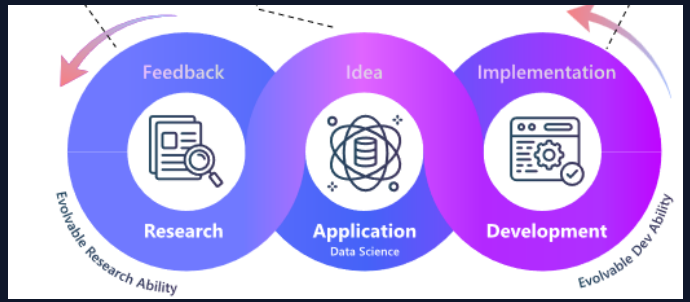
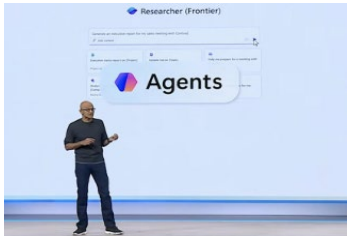


LLM-based Auto-evolving Agents for Data-driven R&D

Shikai Fang
Assistant Professor@Zhejiang University
Talk for Yale · 2026.04

fsk@zju.edu.cn

Everyone is talking about agent!



What Is Agent?

Agents are systems that **independently** accomplish tasks on your behalf.

Python

```python
weather_agent = Agent(
    name="Weather agent",
    instructions="You are a helpful agent who can talk to users about the weather.",
    tools=[get_weather],
)
```

What Is Agent in Real World..

- “Fancy words are invented everyday… and combined randomly”

Perception

Structured feedback (logs, metrics, curves)

Reasoning

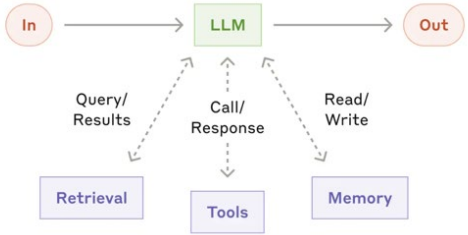
LLM-driven hypothesis formation

Action

Code execution, API calls, simulator runs

Memory

Accumulates outcomes · makes it evolving



“Augmented LLM”

“evaluator-optimizer workflow”

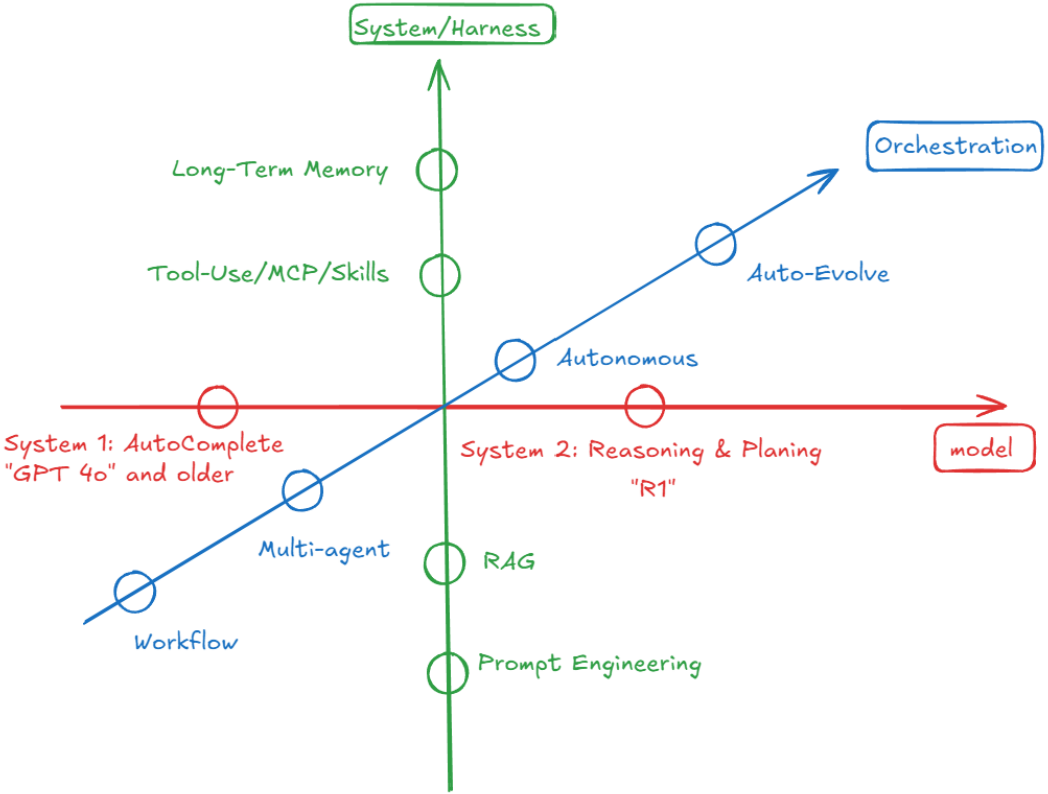
“Orchestrator-workers”

“Fully Autonomous”

Figs: “Building effective agents”, blog of anthropic

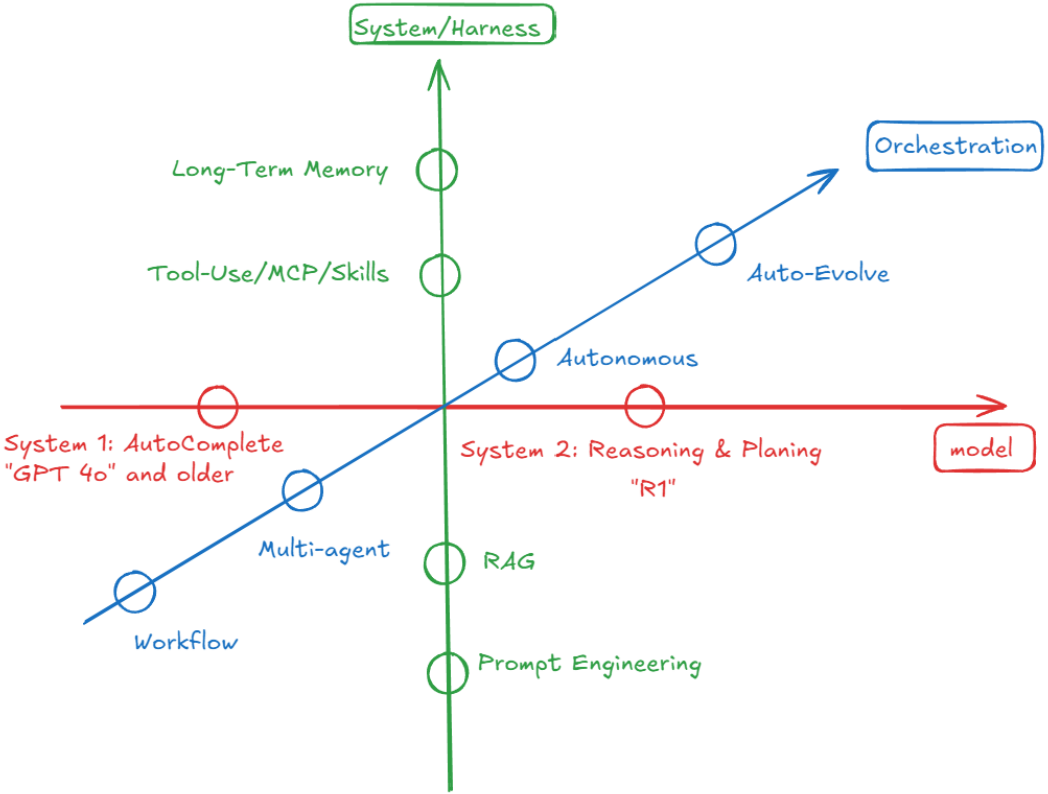
What Is Agent in Real World..

- “Trying to sort out such fancy words..”



What Is Agent in Real World..

- **“Trying to sort out such fancy words..”**



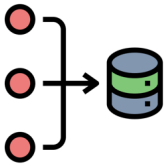
Fancy Enough!...?

for tasks like
AI4Coding/GUI/Web:

- Clear Feedback (Yes/No)
- Short-Term Evaluation

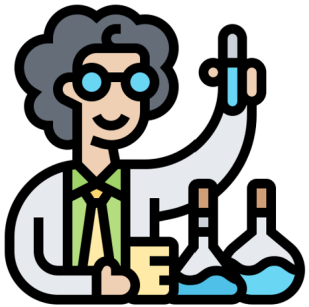
High-valued R&D tasks in industry

- Deliver solutions for **objectively measurable**, **data-driven** industrial applications.

