Meta-Trial Agent: A Self-Evolving Framework for Solution Discovery in Clinical Trial Tasks

Anonymous ACL submission

Abstract

Predicting clinical trial outcomes has attracted growing interest, yet existing approaches demand substantial human effort and domain expertise for each new task, whether they rely on deep learning systems with task-specific architectural design or agentic methods built on static, manually designed roles. We introduce Meta-Trial Agent, a self-evolving agentic framework that iteratively searches for solutions to clinical trial prediction tasks within an expressive space encompassing programs, agents, or hybrid compositions of both. A persistent memory accumulates experience across evolution rounds, while a Reflection Agent provides structured feedback when signs of stagnation are detected. We evaluate Meta-Trial Agent on five tasks from the TrialBench dataset against state-of-the-art clinical trial prediction methods and five evolutionary search frameworks. On trial approval prediction, the discovered solution outperforms the best clinical trial baseline by an average of over 0.25 PR-AUC and the best search framework by over 0.03 PR-AUC across phases, with consistent gains across all tasks, phases, and metrics. Furthermore, qualitative analysis shows that, unlike program-evolving baselines that converge to complex pipelines built on surface-level features, Meta-Trial Agent discovers hybrid solutions that combine simpler modeling components with agents grounded in external biomedical knowledge.

1 Introduction

Clinical trials are essential for establishing the safety and efficacy of new treatments, yet they represent one of the most demanding stages of drug development. Bringing a new drug to market takes on average over ten years and costs nearly 2.6 billion dollars, with overall clinical success rates remaining below 10% (Qian et al., 2025). Trials must progress through multiple phases (DiMasi et al., 2016), each requiring complex, interdependent design decisions such as patient selection, dosing regimens, endpoint choice, and eligibility criteria (Getz and Campo, 2017). Even seemingly minor protocol flaws can lead to enrollment failure, safety events, or inconclusive results (Yamaguchi et al., 2021; Baumfeld Andre et al., 2020). While insufficient efficacy and safety concerns account for the majority of failures, a substantial fraction are attributable to avoidable operational and design shortcomings (Dagenais et al., 2022). Given the enormous human and financial cost of these failures, developing computational tools that can reliably predict trial outcomes and support protocol design has become an increasingly active area of research.

Early work applied classical machine learning to expert-curated tabular features, such as chemical properties, drug targets, and trial-level statistics (Gayvert et al., 2016; Lo et al., 2019; Siah et al., 2021). While effective, these methods could not incorporate unstructured or heterogeneous biomedical data. Deep learning approaches addressed this limitation by jointly modeling diverse modalities, including drug molecules, disease ontologies, and trial protocols (Fu et al., 2022; Wang et al., 2023; Zheng et al., 2025). These methods achieved progressively stronger predictive accuracy, but they share a fundamental constraint: each new task demands substantial human effort and domain expertise to design appropriate features, select model architectures, engineer data representations, and tune training procedures. When the task changes, much of this design work must be repeated from scratch, making these approaches difficult to scale across the diverse landscape of clinical trial prediction problems. The emergence of large language models (LLMs) has opened a new direction by enabling systems that introduce reasoning into clinical trial prediction (Yue et al., 2024; Liu et al., 2025). However, existing agentic approaches either rely on manually designed, static agent architec-

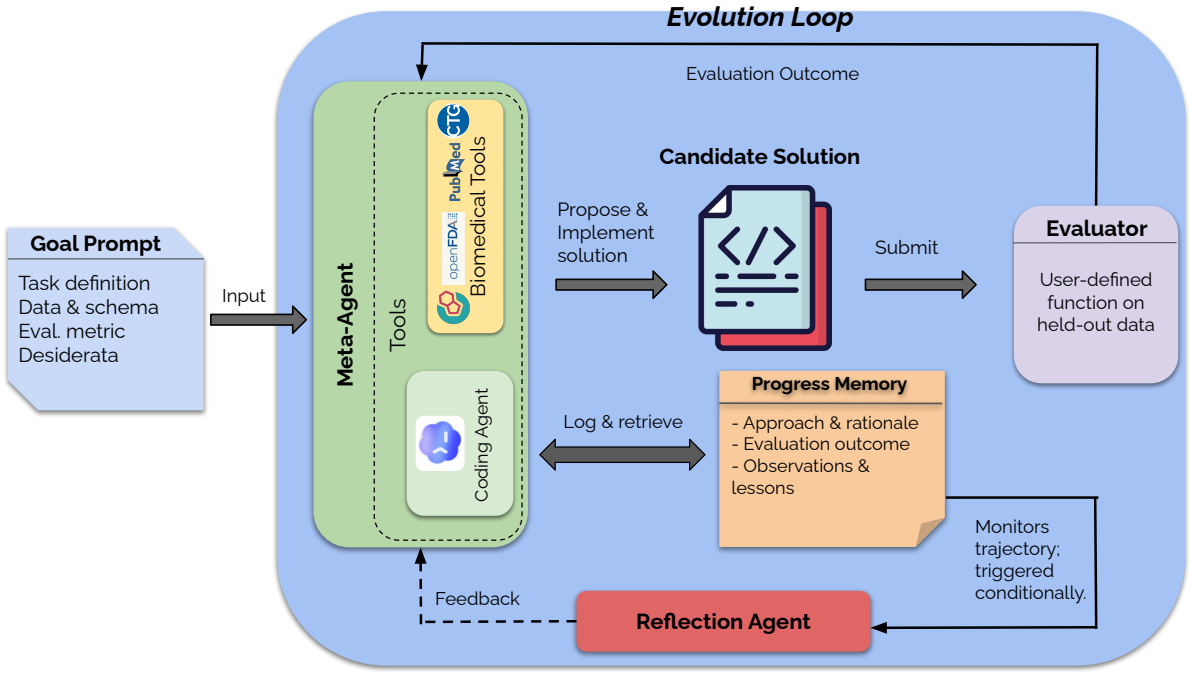


Figure 1: Overview of the Meta-Trial Agent framework. The Meta-Agent proposes candidate solutions. Each candidate is scored by the Evaluator, which returns the outcome to the Meta-Agent. The Meta-Agent then logs each round in Progress Memory, recording the approach and its rationale, the evaluation outcome, and observations for future rounds. The Reflection Agent provides feedback when triggered by conditions such as regression or plateau.

tures that require domain expertise to orchestrate (Yue et al., 2024), or adopt an agentic automated machine learning (AutoML) workflow that confines optimization to the feature space alone while leaving the underlying predictive model and solution structure fixed (Liu et al., 2025).

To fill this gap, we introduce Meta-Trial Agent, a self-evolving agentic framework that evolves solutions for clinical trial prediction tasks. Meta-Trial Agent is motivated by several converging insights. First, searching for the algorithm that solves a problem has proven more effective than searching for the solution directly (Romera-Paredes et al., 2024), and the history of machine learning shows that hand-designed artifacts are replaced by learned alternatives as compute and data scale (Clune, 2019; Sutton, 2019). Second, search algorithms benefit from more expressive search spaces, and it has been shown that richer representations lead to finding better solutions (Gopal et al., 2023; Hu et al., 2025; Novikov et al., 2025). Third, rapid progress in LLM-based autonomy has demonstrated that agents can now carry out complex, open-ended scientific discovery with increasing reliability (Wang et al., 2024; Lu et al., 2024).

Meta-Trial Agent instantiates these principles for clinical trial prediction. The framework consists of a Meta-Agent that iteratively proposes candidate solutions, which may take the form of programs, agent systems with or without tool access, or hybrid compositions of both, all expressed in code, and evaluates them against task-specific objectives. A Reflection Agent analyzes proposed solutions and their results and returns structured feedback to the Meta-Agent. The accumulated experience across rounds is stored in a persistent memory. This growing repository of past attempts, outcomes, and insights enables the Meta-Agent to make increasingly informed proposals in subsequent iterations.

