```
    "baselines": ["ResNet-18", "EfficientNet-B0", "ViT-tiny"],
    "datasets": ["Oxford-IIIT Pets", "Flowers102"],
    "user_requirements": "Return a practical method and a benchmarkable
    implementation plan. Favor compact modules, fair baselines, and
    simple training code.",
    "extra_context": "The benchmark should remain feasible on a single
    GPU with standard torchvision or timm tooling and should support
    clear ablations."
  },
  {
    "question_id": "cv_multilabel_cls",
    "domain": "CV",
    "difficulty": "incremental_innovation",
    "background": "Compact multi-label image classification is a useful
    stress test for calibration and feature-sharing ideas while staying
    much cheaper than large-scale detection pipelines.",
    "problem_statement": "Propose a lightweight multi-label
    image-classification method that improves standard baselines without
    introducing a heavy detection or segmentation stack.",
    "baselines": ["ResNet-18 multi-label baseline", "MobileNetV3
    multi-label baseline", "ViT-tiny multi-label baseline"],
    "datasets": ["Pascal VOC 2007"],
    "user_requirements": "Generate a practical idea and an executable
    implementation plan. Keep the model small, the training pipeline
    standard, and the ablations clean.",
    "extra_context": "Prefer methods that can run with standard
    image-classification backbones and sigmoid multi-label heads on a
    single GPU."
  },
  {
    "question_id": "tabular_imbalance_cls",
    "domain": "Tabular ML",
    "difficulty": "incremental_innovation",
    "background": "Imbalanced tabular classification is common in real
    applications and is still easy to benchmark with compact models and
    short training loops.",
    "problem_statement": "Design a lightweight tabular method for
    improving imbalanced classification without relying on large
    ensembles, costly resampling pipelines, or heavy AutoML stacks.",
    "baselines": ["XGBoost", "LightGBM", "MLP baseline"],
    "datasets": ["Credit Card Fraud", "Telco Churn"],
    "user_requirements": "Return a practical method and a runnable
    plan. Favor compact architectures, fair imbalance-aware metrics, and
    fast experimentation.",
    "extra_context": "The project should be implementable with
    sklearn-style preprocessing and short training jobs on CPU or a
    single modest GPU."
  },
  {
    "question_id": "tabular_missing_value_cls",
    "domain": "Tabular ML",
    "difficulty": "incremental_innovation",
    "background": "Missing values are a realistic source of difficulty
    in tabular learning and provide a clean testbed for lightweight
    robustness ideas under short iteration cycles.",
    "problem_statement": "Propose a lightweight tabular-classification
    method that improves robustness to missing-value patterns without
    relying on expensive imputation ensembles or large stacked models.",
    "baselines": ["XGBoost", "TabTransformer", "MLP baseline with
    imputation"],
    "datasets": ["Adult", "Higgs Small"],
    "user_requirements": "Produce a practical idea and an executable
    implementation plan. Keep preprocessing simple, ablations clear, and
    compute modest.",
```

```
    "extra_context": "Prefer methods that can be implemented with
    simple masking or feature-gating ideas and benchmarked quickly on
    standard tabular datasets."
  },
  {
    "question_id": "timeseries_ecg_cls",
    "domain": "Time Series",
    "difficulty": "incremental_innovation",
    "background": "Compact ECG and UCR-style sequence benchmarks are
    cheap to run and make it easy to compare lightweight temporal
    architectures without specialized infrastructure.",
    "problem_statement": "Design a lightweight time-series
    classification method that improves compact ECG-style benchmarks
    without using a large transformer stack or custom hardware
    assumptions.",
    "baselines": ["1D CNN baseline", "GRU baseline",
    "InceptionTime-small"],
    "datasets": ["ECG200", "FordA"],
    "user_requirements": "Return a practical method and a reproducible
    implementation plan. Favor small temporal modules, interpretable
    ablations, and fast turnaround.",
    "extra_context": "The full experiment should remain compatible with
    plain PyTorch and run comfortably on a single GPU."
  },
  {
    "question_id": "graph_link_pred",
    "domain": "Graph ML",
    "difficulty": "incremental_innovation",
    "background": "Lightweight link prediction on citation or
    collaboration graphs is a compact graph-learning setting that still
    allows meaningful architectural comparison and ablation.",
    "problem_statement": "Propose a lightweight graph-learning method
    that improves standard link-prediction baselines while keeping the
    codebase simple and easy to reproduce.",
    "baselines": ["GCN encoder + dot-product decoder", "GraphSAGE
    encoder + MLP decoder", "GAT encoder baseline"],
    "datasets": ["Cora", "Citeseer"],
    "user_requirements": "Generate a new idea and an executable plan.
    Favor simple neighborhood or edge-scoring modifications, clean
    ablations, and modest compute cost.",
    "extra_context": "Avoid large graph pretraining and keep the
    project runnable on a single GPU with standard PyTorch Geometric
    tooling."
  },
  {
    "question_id": "audio_emotion_cls",
    "domain": "Audio",
    "difficulty": "incremental_innovation",
    "background": "Small audio emotion benchmarks are compact enough
    for repeated experimentation and are a useful testbed for
    lightweight temporal and spectral feature-learning ideas.",
    "problem_statement": "Design a lightweight audio-classification
    method that improves small emotion-recognition benchmarks without
    depending on a large speech foundation model.",
    "baselines": ["CNN spectrogram baseline", "CRNN baseline", "tiny
    conformer baseline"],
    "datasets": ["RAVDESS", "CREMA-D"],
    "user_requirements": "Return a practical idea and an executable
    plan. Keep preprocessing standard, the model compact, and the
    implementation reproducible.",
    "extra_context": "Favor torchaudio-compatible pipelines, short
    training cycles, and methods that can be compared fairly on a single
    GPU."
  },
  {
```

```
    "question_id": "multimodal_hateful_memes",
    "domain": "Multimodal",
    "difficulty": "nontrivial_recomposition",
    "background": "Compact multimodal classification tasks are useful
    for testing whether the system can improve cross-modal fusion
    quality without leaning on a very large vision-language model.",
    "problem_statement": "Propose a lightweight multimodal
    classification method that improves compact vision-language
    benchmarks without introducing a heavy multi-stage or large-model
    serving stack.",
    "baselines": ["late-fusion multimodal baseline", "CLIP linear-probe
    baseline", "compact VLM baseline"],
    "datasets": ["Hateful Memes"],
    "user_requirements": "Produce a novel idea, an executable plan, and
    a benchmarkable implementation path. Keep the model compact, the
    fusion strategy interpretable, and the evaluation fair.",
    "extra_context": "Reward methods with clean image-text interfaces
    and realistic single-GPU training requirements."
  }
]
```