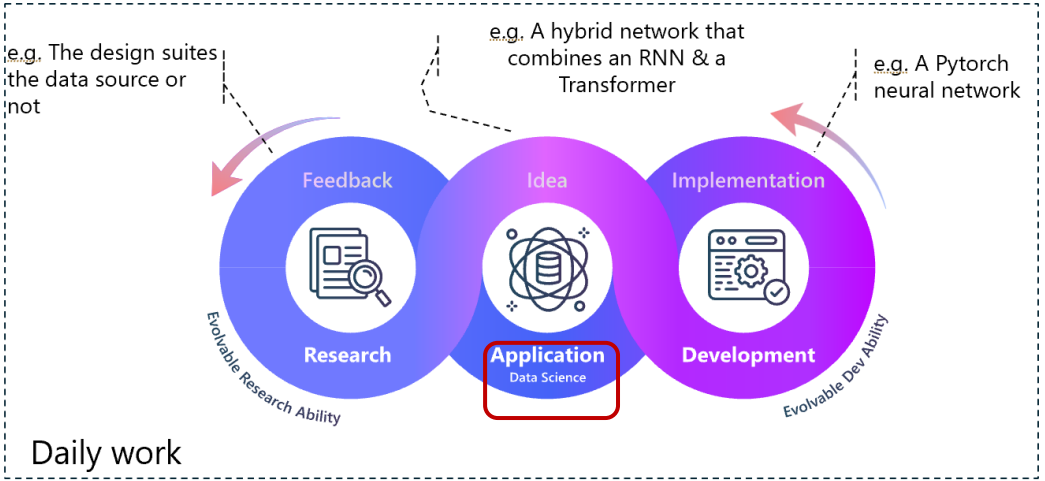


Unique data source

Objectively measurable task

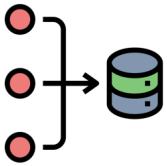


Applied data scientist



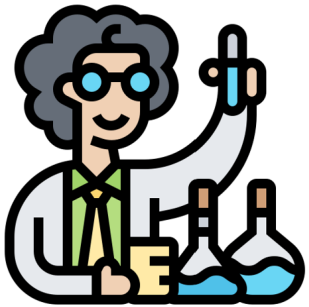
High-valued R&D tasks in industry

- Long-term and long-chain evaluations
- Mul-Obj., Mul-Solutions, Rich feedback



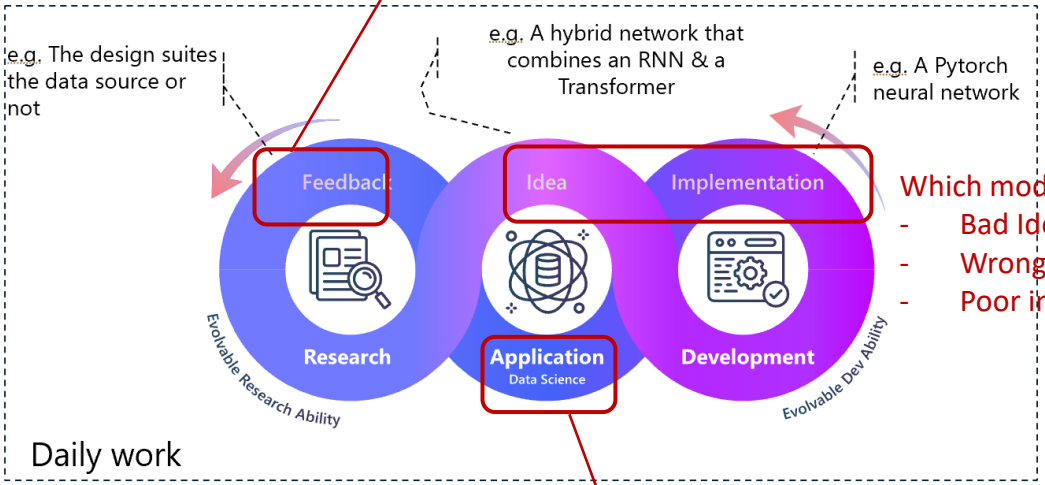
Unique data source

Objectively measurable task



Applied data scientist

Rich feedback, beyond Yes/No



Which module to polish?:

- Bad Idea?
- Wrong feature?
- Poor implementation?

Long-Term, High-Cost, Black-Box

How to design agent for R&D tasks?

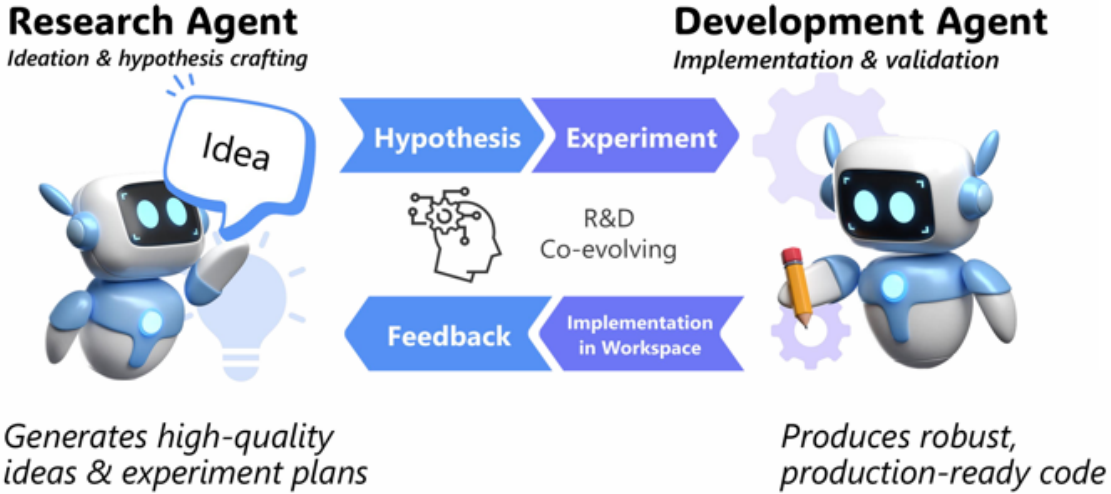
High-level principals:

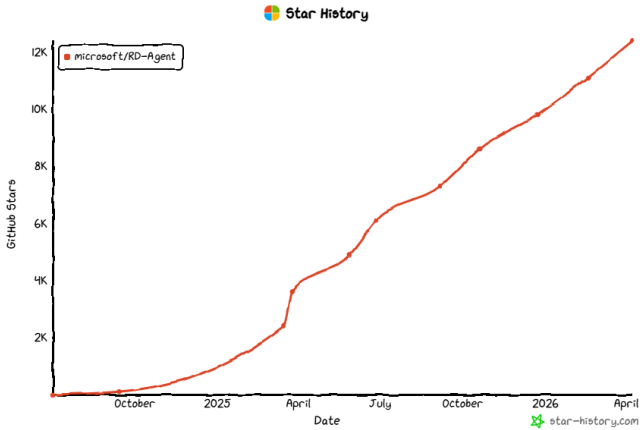
- Trial-and-error loops

- Reasoning as Optimization

A simple two-agent framework: R&D-Agent

Traditional Applied-Data-Science R&D: A Trial-and-Error Evolving
***R&D-Agent* codifies human experts' R-D Co-Evolution Loops**