We evaluate Meta-Trial Agent on multiple clinical trial tasks. Our experiments across clinical trial tasks show that Meta-Trial Agent finds superior solutions, where on clinical trial approval prediction the searched solution outperforms state-of-the-art methods by over 0.25 points in PR-AUC across phases. Moreover, compared with existing search frameworks, Meta-Trial Agent converges to a higher-performance solution that is also simpler and more generalizable.

The contributions of this work are summarized as follows:

- We introduce Meta-Trial Agent, a simple and generalizable framework for solution discovery in clinical trial tasks. To our knowledge,

this is the first work to formulate solution discovery in this domain.

- We conduct extensive experiments across five clinical trial tasks, demonstrating that our framework achieves superior performance over both human-designed state-of-the-art methods and existing search frameworks, outperforming the best clinical trial baseline by an average of 0.25 PR-AUC and the best search framework by an average of 0.03 PR-AUC across phases on the trial approval prediction task.
- Through qualitative analysis, we show that, compared to existing search frameworks, our framework converges to solutions with simpler modeling pipelines and clinically sound features grounded in external biomedical evidence rather than surface-level engineered features.

2 Related work

2.1 Clinical Trial Outcome Prediction

Early approaches to clinical trial outcome prediction employed classical machine learning on expert-curated tabular features (Gayvert et al., 2016; Lo et al., 2019; Siah et al., 2021). Subsequent deep learning methods addressed these representational limitations by jointly modeling heterogeneous data modalities, including drug molecules, disease ontologies, and trial protocols (Fu et al., 2022; Wang et al., 2023; Zheng et al., 2025). While these systems achieved substantially stronger predictive accuracy, they require significant human effort to design features, architectures, and training procedures for each new task. More recently, LLM-based agentic methods have introduced reasoning and iterative refinement into clinical trial workflows. ClinicalAgent (Yue et al., 2024) decomposes trial prediction into subtasks handled by specialized agents coordinated through LLM-based planning, though these agents are manually designed and remain static throughout prediction. AutoCT (Liu et al., 2025) uses LLM agents to autonomously generate and refine tabular features via Monte Carlo Tree Search, but the underlying predictive model and solution structure remain fixed, confining optimization to the feature space alone. While these agentic approaches represent progress over static pipelines, they require domain knowledge to design agent roles and orchestration, or their search space is limited to the feature space, leaving the model space unexplored.

2.2 Evolutionary Search Methods

A parallel line of research has explored using LLMs as engines for open-ended program and agentic system search. AlphaEvolve (Novikov et al., 2025) demonstrated that LLM-guided evolutionary search over entire codebases can discover novel algorithms across computer science and mathematics, while ShinkaEvolve (Lange et al., 2025) improved sample efficiency through innovations in parent sampling, novelty-based rejection, and bandit-based LLM ensemble selection. AIDE (Jiang et al., 2025) framed machine learning engineering as code-space optimization, using tree search to draft, debug, and refine solutions. GEPA (Agrawal et al., 2025) introduced reflective prompt evolution, using natural language feedback to iteratively optimize any textual parameter within compound AI systems. However, these systems confine their search space to program or prompt optimization, evolving artifacts for a single, well-defined objective function. Beyond program-level optimization, ADAS (Hu et al., 2025) formulated the automated design of agentic systems as a meta-search problem, where a meta-agent iteratively programs new agents. Yet across all these systems, the search process itself remains static: parent selection, prompt construction, and mutation are governed by predefined heuristics, with the LLM serving only as a generation engine rather than an autonomous decision-maker over the direction of search. Moreover, these systems restrict what can be expressed within the code they evolve, either standalone programs or agent architectures, but not both. Furthermore, none of them has been applied to domains requiring heterogeneous multimodal data and diverse task formulations, such as clinical trial tasks. Meta-Trial Agent aims to bridge these two lines of work by bringing self-evolving solution discovery to clinical trial tasks.

3 Method

We introduce Meta-Trial Agent, a self-evolving agentic framework in which a Meta-Agent iteratively evolves candidate solutions toward a user-defined goal. The framework comprises five components: a goal prompt that defines the task, data, and optimization objective; a Meta-Agent that autonomously proposes, implements, and refines candidate solutions using a set of tools; a memory mod-

ule that accumulates experiences across rounds; a Reflection Agent that analyzes the solution trajectory and provides structured feedback under specific conditions; and an evaluator that scores each candidate solution against the target metric. An overview of the framework is shown in Figure 1.

3.1 Goal Prompt

The goal prompt defines the task and optimization objective in natural language. It specifies the task itself (e.g., trial approval prediction), the location and schema of the data, and the evaluation metric to optimize. It also communicates broader desiderata such as generalization and clinical soundness. Additional task-specific constraints or background knowledge can be included as needed. Crucially, the goal prompt remains open-ended and avoids prescribing any particular modeling strategy, allowing the Meta-Agent to freely explore the solution space. The goal prompt is fixed throughout the run and included in the Meta-Agent's context at every round. An example of a goal prompt is provided in Appendix A.2.1.

3.2 Meta-Agent

The Meta-Agent is the central component of the framework: an autonomous agent with full decision-making authority over solution construction. Given a task, it may pursue one of three broad strategies: implementing the solution as a self-contained program, defining specialized agents equipped with retrieval tools from the biomedical knowledge pool, or composing a hybrid that combines both, all expressed in code. The Meta-Agent is further equipped with a sandbox environment for code execution and file operations, allowing it to operate end-to-end without external intervention. Each evolution round constitutes a single attempt to improve upon the current best solution. The Meta-Agent synthesizes its working context from the goal prompt, accumulated memory, the history of prior rounds with their evaluation outcomes, and any structured insights from the Reflection Agent. It then deliberates on the next direction, designs and implements a complete candidate solution, which is subsequently scored by the evaluator. The approach taken, the outcome, and the Meta-Agent's observations are recorded in memory for future rounds. The Meta-Agent instruction prompt is provided in Appendix A.2.2.

3.2.1 Biomedical Tools

To ground the framework in biomedical domain knowledge, we curate a set of retrieval tools connected to authoritative resources. Each tool is made accessible through the Model Context Protocol (MCP). The current pool spans four complementary sources: PubMed for biomedical literature, ClinicalTrials.gov for registered trial records, OpenFDA for drug approvals, labeling, and adverse event reports, and ChEMBL for bioactive molecules and their biological targets. Together, these tools cover the major dimensions of evidence relevant to clinical trial reasoning. Crucially, irrelevant information is dynamically filtered from retrieval outputs, while strict temporal constraints are enforced across all queries to prevent outcome leakage.

3.2.2 Coding Agent

The Coding Agent translates high-level task descriptions into executable code. When the Meta-Agent determines that part of a solution requires code, it delegates the implementation to the Coding Agent along with the relevant context; the Coding Agent produces the code and returns it to the Meta-Agent, which may inspect it, request revisions, or execute it in the sandbox. This division of labor allows the Meta-Agent to reason at a strategic level while the Coding Agent focuses on producing correct, runnable code. All generated code is preserved as part of the run record.

3.3 Memory

At the conclusion of each round, the Meta-Agent records an entry summarizing what was attempted, why it was chosen, how it performed, and what was learned. Each entry comprises a description of the candidate solution and its rationale, the evaluation outcome with an indication of whether it improved upon the best result so far, and observations the Meta-Agent considers important for future rounds, such as recurring failure modes, promising directions, or hypotheses to test next. The accumulated memory is made available at the beginning of every subsequent round, enabling the Meta-Agent to recognize when a line of work has plateaued, avoid revisiting failed approaches, and build on promising directions.

3.4 Reflection Agent

The Reflection Agent monitors the evolutionary trajectory and intervenes when the Meta-Agent shows signs of inefficient exploration. It is engaged only

when one of four conditions is observed: a *failure*, when the previous round produced no valid evaluation; a *regression*, when the most recent metric falls substantially below the best result; a *plateau*, when the metric remains flat over consecutive rounds; or a *staleness* fallback, ensuring periodic checks during long stretches of steady progress. When invoked, the Reflection Agent reviews the trajectory of prior rounds, recent code artifacts, and the narrative summaries the Meta-Agent has recorded in memory, and produces a structured analysis covering several complementary dimensions: solution performance assessments with recommended actions, redundancies and capability gaps in the solution candidate, targeted ablation proposals, and a prioritized list of recommendations. This guidance is provided to the Meta-Agent as non-prescriptive input, preserving its autonomy while steering exploration when the search shows signs of stagnation. The Reflection Agent instruction prompt is provided in Appendix A.2.3.