Listing 1: Full JSON specification of the 20 research topics used in our evaluation.

Q Limitations and Broader Impact

Limitations. The controlled 20-task benchmark uses Claude to generate tasks and simulated feedback, enabling broad, repeatable coverage while retaining a model-mediated evaluation loop. We address this directly with ten human-authored tasks, matched baselines, and independent blinded reviews. The current human study covers ten researchers and AI/ML problems and directly validates real-user adaptation; larger cohorts, held-out tasks from returning users, and fields requiring physical experiments are the next evaluation scale. Round 1–Round 3 measure within-task adaptation, while cross-project transfer is a distinct follow-up setting. The component ablations establish the contribution of the three-layer design, and Planner+SDPO versus Planner Only isolates a parameter-update gain with Skill and Memory retrieval absent. An exact same-feedback frozen-planner control would further separate persistent learning from transient in-context use. The system depends on closed model APIs, whose version changes can affect reproducibility despite snapshot logging. Finally, persistent user memory creates privacy, retention, and cross-user contamination risks that require explicit access control and deletion policies.

Broader Impact. NanoResearch can reduce the effort required to plan, execute, and document computational research, particularly when user constraints are preserved across iterations. The same automation can also produce plausible but incorrect, weakly supported, or misleading scientific artifacts at scale. Outputs should therefore be treated as research drafts: dataset provenance, code, statistical analysis, citations, and claims require human verification before dissemination. Deployments should isolate user-specific adapters and memories, minimize retained personal data, support deletion and audit logs, and never allow one user's private records to enter another user's context. Dependence on large proprietary models also creates cost, accessibility, and concentration-of-control concerns.