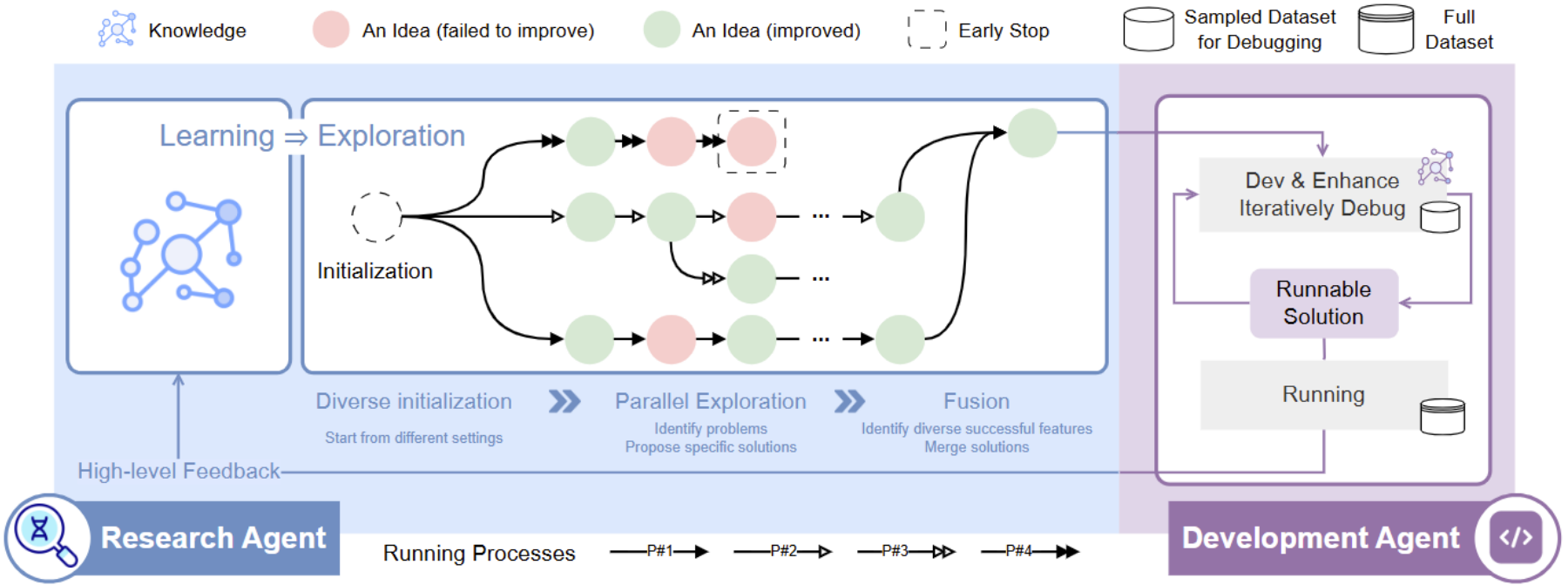




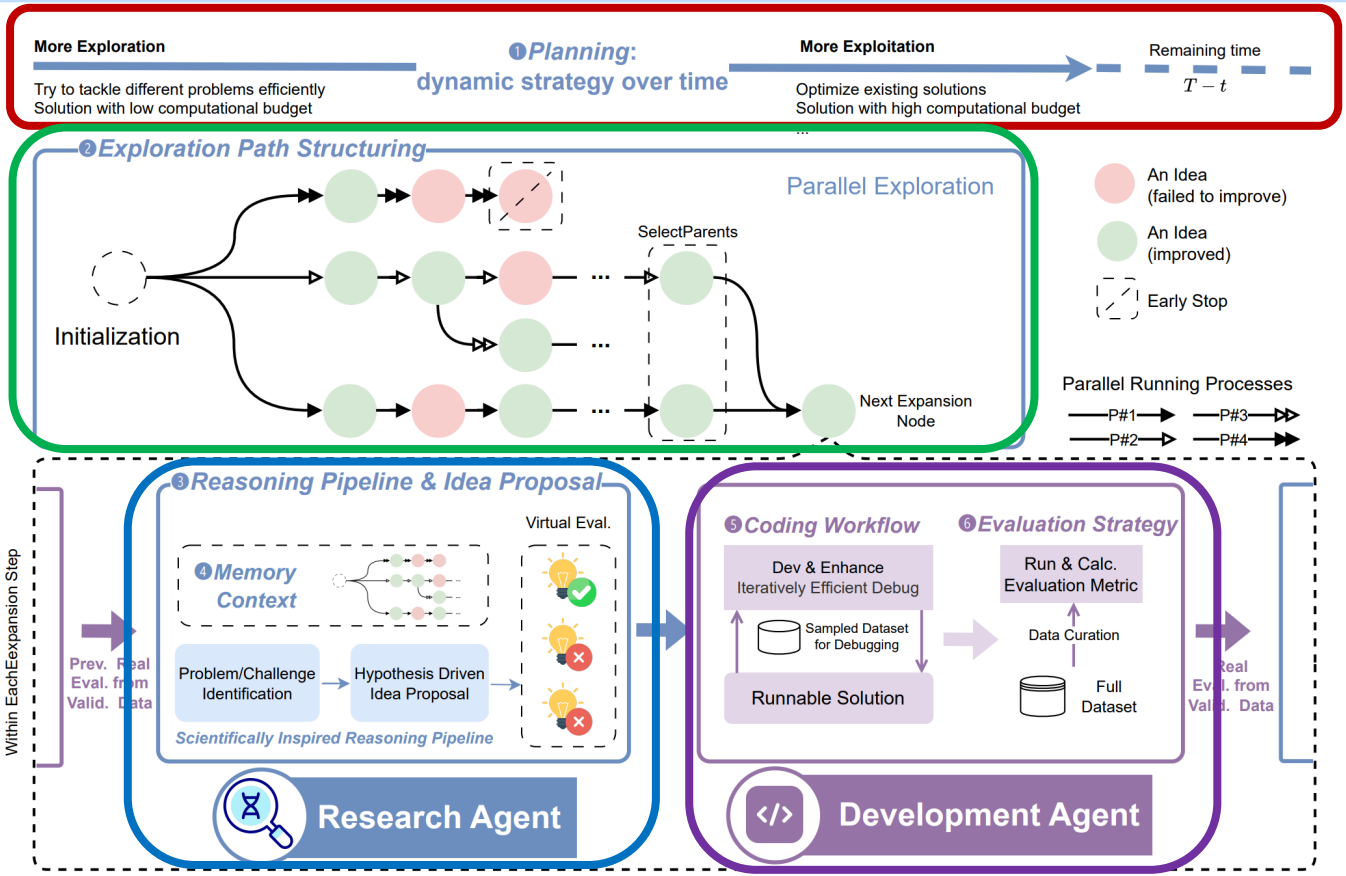
- https://github.com/microsoft/RD-Agent
- Arxiv: 2505.14738v2, 2603.01692v2

R&D-Agent: deep dive

- Research agent: more **efficient** exploration
- Dev. agent: more **robust** implementation



R&D-Agent: deep dive



R&D-Agent: Prompt Example

Scientific Hypothesis Generation —— Reasoning Pipeline Component

You are a research scientist formulating testable hypotheses. For each hypothesis, perform two main tasks: hypothesis proposal and rigorous five-dimensional evaluation.

Hypothesis Development Guidelines:

1. **Specificity & Decisiveness:** State exact, unambiguous changes. Avoid vague goals or alternatives.
2. **Testability & Actionability:** Describe implementable and measurable changes. Focus on single, unified conceptual improvements.
3. **Evidence-Based Reasoning:** Ground hypotheses in experimental history or domain knowledge.
4. **Implementation Feasibility:** Consider resource constraints and technical complexity.

Five-Dimensional Evaluation Protocol: Score each hypothesis (1-10) across:

- **Problem-Hypothesis Alignment:** How well the hypothesis addresses the identified problem
- **Expected Impact:** The estimated improvement after applying the hypothesis
- **Novelty:** Degree of innovation compared to previous attempts
- **Feasibility:** The ease of implementing the proposed hypothesis
- **Risk-Reward Balance:** The exploration-exploitation balance of the proposed hypothesis

Component Classification: Assign each hypothesis to: `DataLoadSpec`, `FeatureEng`, `Model`, `Ensemble`, or `Workflow`.

Memory-Enhanced Code Generation —— Memory Context Component

You are a grandmaster-level data scientist generating robust, debuggable code following systematic development process.

Important Context: You are working on sample datasets and your code will go through automated iterations. Design your code to be iteration-friendly with comprehensive print statements and clear debugging information to facilitate the automatic improvement process.

Memory-Enhanced Guidelines:

1. **Historical Learning:** Reference previous failed attempts and their feedback
2. **Pattern Reuse:** Apply successful patterns from similar tasks
3. **Error Prevention:** Avoid mistakes that occurred in previous experiments
4. **Performance Optimization:** Implement improvements suggested in historical feedback

Quality Assurance Requirements:

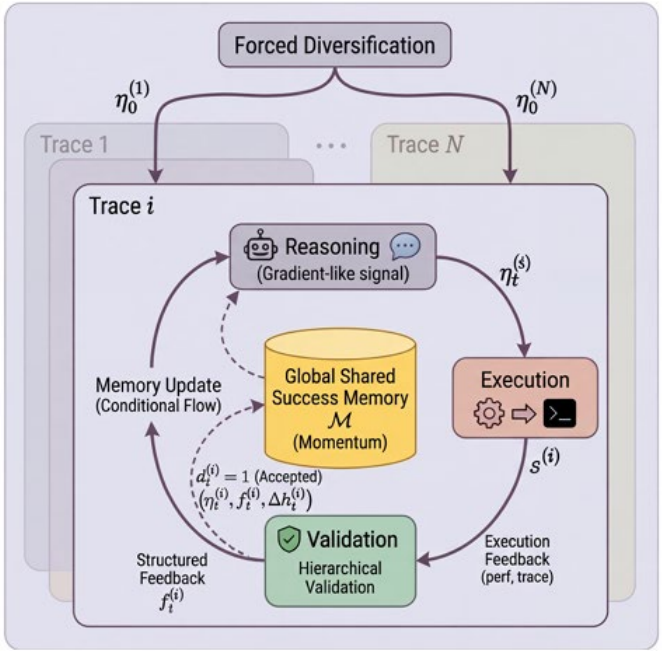
- **Debug Mode Integration:** Support `-debug` flag with data sampling and timing estimation
- **Structured Output:** Use print statements for progress tracking, avoid external logging dependencies
- **Reproducibility:** Implement proper random seed management and deterministic behavior
- **Resource Management:** Dynamic resource allocation and proportional data splitting

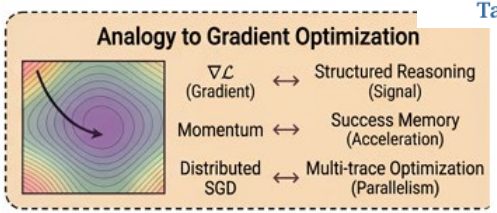
R&D-Agent : deep dive

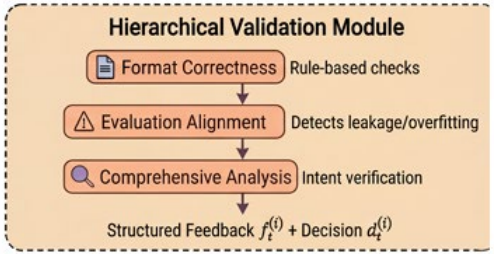
- **Reasoning as Optimization**

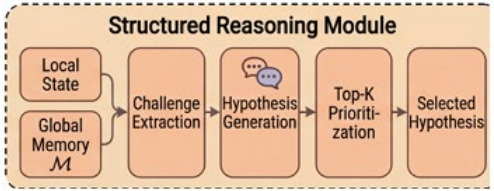
Concept	GOME Component	Functional Role
Gradient ∇L	Structured Reasoning	Signal: decides how to update
Momentum	Success Memory	Acceleration via proven patterns
Distributed SGD	Multi-trace Optimization	Parallelism with knowledge sharing

Table 2: Analogy between GOME and gradient-based optimization.