3.5 Evaluator

The evaluator measures how well each candidate solution performs against the user-defined objective using an evaluation function provided by the user, keeping the framework agnostic to any specific task or metric. It receives the candidate solution together with held-out data, computes the relevant metrics, and reports a designated primary metric that the framework tracks across rounds. This design enforces a separation between solution construction and performance measurement: the Meta-Agent never has access to the held-out labels and interacts with the evaluator solely by receiving scores, eliminating the possibility of label leakage during the search.

4 Experiments

We evaluate our framework along two lines of comparison. First, we compare the performance of discovered solutions against state-of-the-art methods for clinical trial prediction across five tasks from the TrialBench dataset (Chen et al., 2025), demonstrating the generalizability of our framework across diverse task formulations. Second, we compare our framework against state-of-the-art evolutionary search methods on the same five tasks, assessing how the solutions discovered by our framework differ from and perform relative to those found by existing search procedures.

4.1 Experimental Setup

Dataset. We use the TrialBench dataset (Chen et al., 2025), a comprehensive benchmark curated from ClinicalTrials.gov, DrugBank, and TrialTrove, covering several clinical trial tasks with multimodal input features including drug SMILES strings, disease ICD-10 codes, MeSH terms, textual descriptions, and categorical/numerical features. We select five predictive tasks: trial approval prediction, patient dropout prediction, adverse event prediction, mortality event prediction, and trial failure reason identification. We follow the original train/test split across all four phases. We further split the training set into 80% training and 20% validation subsets, where the validation set guides the search process and the test set is used solely for final evaluation. Following prior studies (Fu et al., 2022; Liu et al., 2025), we report PR-AUC, F1 score, and ROC-AUC across phases.

Setup. We use GPT-5 (Singh et al., 2025) as the base model for the Meta-Agent and Reflection Agent, GPT-5.1-Codex as the Coding Agent, and GPT-4o (Hurst et al., 2024) for agents created by the Meta-Agent. Since our framework and each search baseline differ in their iteration structure and evaluation schema, we adopt a cost-controlled budget design: all search frameworks are run until a fixed budget of 10 USD is exhausted. For fair comparison, we use GPT-5 as the search model across all frameworks and GPT-4o for agents created within ADAS, consistent with our own configuration. We use mean PR-AUC across phases of a given task as the primary optimization objective for all frameworks.

4.2 Baselines

Clinical Trial Baselines. We compare the discovered solutions against both classical machine learning (ML) models and state-of-the-art deep learning methods. For classical approaches, we include Logistic Regression (Siah et al., 2021; Lo et al., 2019) and XGBoost (Siah et al., 2021). For deep learning, we compare against HINT (Fu et al., 2022) and the multimodal fusion method introduced in the TrialBench (Chen et al., 2025).

Search Baselines. We compare our framework against five search baselines spanning two categories. For program-evolving search, we use ShinkaEvolve (Lange et al., 2025), AIDE (Jiang et al., 2025), OpenEvolve (Sharma, 2025), and

GEPA (Agrawal et al., 2025). GEPA is primarily designed for prompt optimization but can be applied to any textual parameter; we adapt it to program evolution by setting the evolving artifact to a program rather than a prompt string. For agent-level search, we use ADAS (Hu et al., 2025). More detailed descriptions of baselines and experiments are provided in Appendix A.3.

4.3 Results

4.3.1 Comparison with Clinical Trial Baselines

Table 1 reports the performance of the searched solution and clinical trial baselines on the trial approval prediction task. Deep learning methods, HINT and TrialBench's multimodal fusion, consistently outperform classical ML baselines, reflecting the value of modeling heterogeneous modalities. However, the solution discovered by Meta-Trial Agent substantially outperforms all baselines across every metric and phase, with PR-AUC improvements of over 25 absolute points compared to the strongest baseline in most phases. Notably, the discovered solution is considerably simpler than HINT and multimodal fusion, both of which rely on carefully engineered architectures and domain-specific design choices; we discuss the nature of the discovered solution in detail in Section 4.5.

To assess the generalizability of our framework beyond a single task, we repeat the experiments across four additional clinical trial tasks. Results are reported in Table 3 in the Appendix. Meta-Trial Agent's discovered solutions consistently achieve superior performance across all tasks. The gains are particularly pronounced for mortality event prediction, where PR-AUC improves by over 22 points compared to the best baseline, and for adverse event prediction, where the discovered solution reaches PR-AUC above 0.94 in Phases 1–3.

4.3.2 Comparison with Search Baselines

Table 2 reports the performance of solutions discovered by different search frameworks on the trial approval prediction task. Among the baselines, ADAS performs notably worse than the program-evolving methods, suggesting that searching solely over agent architectures without the ability to evolve computational pipelines is insufficient for this domain. The four program-evolving baselines, OpenEvolve, GEPA, ShinkaEvolve, and AIDE, achieve broadly similar results, with ShinkaEvolve holding a slight edge in PR-AUC across most phases. The solution discovered by Meta-Trial Agent outperforms all baselines across all metrics and phases, achieving approximately 2 to 5 points higher PR-AUC than ShinkaEvolve, with even larger margins on F1 and ROC-AUC. The gap is most notable in Phase 2, where our framework achieves a ROC-AUC of 0.887 against 0.850 for the strongest baseline.

We extend the comparison to four additional tasks in Table 4 in the Appendix. The pattern from trial approval prediction largely holds: ADAS consistently underperforms, while the program-evolving baselines are competitive. Meta-Trial Agent achieves the best or near-best performance across the majority of task-phase-metric combinations. The improvements are most pronounced on mortality event prediction, where our framework achieves PR-AUC of 0.938 in Phase 2 compared to 0.896 for the strongest baseline, and on trial failure reason identification, where our framework leads by over 7 points in PR-AUC in Phases 1 to 3, a substantial margin on this challenging multi-class task. For adverse event prediction, the margins are narrower, as all program-evolving baselines already achieve strong performance, yet our framework still achieves the highest F1 scores across all phases.

4.4 Ablation Study

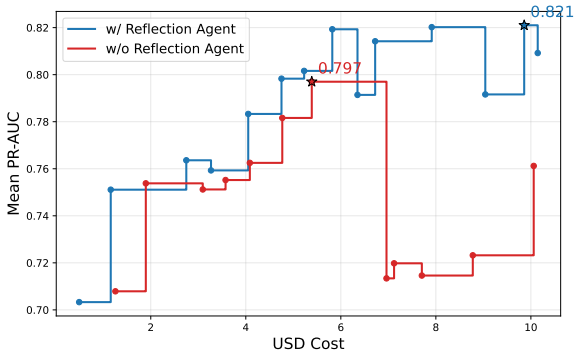


Figure 2: Search trajectory with and without Reflection Agent, plotted as mean PR-AUC against cumulative USD cost.

To isolate the effect of the Reflection Agent on the search process and its role in guiding the framework toward simpler and more generalizable solutions, we conduct an ablation study removing it while keeping all other components unchanged. Figure 2 shows the search trajectory for both runs under the same $10 budget. Both runs climb similarly through the first seven rounds, reaching comparable peaks (0.79 without reflection, 0.80 with

Table 1: Clinical trial approval prediction results across phases.

	Phase 1			Phase 2			Phase 3			Phase 4		
Method	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC
LR	0.508	0.258	0.590	0.440	0.199	0.579	0.649	0.700	0.614	0.453	0.339	0.578
XGBoost	0.503	0.213	0.577	0.436	0.172	0.575	0.672	0.707	0.636	0.464	0.331	0.583
HINT	0.535	0.381	0.606	0.496	0.372	0.616	0.707	0.649	0.683	0.539	0.564	0.646
TrialBench	0.519	0.517	0.765	0.494	0.541	0.736	0.667	0.672	0.730	0.440	0.580	0.605
Ours	**0.805**	**0.805**	**0.865**	**0.762**	**0.753**	**0.887**	**0.917**	**0.840**	**0.898**	**0.813**	**0.765**	**0.901**

Table 2: Search frameworks results for clinical trial approval prediction.