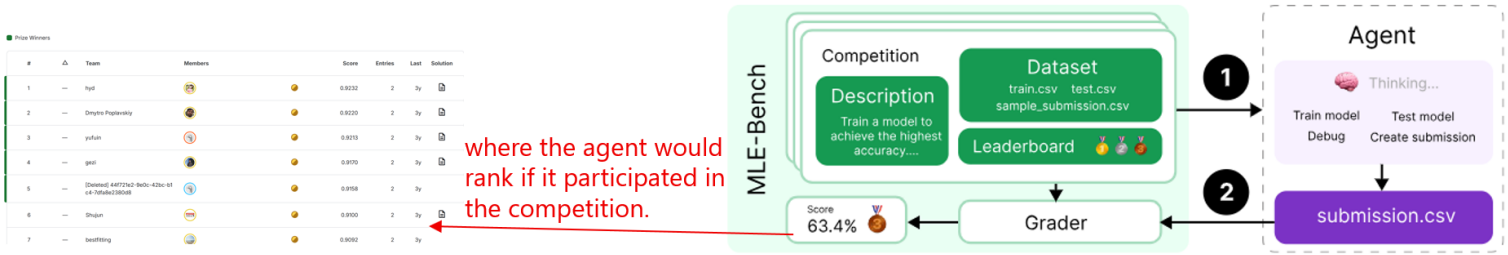






➥ MLE-Bench: the largest public ML-engineering benchmark

- 75 kaggle competitions crossing several domains published and open-sourced by OpenAI



- Industry span

- Task span
  - classification, regression, object detection, semantic segmentation, sequence prediction, QA…

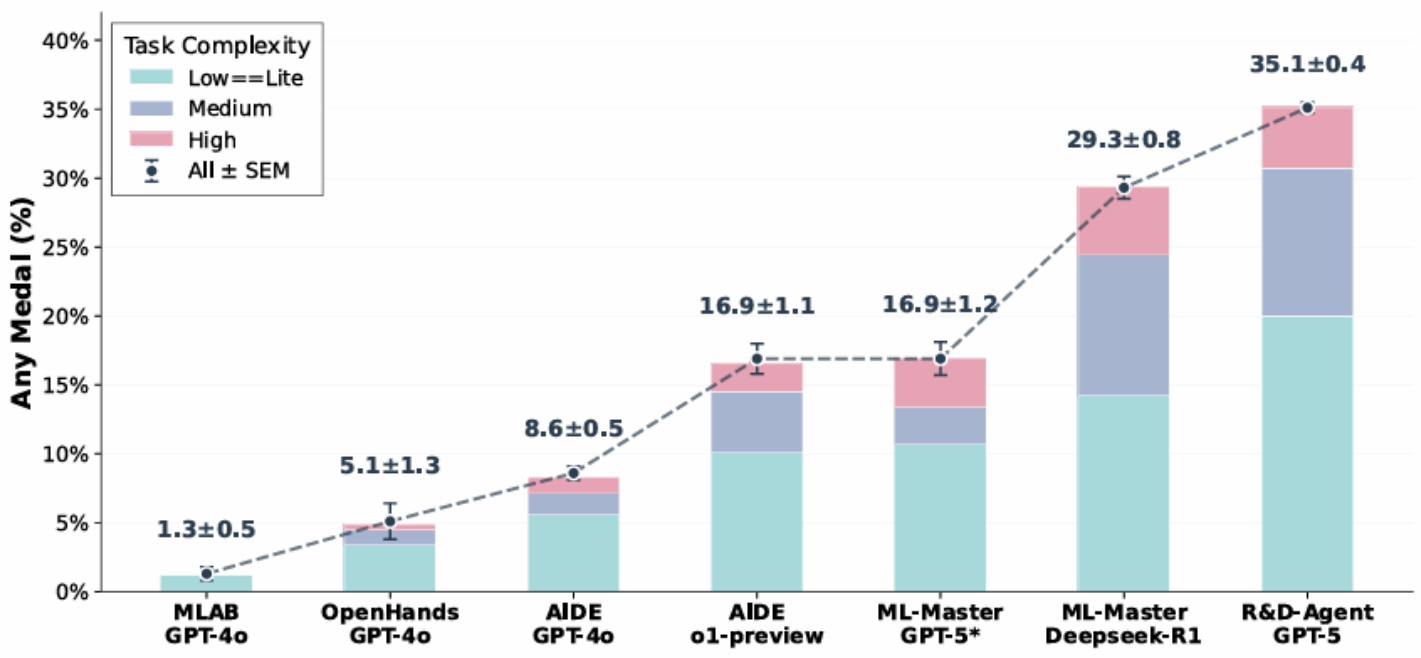


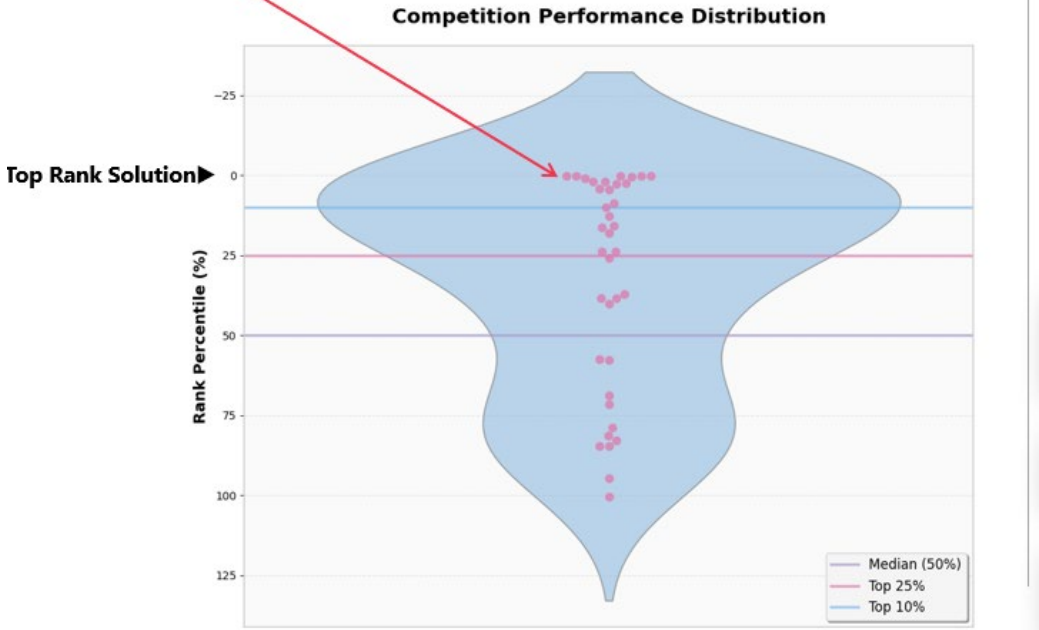
Figure 1: Agent performance on MLE-Bench. Stacked bars show any medal rates for Low==Lite (22 tasks), Medium (38 tasks), and High (15 tasks) complexity levels. The dashed line indicates overall performance (mean ± SEM). R&D-Agent achieves SOTA performance at 35.1 ± 0.4%. * indicates our re-evaluation of ML-Master within our environment.

Timestamp： 2025 May

- Is it possible to surpass the best solutions currently achieved by human experts ?

We have solutions that goes beyond the best ever human experts' solutions!

9.3% chance of exceeding the 🥇 **top 1 solution** achieved by human experts(evaluated on MLE-Bench).



Case study

The best ever human experts' solution

	Info
Competition	stanford-covid-vaccine
Target	nucleotide-wise degradation of vaccine

JIAYANG GAO · 1ST IN THIS COMPETITION · POSTED 5 YEARS AGO

1st place solution

First, many thanks to the Kaggle team and Stanford team for hosting this competition, and their kernels and ideas - I learnt a lot from you and was inspired by you. And congrats to al

My ideas are mainly on the data side (especially the way to use pseudo labeling), as architecture - my initial versions were based on @xhlulu's solid gru baseline. I added

based on @xhlulu's solid gru baseline. I added distance eml

xhlulu's solid gru baseline. I added distance embedding (to be ublished the excellent ae+gnn kernel. I merged my frame work

What is R&D-Agent's solution that surpasses the best human solutions?