	Phase 1			Phase 2			Phase 3			Phase 4		
Method	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC
ADAS	0.438	0.558	0.510	0.379	0.534	0.513	0.584	0.715	0.521	0.448	0.574	0.536
Openevolve	0.759	0.745	0.842	0.713	0.685	0.839	0.873	0.793	0.854	0.753	0.727	0.855
GEPA	0.751	0.727	0.839	0.742	0.683	0.841	0.872	0.803	0.855	0.762	0.729	0.851
ShinkaEvolve	0.785	0.748	0.854	0.734	0.693	0.840	0.882	0.801	0.864	0.766	0.721	0.858
AIDE	0.767	0.736	0.843	0.740	0.708	0.850	0.876	0.805	0.860	0.759	0.723	0.860
Ours	**0.805**	**0.805**	**0.865**	**0.762**	**0.753**	**0.887**	**0.917**	**0.840**	**0.898**	**0.813**	**0.765**	**0.901**

reflection), then diverge sharply. Without reflection, the Meta-Agent drops at round 8 to 0.713 and never recovers, remaining stuck in the low-0.72 range for the remainder of the budget. With reflection, the trajectory continues climbing to a best of 0.821 at round 13, with only shallow dips (-0.028 and -0.029) recovered the very next round. The round-8 regression of 0.084 far exceeds the regression threshold of 0.03, which would have triggered a recovery directive at round 9. The Reflection Agent's value thus lies not in peak performance but in regression recovery and downside protection, transforming the trajectory from a climb-then-cliff pattern into a sustained upward climb. We also perform a qualitative analysis of the discovered solutions under both settings. Without the Reflection Agent, the Meta-Agent converges to a more complex modeling pipeline: three text models, a hashing vectorizer with over 200k features, simplex weight search, reciprocal rank fusion, and conditional isotonic calibration. It creates three agents: Schema Analyst and Text Modeler, which are responsible for designing feature and modeling plans from their internal knowledge, and evidence miner, which queries PubMed, ClinicalTrials.gov, and OpenFDA to gather evidence linking pre-registration design features to trial success. The result is a solution that prioritizes model complexity over external knowledge enrichment. With the Reflection Agent, the solution takes the opposite approach: the modeling pipeline is simpler (one TF-IDF logistic regression and one HistGradientBoosting head with softmax-weighted blending), while three of the four agents actively query external biomedical databases — Clinical Priors validates design indicators against PubMed and ClinicalTrials.gov, Sponsor Tierer cross-checks sponsor capacity against OpenFDA, and Drug Linker queries ChEMBL to inject molecule-level regulatory priors such as ATC classification. The Reflection Agent appears to have steered the search away from pipeline complexity toward broader external knowledge acquisition, resulting in a simpler yet better-performing solution.

4.5 Qualitative Comparison of Searched Solutions

To better understand the qualitative differences between search frameworks, we examine the solutions discovered for the trial approval prediction task. The program-evolving baselines all converge to variations of the same general recipe: gradient-boosted trees over tabular features combined with bag-of-words text representations, differing mainly in the choice of vectorizer and the complexity of the surrounding ensemble. OpenEvolve produces the simplest variant, a three-seed LightGBM ensemble with 512-dimension feature hashing and crude SMILES character counts as a substitute for molecular descriptors. ShinkaEvolve discovers the most elaborate, a multi-stream stacked ensemble with four hashing vectorizer configurations at 2^{18} dimensions each, isotonic calibration, and a logistic regression meta-blender. AIDE falls in between with a multi-seed LightGBM bag blended with a text-only SGD classifier over hashed features. GEPA

produces a three-model blend of LightGBM, text-only logistic regression over TF-IDF features, and SMILES logistic regression, with blend weights estimated from a single stratified holdout, though its hand-crafted text features consist of formatting artifacts such as text lengths, digit counts, and bullet counts. Despite their differences in complexity, all four converge to solutions characterized by elaborate modeling pipelines built on surface-level feature engineering and bag-of-words text representations that discard vocabulary structure, with no mechanism to move beyond the statistical patterns available in the training data. ADAS produces an entirely different kind of solution: a multi-agent debate pipeline with a structural prior estimator, a risk-of-bias rater, a design auditor, and a five-role specialist panel that debate over two rounds before a meta-analyst fuses their outputs. Because it reasons directly over raw trial records without any training procedure, it operates as a zero-shot system with no ability to learn dataset-specific patterns, explaining its substantially lower performance. In contrast, Meta-Trial Agent discovers a hybrid solution that composes a multi-agent system with a modeling pipeline. The architecture of this solution is illustrated in Figure 6. Four agents are created: a clinical priors agent supplies domain-grounded design and endpoint indicators validated against PubMed and ClinicalTrials.gov; a sponsor tierer classifies sponsors into capacity tiers cross-checked against the OpenFDA database; a drug linker queries ChEMBL to build molecule-level regulatory priors including ATC classification and biologic status; and a data architect routes columns into semantic groups. The first three agents gather external biomedical knowledge, while the fourth performs reasoning over the dataset structure. This domain-augmented information feeds a two-head late-fusion pipeline: a TF-IDF logistic regression text head and a HistGradientBoosting tabular head, blended with data-driven weights from a stratified holdout. A detailed description is provided in Appendix A.4. The modeling components are notably simpler than the baselines, with no multi-stream stacking, no high-dimensional hashing, and no meta-blending. To further understand the predictions of the discovered solution, we perform a SHAP (SHapley Additive exPlanations) (Lundberg and Lee, 2017) analysis. As shown in Appendix A.5, we present selected correctly classified test examples whose SHAP values highlight the contribution of individual features to the predicted outcome. Despite this simplicity, the discovered solution outperforms all baselines, suggesting that grounding features in external domain knowledge and combining them with dataset-specific learning is a more effective strategy than scaling up surface-level feature engineering. This strategy is enabled by a more expressive code space that can represent both computational pipelines and agent compositions with access to external biomedical knowledge, rather than restricting what can be expressed to standalone programs or agent architectures alone.

5 Conclusion

In this work, we introduced Meta-Trial Agent, a self-evolving agentic framework that redefines solution discovery for clinical trial prediction tasks. By moving beyond rigid feature spaces and static, human-orchestrated agent roles, our approach leverages an open-ended code space to dynamically compose both computational pipelines and tool-using agents. Empirical evaluations across five tasks from the TrialBench dataset demonstrate that Meta-Trial Agent consistently outperforms state-of-the-art human-designed deep learning baselines and evolutionary search frameworks across all phases and metrics. Qualitative and ablation analyses highlight that the framework's success is driven by its hybrid nature, combining simple modeling heads with deep external biomedical evidence retrieval, and steered effectively by a Reflection Agent that safeguards against stagnation and pipeline over-engineering. Meta-Trial Agent shifts the paradigm from engineering static predictive models to building self-evolving, domain-grounded scientific discovery systems. Future research will explore how the framework scales under extended search horizons and other clinical tasks, and investigate transitioning to open-source foundation models to lower execution costs and make self-evolving clinical discovery more broadly accessible.

6 Limitations

While Meta-Trial Agent demonstrates consistent performance gains, several limitations remain. First, all experiments are conducted under a controlled cost budget; although this ensures fair comparison across frameworks, the system's behavior under substantially larger budgets or longer search horizons remains unexplored. Second, the framework relies on proprietary foundation models, introducing a dependency on commercial API availabil-

ity and incurring non-trivial execution costs that may limit accessibility. Third, the framework's performance may be sensitive to the design of the goal prompt, and we do not systematically study how variations in prompt formulation affect the discovered solutions.