Key Idea:

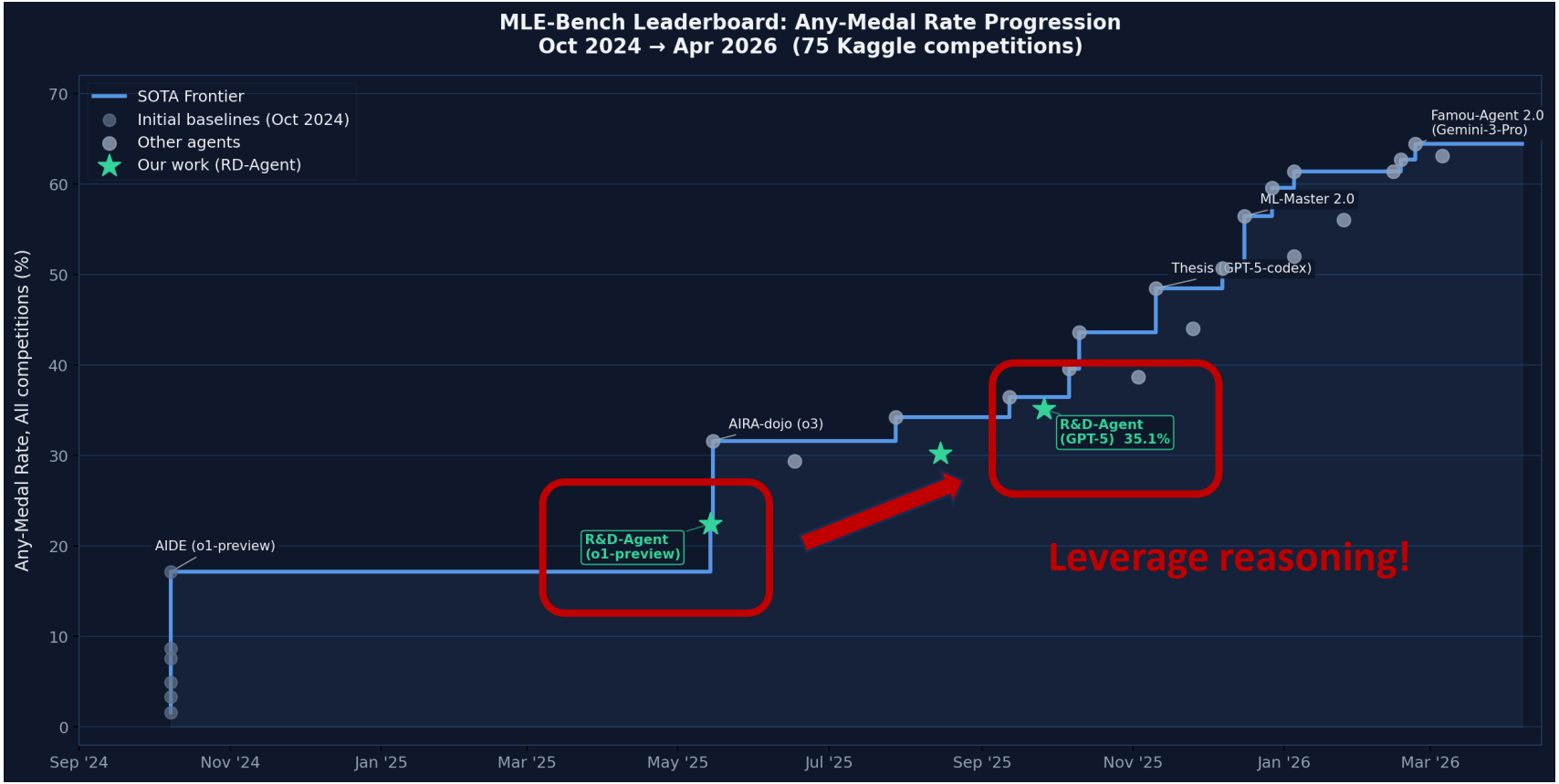
Crafting features in traditional models

Employing advanced models that learn patterns automatically

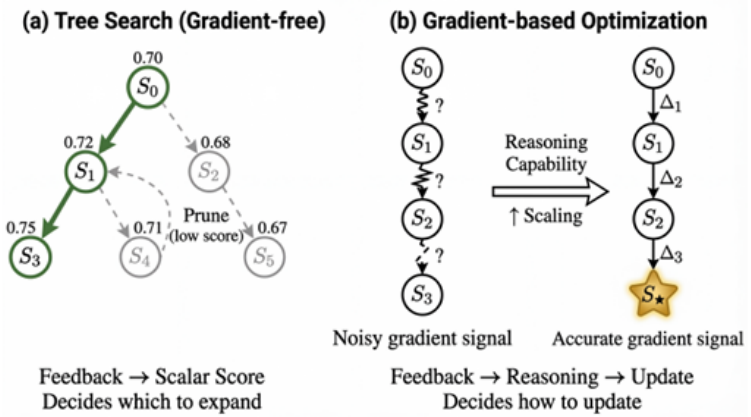
- A transformer with a customized attention mechanism
- Learning-driven attention instead of complex feature engineering.
  - Static graph relations are learned by a shared bias weight.
  - Dynamic graph relations are learned by diverse embeddings.

```python
class RelationalMultiHeadSelfAttention(nn.Module):
    def __init__(self, d_model, n_heads, dropout=0.1, n_relations=3):
        super().__init__()
        self.d_model = d_model
        self.n_heads = n_heads
        self.n_relations = n_relations
        assert d_model % n_heads == 0, "d_model must be divisible by n_heads"
        self.d_head = d_model // n_heads

        self.W_q = nn.Linear(d_model, d_model)
        self.W_k = nn.Linear(d_model, d_model)
        self.W_v = nn.Linear(d_model, d_model)
        self.W_o = nn.Linear(d_model, d_model)
        self.dropout = nn.Dropout(p=dropout)

        # Learnable bias per relation-type and head, initialized to 0.1
        # Shape: (n_relations, n_heads)
        self.bias_table = nn.Parameter(torch.full((n_relations, n_heads), 0.1))
```

```python
class TransformerSeqRegressor(nn.Module):
    def __init__(self,
                 vocab_nuc_size,
                 vocab_struct_size,
                 vocab_loop_size,
                 pos_emb_size,
                 d_model=128,
                 n_layers=4,
                 n_heads=8,
                 d_ff=256,
                 dropout=0.1,
                 num_targets=5,
                 use_rel_bias=True):
        super().__init__()
        self.nuc_emb = nn.Embedding(vocab_nuc_size, 16)
        self.struct_emb = nn.Embedding(vocab_struct_size, 8)
        self.loop_emb = nn.Embedding(vocab_loop_size, 8)
        self.pos_emb = nn.Embedding(SEQ_LEN, 32)
        self.pair_emb = nn.Embedding(108, 16)
        self.dist_emb = nn.Embedding(8, 16)
        self.layernorm = nn.LayerNorm(96)
        self.linear_proj = nn.Linear(96, d_model)
```

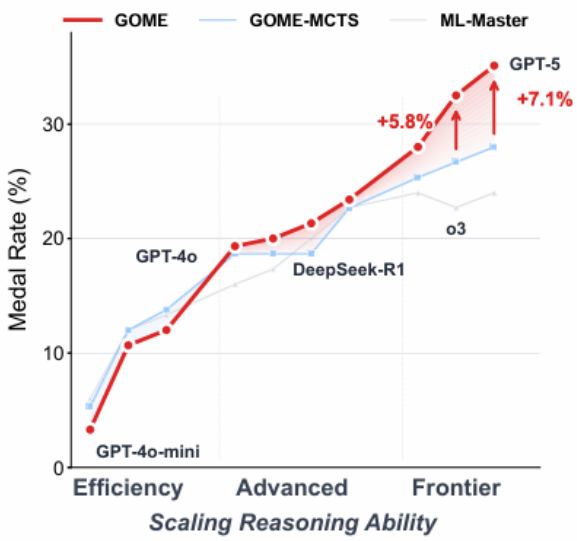

MLE-Bench Leaderboard: Any-Medal Rate Progression
Oct 2024 → Apr 2026 (75 Kaggle competitions)
SOTA Frontier
Initial baselines (Oct 2024)
Other agents
Our work (RD-Agent)
Any-Medal Rate, All competitions (%)
0
10
20
30
40
50
60
70
Sep '24
Nov '24
Jan '25
Mar '25
May '25
Jul '25
Sep '25
Nov '25
Jan '26
Mar '26
AIDE (o1-preview)
AIRA-dojo (o3)
Thesis (GPT-5-codex)
ML-Master 2.0
Famou-Agent 2.0
(Gemini-3-Pro)
R&D-Agent
(o1-preview)
R&D-Agent
(GPT-5) 35.1%
Leverage reasoning!