References

Lakshya A Agrawal, Shangyin Tan, Dilara Soylu, Noah Ziems, Rishi Khare, Krista Opsahl-Ong, Arnav Singhvi, Herumb Shandilya, Michael J Ryan, Meng Jiang, and 1 others. 2025. Gepa: Reflective prompt evolution can outperform reinforcement learning. *arXiv preprint arXiv:2507.19457*.

Elodie Baumfeld Andre, Robert Reynolds, Patrick Caubel, Laurent Azoulay, and Nancy A Dreyer. 2020. Trial designs using real-world data: the changing landscape of the regulatory approval process. *Pharmacoepidemiology and drug safety*, 29(10):1201–1212.

Jintai Chen, Yaojun Hu, Mingchen Cai, Yingzhou Lu, Yue Wang, Xu Cao, Miao Lin, Hongxia Xu, Jian Wu, Xiao Cao, and 1 others. 2025. Trialbench: Multimodal ai-ready datasets for clinical trial prediction. *Scientific Data*, 12(1):1564.

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

Simon Dagenais, Leo Russo, Ann Madsen, Jen Webster, and Lauren Becnel. 2022. Use of real-world evidence to drive drug development strategy and inform clinical trial design. *Clinical Pharmacology & Therapeutics*, 111(1):77–89.

Joseph A DiMasi, Henry G Grabowski, and Ronald W Hansen. 2016. Innovation in the pharmaceutical industry: new estimates of r&d costs. *Journal of health economics*, 47:20–33.

Tianfan Fu, Kexin Huang, Cao Xiao, Lucas M Glass, and Jimeng Sun. 2022. Hint: Hierarchical interaction network for clinical-trial-outcome predictions. *Patterns*, 3(4).

Kaitlyn M Gayvert, Neel S Madhukar, and Olivier Elemento. 2016. A data-driven approach to predicting successes and failures of clinical trials. *Cell chemical biology*, 23(10):1294–1301.

Kenneth A Getz and Rafael A Campo. 2017. Trial watch: trends in clinical trial design complexity. *Nature Reviews Drug Discovery*, 16(5):307–308.

Bhavna Gopal, Arjun Sridhar, Tunhou Zhang, and Yiran Chen. 2023. Lissnas: Locality-based iterative search space shrinkage for neural architecture search. *arXiv preprint arXiv:2307.03110*.

Shengran Hu, Cong Lu, and Jeff Clune. 2025. Automated design of agentic systems. In *International Conference on Learning Representations*, volume 2025, pages 21344–21377.

Aaron Hurst, Adam Lerer, Adam P Goucher, Adam Perelman, Aditya Ramesh, Aidan Clark, AJ Ostrow, Akila Welihinda, Alan Hayes, Alec Radford, and 1 others. 2024. Gpt-4o system card. *arXiv preprint arXiv:2410.21276*.

Zhengyao Jiang, Dominik Schmidt, Dhruv Srikanth, Dixing Xu, Ian Kaplan, Deniss Jacenko, and Yuxiang Wu. 2025. Aide: Ai-driven exploration in the space of code. *arXiv preprint arXiv:2502.13138*.

Robert Tjarko Lange, Yuki Imajuku, and Edoardo Cetin. 2025. Shinkaevolve: Towards open-ended and sample-efficient program evolution. *arXiv preprint arXiv:2509.19349*.

Fengze Liu, Haoyu Wang, Joonhyuk Cho, Dan Roth, and Andrew Lo. 2025. Autoct: Automating interpretable clinical trial prediction with llm agents. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 30933–30958.

Andrew W Lo, Kien Wei Siah, and Chi Heem Wong. 2019. Issue 1 (machine learning with statistical imputation for predicting drug approval).

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

Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems*, 30.

Alexander Novikov, Ngân Vũ, Marvin Eisenberger, Emilien Dupont, Po-Sen Huang, Adam Zsolt Wagner, Sergey Shirobokov, Borislav Kozlovskii, Francisco JR Ruiz, Abbas Mehrabian, and 1 others. 2025. Alphaevolve: A coding agent for scientific and algorithmic discovery. *arXiv preprint arXiv:2506.13131*.

Long Qian, Xin Lu, Parvez Haris, Jianyong Zhu, Shuo Li, and Yingjie Yang. 2025. Enhancing clinical trial outcome prediction with artificial intelligence: a systematic review. *Drug Discovery Today*, 30(4):104332.

Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Matej Balog, M Pawan Kumar, Emilien Dupont, Francisco JR Ruiz, Jordan S Ellenberg, Pengming Wang, Omar Fawzi, and 1 others. 2024. Mathematical discoveries from program search with large language models. *Nature*, 625(7995):468–475.

Asankhaya Sharma. 2025. Openevolve: an open-source evolutionary coding agent.

Kien Wei Siah, Nicholas W Kelley, Steffen Ballerstedt, Björn Holzhauer, Tianmeng Lyu, David Mettler, Sophie Sun, Simon Wandel, Yang Zhong, Bin Zhou, and 1 others. 2021. Predicting drug approvals: The novartis data science and artificial intelligence challenge. *Patterns*, 2(8).

Aaditya Singh, Adam Fry, Adam Perelman, Adam Tart, Adi Ganesh, Ahmed El-Kishky, Aidan McLaughlin, Aiden Low, AJ Ostrow, Akhila Ananthram, and 1 others. 2025. Openai gpt-5 system card. *arXiv preprint arXiv:2601.03267*.

Rich Sutton. 2019. The bitter lesson. *Incomplete Ideas*.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, and 1 others. 2024. A survey on large language model based autonomous agents. *Frontiers of Computer Science*, 18(6):186345.

Zifeng Wang, Cao Xiao, and Jimeng Sun. 2023. Spot: Sequential predictive modeling of clinical trial outcome with meta-learning. In *Proceedings of the 14th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*, pages 1–11.

Shingo Yamaguchi, Masayuki Kaneko, and Mamoru Narukawa. 2021. Approval success rates of drug candidates based on target, action, modality, application, and their combinations. *Clinical and Translational Science*, 14(3):1113–1122.

Ling Yue, Sixue Xing, Jintai Chen, and Tianfan Fu. 2024. Clinicalagent: Clinical trial multi-agent system with large language model-based reasoning. In *Proceedings of the 15th ACM International Conference on Bioinformatics, Computational Biology and Health Informatics*, pages 1–10.

Wenhao Zheng, Liaoyaqi Wang, Dongshen Peng, Hongxia Xu, Yun Li, Hongtu Zhu, Tianfan Fu, and Huaxiu Yao. 2025. Lifted: Multimodal clinical trial outcome prediction via large language models and mixture-of-experts. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 7503–7517.

A Appendix

A.1 Results

Table 3: Results on four additional clinical trial tasks across phases. For trial failure reason identification (multi-class), macro-averaged metrics are reported.

		Phase 1			Phase 2			Phase 3			Phase 4		
Task	Method	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC
Patient dropout	LR	0.613	0.702	0.545	0.793	0.859	0.552	0.936	0.950	0.613	0.748	0.800	0.559
	XGBoost	0.632	0.705	0.562	0.789	0.864	0.545	0.944	0.951	0.644	0.755	0.811	0.568
	HINT	0.706	0.700	0.630	0.810	0.869	0.574	0.932	0.953	0.622	0.768	0.827	0.592
	TrialBench	0.712	0.714	0.724	0.773	0.868	0.718	0.919	0.946	0.761	0.709	0.800	0.652
	Ours	**0.857**	**0.784**	**0.786**	**0.939**	**0.808**	**0.809**	**0.981**	**0.886**	**0.860**	**0.897**	**0.794**	**0.810**
Adverse event	LR	0.606	0.450	0.671	0.817	0.844	0.636	0.904	0.919	0.672	0.517	0.372	0.638
	XGBoost	0.617	0.430	0.661	0.830	0.849	0.664	0.908	0.915	0.677	0.509	0.340	0.629
	HINT	0.763	0.652	0.767	0.849	0.845	0.687	0.906	0.915	0.669	0.492	0.411	0.619
	TrialBench	0.753	0.804	0.891	0.826	0.870	0.795	0.906	0.930	0.870	–	–	–
	Ours	**0.944**	**0.858**	**0.937**	**0.963**	**0.895**	**0.894**	**0.984**	**0.954**	**0.936**	**0.832**	**0.725**	**0.869**
Mortality event	LR	0.471	0.421	0.678	0.633	0.508	0.686	0.658	0.503	0.660	0.243	0.171	0.600
	XGBoost	0.509	0.457	0.728	0.648	0.512	0.696	0.678	0.514	0.681	0.265	0.155	0.603
	HINT	0.640	0.633	0.846	0.731	0.718	0.797	0.740	0.641	0.735	0.239	0.098	0.591
	TrialBench	0.653	0.770	0.909	0.663	0.692	0.827	0.669	0.683	0.814	–	–	–
	Ours	**0.881**	**0.873**	**0.957**	**0.938**	**0.838**	**0.925**	**0.928**	**0.845**	**0.916**	**0.698**	**0.547**	**0.888**
Failure reason	LR	0.271	0.282	0.523	0.286	0.263	0.555	0.292	0.282	0.562	0.255	0.244	0.511
	XGBoost	0.281	0.283	0.521	0.290	0.269	0.558	0.294	0.262	0.540	0.245	0.214	0.478
	HINT	0.281	0.205	0.563	0.304	0.267	0.573	0.308	0.258	0.562	0.275	**0.258**	**0.554**
	TrialBench	0.255	0.197	0.475	0.295	0.150	0.569	0.279	0.191	0.550	0.256	0.199	0.477
	Ours	**0.362**	**0.350**	**0.671**	**0.409**	**0.436**	**0.663**	**0.423**	**0.477**	**0.671**	**0.293**	0.243	0.541

Table 4: Searching frameworks results for four additional clinical trial tasks across phases. For trial failure reason identification (multi-class), macro-averaged metrics are reported.

		Phase 1			Phase 2			Phase 3			Phase 4		
Task	Method	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC	PR-AUC	F1	ROC-AUC
Patient dropout	ADAS	0.586	0.727	0.493	0.766	0.865	0.499	0.911	0.946	0.500	0.704	0.817	0.496
	Openevolve	0.836	0.751	0.765	0.918	0.861	0.769	0.982	0.954	0.860	0.893	0.825	0.771
	GEPA	0.821	0.757	0.765	0.924	0.870	0.780	**0.984**	0.953	**0.866**	0.892	**0.840**	0.775
	ShinkaEvolve	0.837	0.730	0.777	0.926	0.749	0.787	**0.984**	0.821	**0.866**	0.893	0.728	0.781
	AIDE	0.824	0.705	0.754	0.920	0.718	0.769	**0.984**	0.721	0.861	0.883	0.703	0.754
	Ours	**0.857**	**0.784**	**0.786**	**0.939**	**0.808**	**0.809**	0.981	**0.886**	0.860	**0.897**	0.794	**0.810**
Adverse event	ADAS	0.522	0.612	0.582	0.752	0.851	0.507	0.857	0.909	0.535	0.396	0.554	0.519
	Openevolve	0.918	0.824	0.916	0.955	0.883	0.880	0.983	0.936	0.915	0.807	0.675	0.844
	GEPA	0.895	0.800	0.895	0.947	0.865	0.862	0.983	0.924	0.913	0.773	0.688	0.827
	ShinkaEvolve	0.922	0.805	0.918	0.957	0.809	0.884	**0.984**	0.771	0.920	0.814	0.708	0.853
	AIDE	0.922	0.822	0.923	0.961	0.879	0.895	**0.984**	0.940	0.920	0.811	0.698	0.856
	Ours	**0.944**	**0.858**	**0.937**	**0.963**	**0.895**	**0.894**	**0.984**	**0.954**	**0.936**	**0.832**	**0.725**	**0.869**
Mortality event	ADAS	0.294	0.439	0.557	0.466	0.630	0.515	0.499	0.649	0.525	0.184	0.305	0.532
	Openevolve	0.852	0.813	0.944	0.881	0.800	0.896	0.884	0.767	0.869	0.595	0.426	0.854
	GEPA	0.869	0.836	0.953	0.893	0.817	0.904	0.902	0.798	0.890	0.603	0.514	0.854
	ShinkaEvolve	0.868	0.820	0.949	0.896	0.819	0.911	0.903	0.800	0.892	0.653	**0.558**	0.867
	AIDE	0.870	0.673	0.950	0.891	0.814	0.907	0.900	0.798	0.887	0.651	0.469	0.862
	Ours	**0.881**	**0.873**	**0.957**	**0.938**	**0.838**	**0.925**	**0.928**	**0.845**	**0.916**	**0.698**	0.547	**0.888**
Failure reason	ADAS	0.251	0.232	0.502	0.252	0.182	0.503	0.250	0.196	0.499	0.251	0.206	0.499
	Openevolve	0.287	0.346	0.560	0.309	0.384	0.589	0.325	0.408	0.607	0.269	0.274	0.533
	GEPA	0.286	0.337	0.582	0.301	0.380	0.598	0.309	0.399	0.606	0.268	**0.293**	**0.543**
	ShinkaEvolve	0.289	**0.353**	0.565	0.308	0.383	0.585	0.332	0.432	0.613	0.264	0.278	0.525
	AIDE	0.265	0.247	0.519	0.298	0.337	0.562	0.331	0.406	0.594	0.266	0.269	0.528
	Ours	**0.362**	0.350	**0.671**	**0.409**	**0.436**	**0.663**	**0.423**	**0.477**	**0.671**	**0.293**	0.243	0.541

A.2 Prompts

A.2.1

Goal Prompt

```
Task: Binary classification — predict whether a clinical trial is approved.

Data: `data_path` contains four phase folders: Phase1/, Phase2/, Phase3/,
Phase4/. Each has train_x.csv, train_y.csv. The first column
of every file is the NCT trial ID — the join key. X and Y rows are NOT
pre-aligned; join by NCT ID. The label column in *_y.csv is `outcome`
(binary 0/1). Phase schemas may differ — explore the data first.

Approach: You can freely decide the approach you want to use for this
task. You can write and run code, create agents —
reasoning-only or tool-using via MCP tools (PubMed, ClinicalTrials.gov,
OpenFDA, ChEMBL, web search) — or hybrid. Pick what fits the
problem and let the evidence reshape it across rounds; don't settle
into one mode out of habit.

Metric: `pr_auc_macro` — the mean of per-phase PR-AUC.

Generalization: ONE approach that works across all 4 phases and would
transfer to other datasets with similar structure. Do not concatenate phases.
Consistency across phases matters more than spiking in one.

Clinical soundness: the solution must be defensible at the clinical /
regulatory level. Solution and decision logic should reflect real-world
drivers of trial outcome.
```

Figure 3: Goal prompt for the trial approval prediction task.