Scaling effect of backbone's reasoning ability

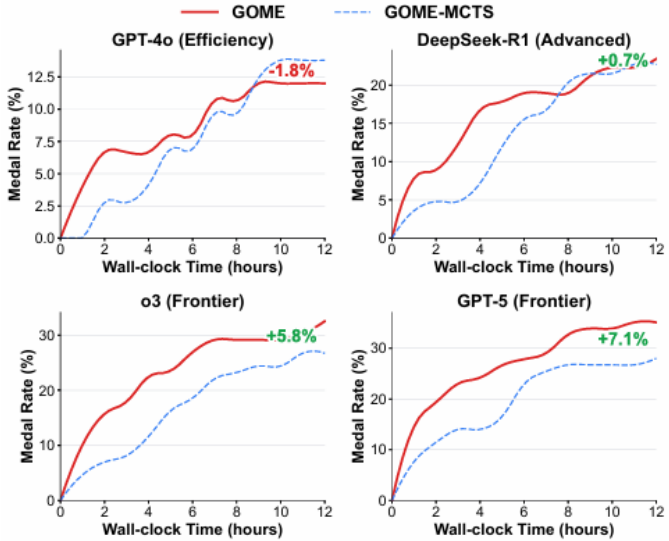


Agent	Any-Medal (%)
KompeteAI† (Gemini-2.5-flash,6h)	51.5±1.5
AIRA (o3,24h)	55.0±1.5
InternAgent-MLE† (DeepSeek-R1,12h)	62.1±3.0
FM Agent† (Gemini-2.5-pro,24h)	62.1±1.5
Thesis† (GPT-5-Codex,24h)	65.2±1.5
Leeroo† (Gemini-3-pro,24h)	**68.2±2.6**
GOME (GPT-5,12h)	**68.2±2.6**

Scaling effect of backbone's reasoning ability



(a) **Scaling with Model Capability.** As model capability increases from Efficiency to Frontier tiers, GOME's advantage over search-based baselines widens significantly.



(b) **Convergence Dynamics.** GOME (solid red) exhibits rapid early convergence, while MCTS (dashed blue) starts slower but catches up on weaker models. On Frontier-tier models, GOME maintains its advantage throughout.

Why not keep benchmark-driving?

- Too expensive…

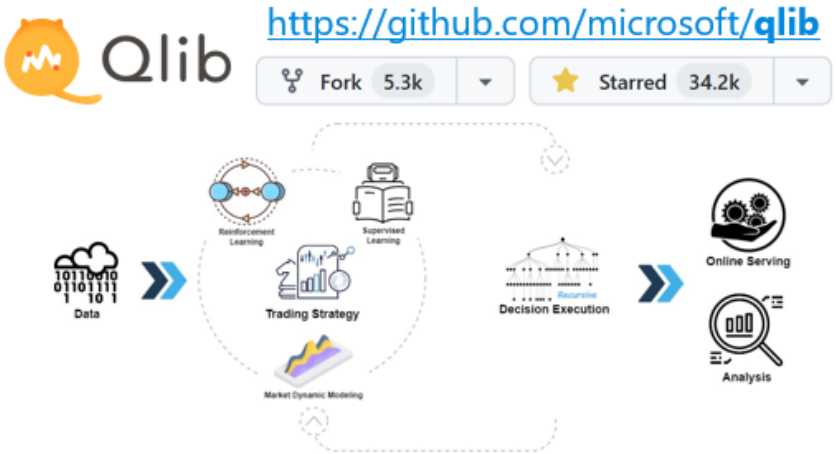
Table 8: Average computational cost per competition for R&D-Agent (GPT-5) across three runs (in USD)

Run	Research Phase (per competition)	Development Phase (per competition)	Total Cost (per competition)
Run 1	$4.68	$11.14	$15.82
Run 2	$7.85	$14.37	$22.22
Run 3	$8.43	$15.75	$24.18
Average	$6.99	$13.75	**$20.74**

X **75** competitions
X **5** runs to get averages
X **???** turns to play & fine-tune & test new idea…

Extend R&D-agent framework to Financial & scientific domains — RD-Q and Battery-Sim-Agent

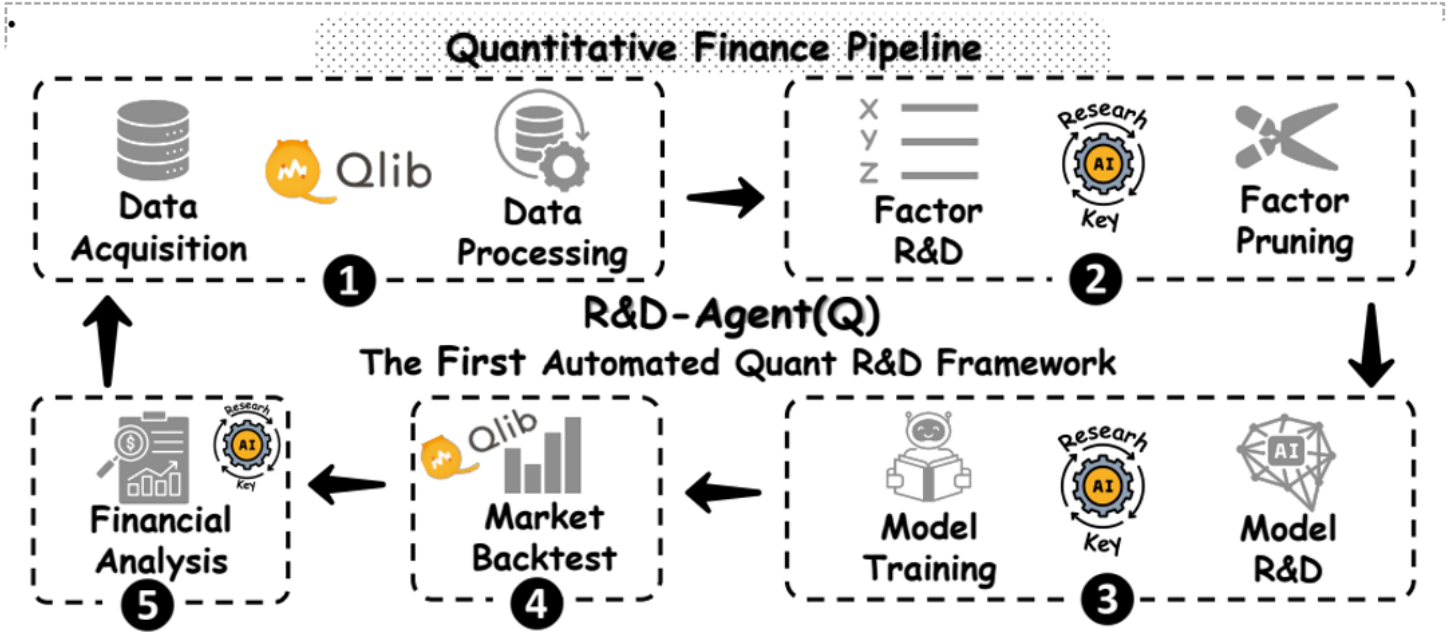
Our financial simulator:



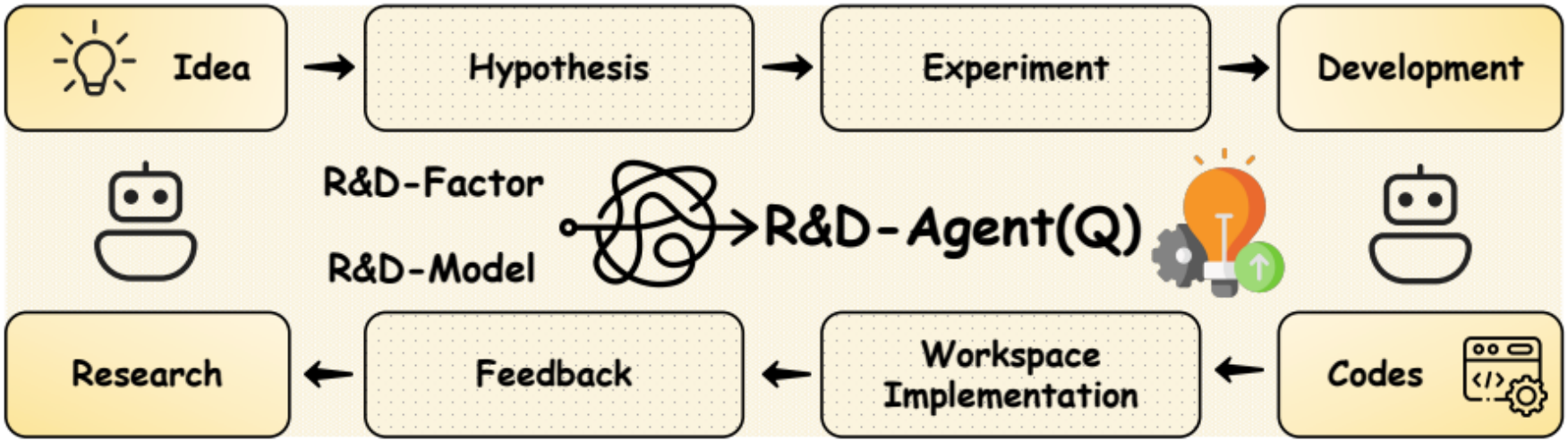
Qlib is an AI-oriented quantitative investment platform that aims to realize the potential, empower research, and create value using AI technologies in quantitative investment, from exploring ideas to implementing productions. Qlib supports diverse machine learning modeling paradigms. including supervised learning, market dynamics modeling, and RL.

Industry Engagement: R&D-Agent-Quant

➥Automate Quant Research & Development



2505.15155 / NeurIPS 2025


Idea
Hypothesis
Experiment
Development
R&D-Factor
R&D-Model
R&D-Agent(Q)
Research
Feedback
Workspace Implementation
Codes

❶ Specification Unit

Background → Data → Format → Mechanism

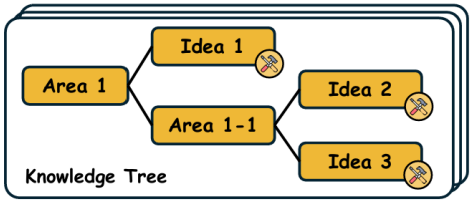
Dynamic Generation based on *Downstream Unit* and *Action*

------Scenario **B**ackground------
Quantitative investment is a data-driven approach to asset management ...
------Source **D**ataset------
daily_pv.h5: HDF5 file with [$open, $close, $volume, ...]; indexed by ...
------Output **F**ormat------
Output result.h5, a DataFrame with MultiIndex and one column named ...
-----Workflow **M**echanism-----
Qlib trains a model using past factor values to predict future returns, ...

❷ Synthesis Unit

Hypothesis Generation

Proposed Ideas



Tasks

❸ Implementation Unit

Co-STEER Agent

Factor Codes

Model Code

❹ Validation Unit

Accuracy

Similarity

Hyperparameter

Merge With SOTA Factor/Model Pool

Backtest in Qlib

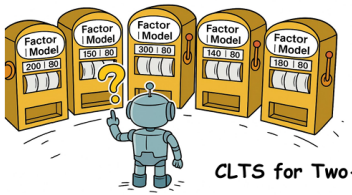
New SOTA

❺ Analysis Unit

Validation → Consistency → Performance → New Potential Idea

Next Iteration Action Choosing

Strategy Performance

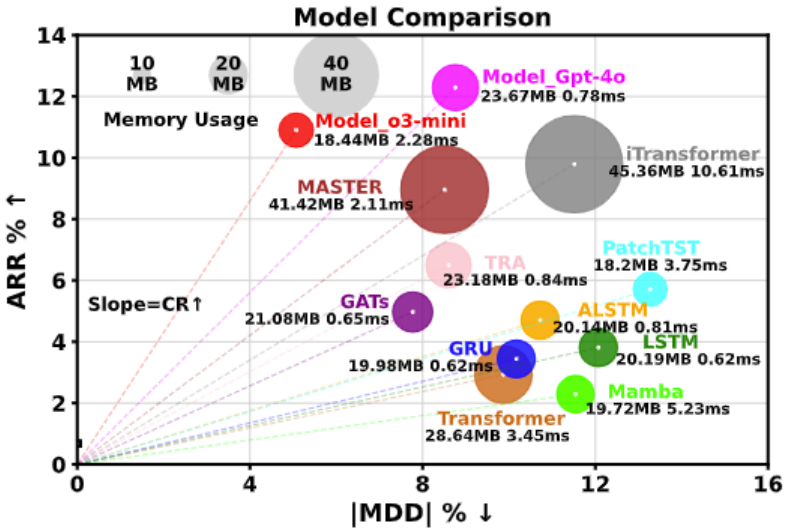


CLTS for Two-Armed Bandit

Action

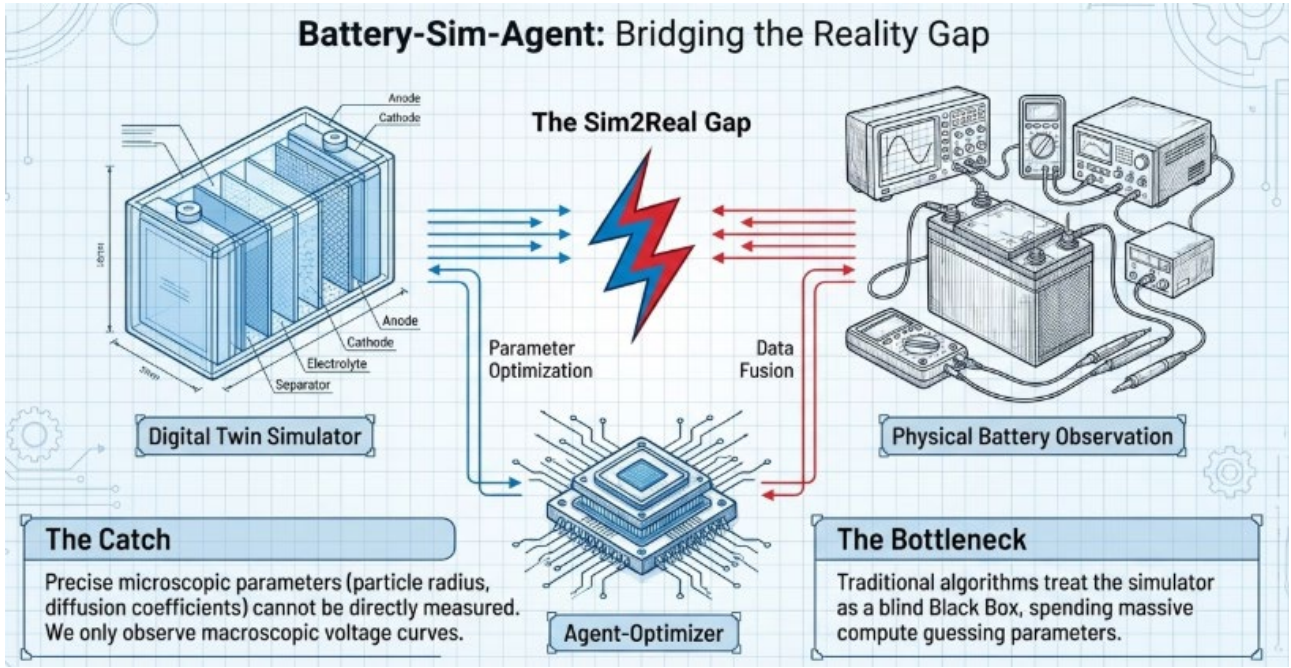
Table 1: Experimental results of all models on the CSI 300 constituent stock dataset, including factor predictive metrics and strategy performance metrics. Visual cues indicate ranking groups: **Best**, Second Best, Good (3–8), Average (9–14), Poor (15–20), and Worse (21–26).

Models		CSI300							
		Factor Predictive Power Metrics				Performance Metrics			
		IC	ICIR	Rank IC	Rank ICIR	ARR	IR	MDD	CR
Machine-Learning Models	Linear	0.0134	0.0992	0.0273	0.1962	-0.0302	-0.3710	-0.1987	-0.1520
	MLP	0.0291	0.2096	0.0412	0.3238	0.0003	0.0037	-0.1390	0.0022
	LightGBM	0.0277	0.2211	0.0386	0.3120	0.0397	0.5664	-0.0855	0.4643
	XGBoost	0.0291	0.2410	0.0384	0.3257	0.0316	0.4620	-0.1139	0.2774
	CatBoost	0.0279	0.2181	0.0393	0.3110	0.0513	0.7008	-0.0924	0.5552
	DoubleEnsemble	0.0294	0.2246	0.0417	0.3211	0.0551	0.7968	-0.0971	0.5675
Deep-Learning Models	Transformer	0.0317	0.2538	0.0434	0.3624	0.0293	0.4267	-0.0987	0.2969
	GRU	0.0315	0.2450	0.0428	0.3440	0.0344	0.5160	-0.1017	0.3382
	LSTM	0.0318	0.2367	0.0435	0.3389	0.0381	0.5561	-0.1207	0.3157
	ALSTM	0.0362	0.2789	0.0463	0.3661	0.0470	0.6992	-0.1072	0.4384
	GATs	0.0349	0.2511	0.0462	0.3564	0.0497	0.7338	-0.0777	0.6396
	PatchTST	0.0247	0.1945	0.0315	0.2463	0.0571	0.7191	-0.1327	0.4303
	iTransformer	0.0270	0.1946	0.0340	0.2365	0.0979	1.2337	-0.1151	0.8506
	Mamba	0.0281	0.2244	0.0374	0.2952	0.0229	0.3163	-0.1154	0.1984
	TRA	0.0404	0.3197	0.0490	0.4047	0.0649	1.0091	-0.0860	0.7547
	MASTER	0.0215	0.1925	0.0296	0.2486	0.0896	1.3406	-0.0851	1.0528
Factor Libraries	Alpha 101	0.0308	0.2588	0.0331	0.2749	0.0512	0.5783	-0.1253	0.4085
	Alpha 158	0.0341	0.2952	0.0450	0.3987	0.0570	0.8459	-0.0771	0.7393
	Alpha 360	0.0420	0.3290	0.0514	0.4225	0.0438	0.6731	-0.0721	0.6074
	AutoAlpha	0.0334	0.2656	0.0361	0.2967	0.0400	0.4288	-0.1225	0.3266
RD-Agent Series Framework*	RD-Factor$_{\text{GPT-4o}}$	0.0489	0.4050	0.0521	**0.4425**	**0.1461**	1.6835	-0.0750	**1.9468**
	RD-Factor$_{\text{o3-mini}}$	0.0497	0.3931	0.0500	0.4246	0.1184	1.3566	-0.0910	1.3016
	RD-Model$_{\text{GPT-4o}}$	0.0326	0.2305	0.0401	0.2767	0.1229	1.6676	-0.0876	1.4029
	RD-Model$_{\text{o3-mini}}$	0.0469	0.3688	**0.0546**	0.4385	0.1009	1.7009	**-0.0694**	1.4538
	RD-Agent(Q)$_{\text{GPT-4o}}$	0.0497	0.4069	0.0499	0.4122	0.1144	1.3167	-0.0811	1.4108
	RD-Agent(Q)$_{\text{o3-mini}}$	**0.0532**	**0.4278**	0.0495	0.4091	0.1421	**1.7382**	-0.0742	1.9150