A.2.2

Meta-Agent Prompt

```
You are the Meta-Agent for a self-evolving research system.

Your job each round: decide the approach, delegate the work.

=== ROUND WORKFLOW ===
1. Read the GOAL, ROUND HISTORY, TRAJECTORY (long-view metric arc), any REFLECTION INSIGHTS,
and PROGRESS. Per-round metric breakdowns appear under `### Evaluator
metrics`; the previous round's plan and explicit handoff appear under `### Plan & rationale`
and `### Handoff` — those are your source of truth for prior performance and prior intent.
2. Decide this round's approach (see GOAL INTERPRETATION + CHOOSING THE APPROACH below).
3. Write the pipeline: ask the coding agent for a complete `pipeline.py`,
 save it to `<ROUND_DIR>/pipeline.py`.
4. The framework runs a user-supplied evaluator after your round ends
and reports the score back into ROUND HISTORY.
5. End your final response with two sections, in this order:

   ### Plan & rationale
   A short narrative (a few short paragraphs): what you did, why it fits the task

   ### Handoff
   - problem: <the concrete problem this round addressed — what was hard or unresolved>
   - solution: <what you actually did this round>
   - prevention: <what to avoid or set up before the next round>

   The framework parses this block at round end and writes a single round entry
   to PROGRESS.md containing the metric, the per-metric evaluator breakdown,
   your plan/rationale, and your handoff.
```

```
=== CHOOSING THE APPROACH ===
Available modes (combine as useful):
- Pure code: (e.g. ML, optimization)
- Agent: reasoning-only or tool-using (e.g. agents wired to  pubmed_search/search_trials)
- Hybrid approach: Code + Agents, sequential/parallel/hierarchical

Pick based on problem fit + what the history shows. Treat every round as potentially the last.

=== PIPELINE CONTRACT ===
Every round MUST produce `<ROUND_DIR>/pipeline.py` that:
- Accepts `--data DIR` (input) and `--out DIR` (output) via argparse.
- Is runnable as `python pipeline.py --data <X> --out <Y>` from outside the framework.

=== REFLECTION HANDLING ===
When the context contains any of these blocks you must engage with them explicitly in this round's plan:

- `[REFLECTION INSIGHTS]` — read it; it summarizes solution analysis.
- `[SOLUTION DIRECTIVE]` — contains `solution_critique`, `ablation_proposals`, `diversity_warnings`.
You MUST either (a) run at least one high/medium-priority ablation this round,
(b) pivot per a diversity warning if the metric has plateaued, or
(c) state briefly in your final answer why the proposal doesn't apply to this round.
Silent ignoring is not allowed.

=== STYLE RULES ===
- Be specific in `request_code` tasks: state the input/output,
the data schema, the metric, and any constraints.
- Treat the `### Handoff` section in step 5 as a real handoff to the next round, not a checkbox —
it's the durable record the next round reads.
```

Figure 4: Meta-Agent instruction prompt.

A.2.3

Reflection Agent Prompt

```
Analyze this self-evolving research system and provide strategic insights.

REFLECTION TRIGGER: {trigger_reason}

The trigger dictates BOTH what to focus on AND how much to emit.
Leave non-listed JSON fields as [] or "".


- "failure:..." → a round produced no real metric. Focus
  ENTIRELY on identifying the failure mode.
    * recommendations: ONE concrete debug step.
    * solution_critique: 1 sentence naming the failure mode.
    * ablation_proposals / diversity_warnings / redundancies / gaps: leave empty.

- "regression:..." → last round was meaningfully worse than running best.
  Focus on what changed between the best round and the last round.
    * solution_critique: 2-3 sentences.
    * ablation_proposals: at most 2.
    * diversity_warnings: 0-1, only if the regression looks family-related.
    * redundancies / gaps: leave empty unless directly relevant to the regression.

- "plateau:..." → metric stuck across recent rounds. Push for a structural
  pivot, not a parameter tweak.
    * diversity_warnings: 1-2 entries — name the stuck family and a
      concrete pivot direction.
    * ablation_proposals: 2-3 entries.
    * solution_critique: comment on whether complexity is growing without
      gain (bloat) or actually warranted.

- "stale:..." → no signal-based trigger fired recently, but enough rounds
  have passed that a periodic health check is worthwhile. The metric is
```

```
  not necessarily stuck and there is no specific failure to diagnose; this
  is the maintenance pass. Stay conservative — only flag things you can
  actually back up from the data.
    * solution_critique: 2-3 sentences on the current trajectory and
      whether the active approach still looks healthy.
    * recommendations: up to 3 concrete, actionable items.
    * ablation_proposals / diversity_warnings: include only if the data
      genuinely supports them; empty is fine here.
    * redundancies / gaps: include only if obvious from the registry.


Round History (metric trajectory across rounds):
{from memory}

Recent Solutions (last 3 rounds):
{codes}

Recent Round Narratives:
{from memory}


=== SOLUTION ANALYSIS ===

Examine the recent code artifacts and metric trajectory. Produce:

1. SOLUTION_CRITIQUE: A concise read of the current approach's health.
   - Is the metric improving, flat, or degrading across rounds?
   - Is complexity growing without matching metric gains? (code size up,
     metric flat/down → bloat)
   - Signs of overfitting or metric hacking? (post-hoc rules, suspicious
    jumps without methodological change)
   - Are recent rounds churning on the same idea without real variation?

2. ABLATION_PROPOSALS: Concrete, testable simplifications to probe
   generalizability. Each should name a specific thing to remove or swap and
   predict what its removal would reveal.
   Bad ablations to avoid: vague ("simplify the pipeline"), untestable, or
   tautological.

3. DIVERSITY_WARNINGS: If the last N rounds are variations on the same theme
   and the metric has plateaued, flag it and suggest a fundamentally different
   direction that still respects any GOAL constraint.

4. RECOMMENDATIONS: A short list of the most important, actionable next steps
   — drawn from both layers.

Return ONLY valid JSON in this exact format:
{{
  "solution_critique": {{
    "trajectory": "improving/flat/degrading",
    "complexity_vs_gain": "healthy/bloating/unclear",
    "overfitting_risk": "low/medium/high",
    "notes": "1-3 sentence narrative explaining the above"
  }},
  "ablation_proposals": [
    {{"change": "what to remove or swap", "rationale": "what this would reveal",
     "priority": "high/medium/low"}}
  ],
  "diversity_warnings": [
    {{"observation": "what's repetitive", "suggested_pivot": "a different direction to try"}}
  ],
  "recommendations": [
    "actionable recommendation 1",
    "actionable recommendation 2"
  ]
}}
```

Figure 5: Reflection Agent instruction prompt.

A.3 Searching Baselines

We compare our framework against five search frameworks. ShinkaEvolve, OpenEvolve, AIDE, and GEPA evolve code solutions for a given task, while ADAS searches over agentic system architectures. As with our framework, each baseline runs a unified search for a given task across all its phases. Below we provide a short description of each baseline:

AIDE (Jiang et al., 2025): frames machine learning engineering as a code optimization problem, casting experimentation as tree search over candidate solutions. Each node in the search tree is a complete, runnable Python program. At every step, the agent uses an LLM to either draft a new solution, debug a failing one, or improve a working one; the candidate is then executed and scored by a validation metric, and the result conditions the next step.

ShinkaEvolve (Lange et al., 2025) is an evolutionary program optimization framework that maintains a population of programs and repeatedly uses an LLM as a mutation operator to rewrite marked code regions, with an external evaluator providing fitness scores to drive selection. It is primarily designed for sample efficiency, achieved through adaptive parent sampling that balances exploration and exploitation, novelty-based rejection filtering that discards redundant candidates before evaluation, and a bandit-based LLM ensemble that dynamically selects the best model for each mutation.

OpenEvolve (Sharma, 2025) is an open-source reimplementation of AlphaEvolve (Novikov et al., 2025). The user marks the mutable region of a seed program; an LLM proposes edits to that region, each offspring is executed and scored, and survivors are kept in a population archive that rewards both quality and diversity. Successive generations are conditioned on previously discovered high-performing programs and evaluator feedback, allowing the system to iteratively refine solutions.

GEPA (Agrawal et al., 2025) is a reflective optimizer whose core mechanism is reflective mutation: an LLM reads its own execution traces, diagnoses why a candidate failed, and proposes a targeted edit. Candidates are evaluated per instance and retained on a Pareto frontier so that solutions specializing on different parts of the task are not prematurely discarded. GEPA was originally introduced to optimize instruction prompts of compound LLM systems; we adapt it for program evolution by setting the seed candidate to a complete Python module rather than a prompt string, so the text being mutated is the whole program. In short, GEPA's reflective mutation and Pareto selection machinery are preserved; only the artifact type is changed from prompt text to program code.

ADAS (Hu et al., 2025) targets the automatic design of agentic systems. Its core algorithm, Meta Agent search, instructs a meta agent to iteratively propose new agent architectures, including control flow and prompting strategy. Each new agent is evaluated on the task and added to an archive of discovered agents that conditions subsequent proposals, producing progressively better agent designs over many iterations.