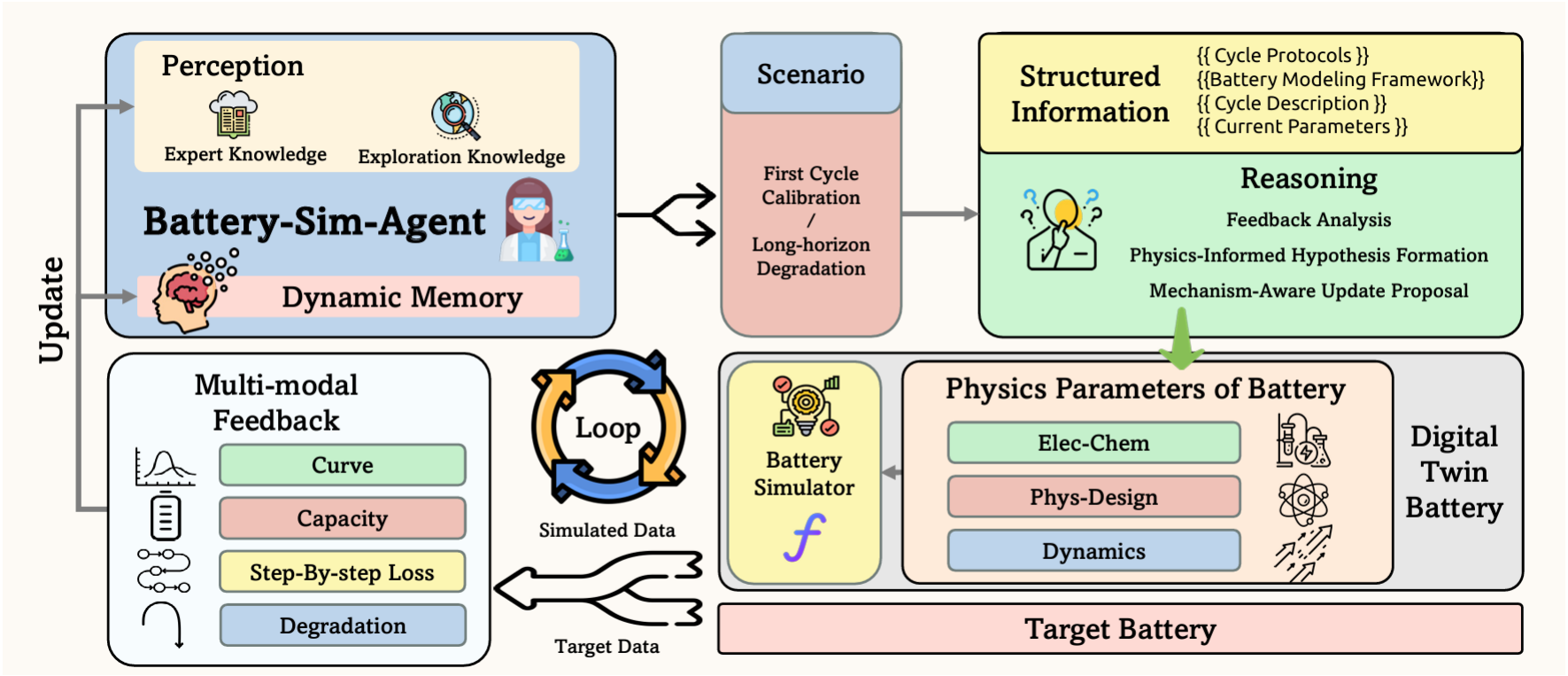


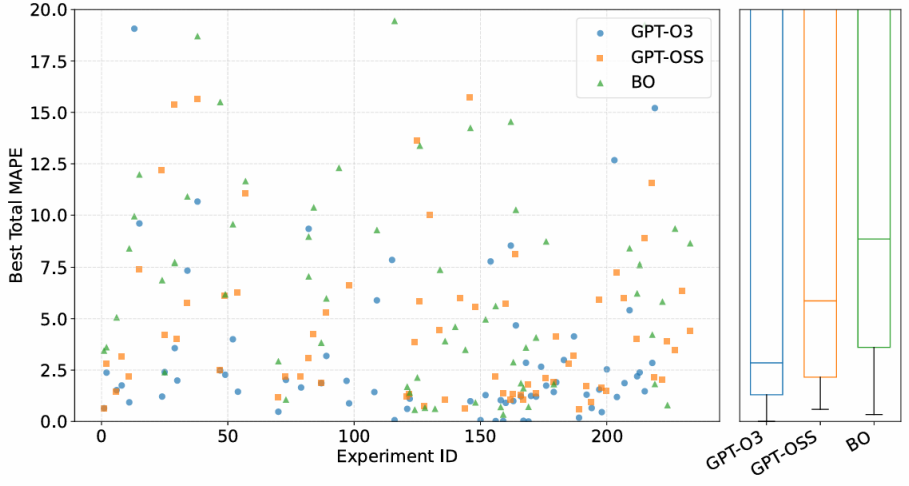
Battery-Sim-Agent

- Auto-built digital twin for battery

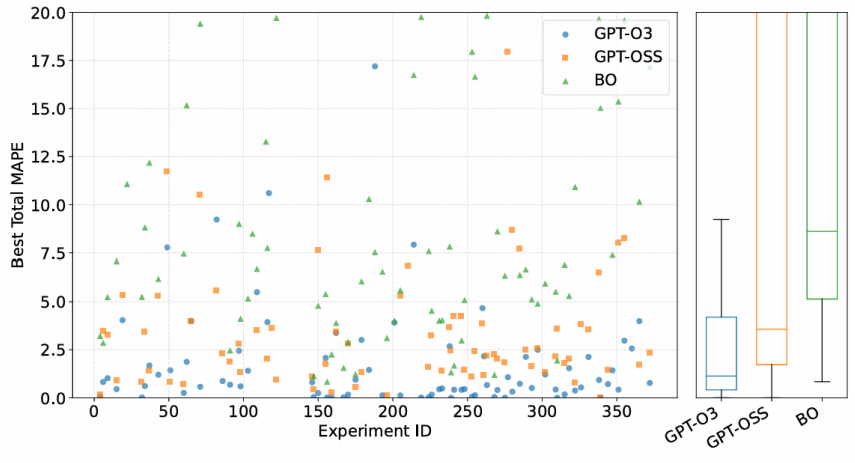


Battery-Sim-Agent



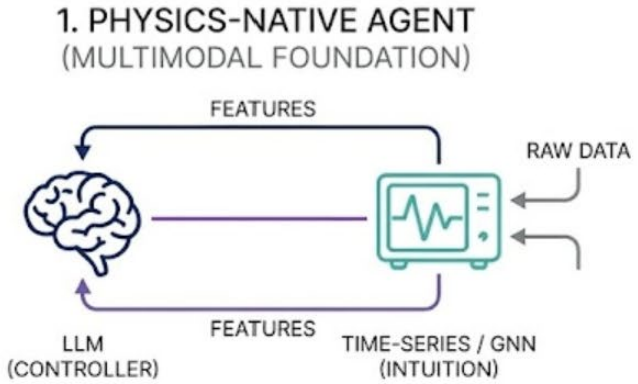


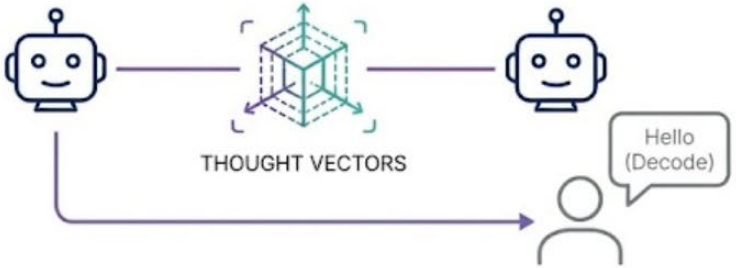
(a) Regular Mode



(b) Extreme Mode

Future Horizons





Thank You

Questions welcome

fsk@zju.edu.cn