A.4 Discovered Solution

Figure 6 shows the solution discovered by Meta-Trial Agent. Input trial data first passes through an Intervention Scanner, a preprocessing step that extracts a ranked list of the most frequent intervention names across the training set. This list is handed to the Drug Linker agent, which queries ChEMBL to attach molecule-level regulatory priors such as ATC classification and biologic status, returned as a structured artifact. In parallel, three additional agents are invoked. The Clinical Priors agent produces a curated dictionary mapping clinically grounded design and endpoint signals (randomization, blinding, control type, among others) to regex patterns, with terminology validated against PubMed and ClinicalTrials.gov. The Sponsor Tierer agent queries OpenFDA to produce a rule set mapping sponsor names to capacity tiers with associated weights. The Data Architect, a reasoning-only agent, provides regex-based patterns that map column names to semantic groups for feature extraction. All four agents operate under fixed system prompts defining their roles, available tools, and required JSON output schemas, producing deterministic structured artifacts rather than free-form text. In the subsequent routing and feature engineering step, the

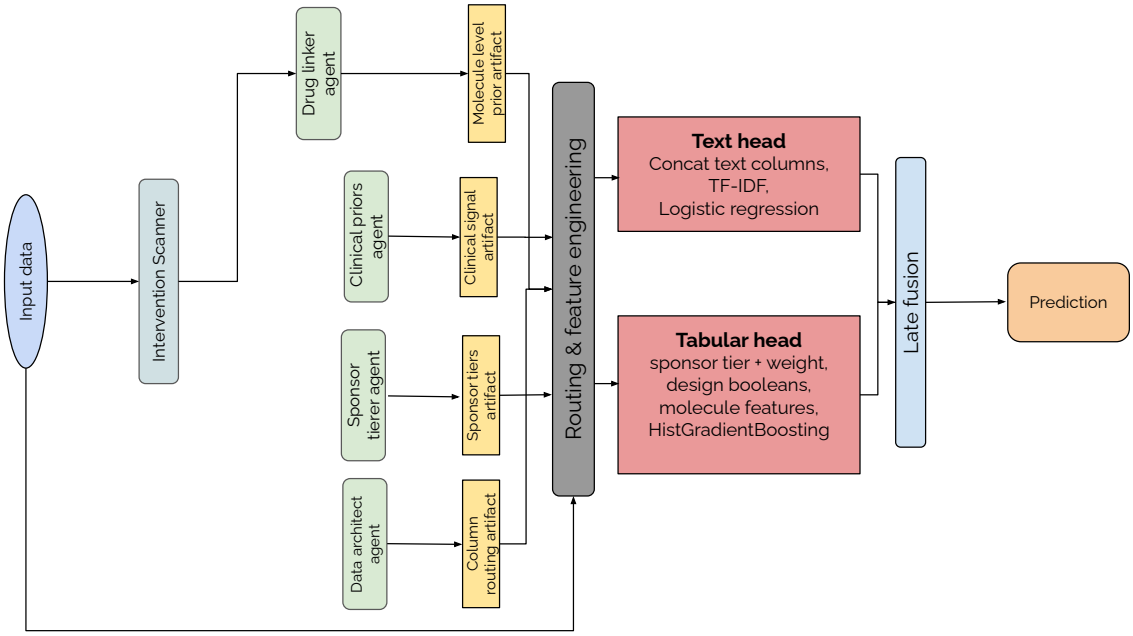
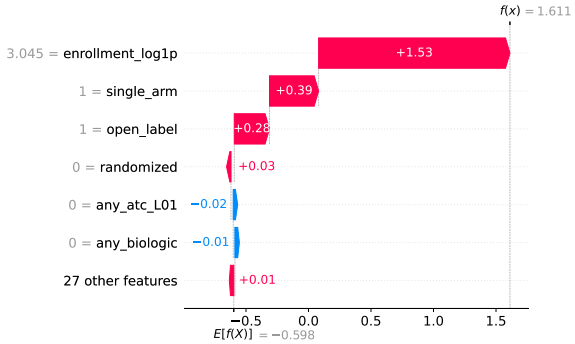


Figure 6: The architecture of the solution discovered by Meta-Trial Agent for the trial approval prediction task. Four specialized agents produce structured artifacts grounded in external biomedical knowledge, which are combined with the trial data in a routing and feature engineering step. The resulting features feed a two-head late-fusion pipeline that produces the final approval prediction.

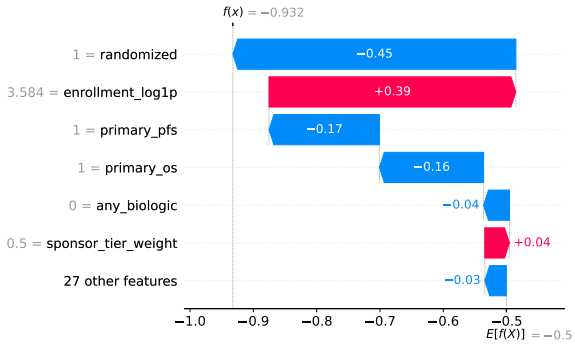
trial data is combined with the four agent-produced artifacts to derive model-ready features. The column mapping identifies which input columns play which role for the current phase; the clinical signal patterns are applied to the routed text to produce binary design and endpoint features; intervention names are looked up in the molecule-level prior artifact to derive molecule-aware features; and sponsor names are mapped to tiers and weights. The Text Head then concatenates the routed text columns, vectorizes them with TF-IDF, and trains a logistic regression classifier to produce a text-based approval probability. The Tabular Head consumes the engineered tabular features and trains a histogram gradient-boosted classifier to produce a complementary probability. The two probabilities are combined in a Late Fusion step, which weights each head according to its measured ranking quality on a held-out portion of the training data, producing the final approval prediction for each trial.

A.5 SHAP Plots

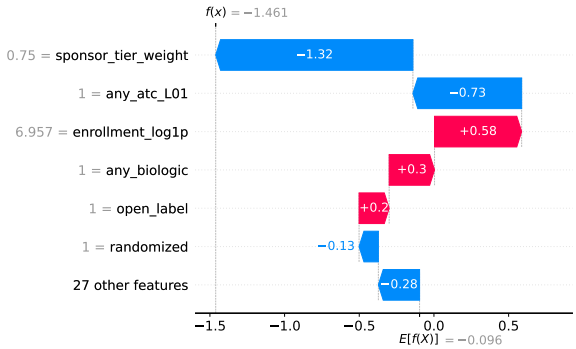
Figures 7a, 7b, 7c, and 7d show SHAP (Lundberg and Lee, 2017) waterfall plots for four correctly classified samples from the test set across all four phases. Enrollment size and randomization status consistently rank among the most influential features, reflecting the well-documented importance of adequate statistical power and bias minimization in determining trial outcomes. Sponsor capacity tiers emerge as strongly predictive in Phase 3, consistent with evidence that sponsor experience is associated with higher approval rates. In Phase 2, primary endpoint indicators such as progression-free survival and overall survival rank highly, aligning with regulatory emphasis on clinically meaningful endpoints. Trial design indicators such as single-arm and open-label status also surface as influential, reflecting how design choices shape the interpretability of trial evidence. Molecule-level features including ATC classification and biologic status further contribute across phases, reflecting the distinct regulatory trajectories of different therapeutic classes.



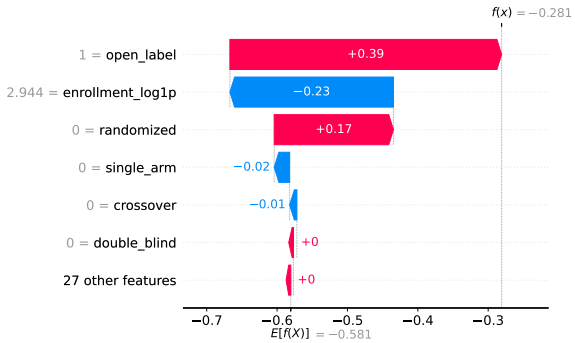
(a) Phase 1 sample



(b) Phase 2 sample



(c) Phase 3 sample



(d) Phase 4 sample

Figure 7: SHAP waterfall plots for correctly classified samples from the test set across all four phases of the trial approval prediction task.