# HETEROGENEOUS SCIENTIFIC FOUNDATION MODEL COLLABORATION

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Agentic large language model systems have demonstrated strong capabilities. However, their reliance on language as the universal interface fundamentally limits their applicability to many real-world problems, especially in scientific domains where domain-specific foundation models have been developed to address specialized tasks beyond natural language. In this work, we introduce *Eywa*, a heterogeneous agentic framework designed to extend language-centric systems to a broader class of scientific foundation models. The key idea of Eywa is to augment domain-specific foundation models with a language-model-based reasoning interface, enabling language models to guide inference over non-linguistic data modalities. This design allows domain-specific foundation models, which are typically optimized for specialized data and tasks, to participate in higher-level reasoning and decision-making processes within agentic systems. Eywa can serve as a drop-in replacement for a single-agent pipeline (*EywaAgent*) or be integrated into existing multi-agent systems by replacing traditional agents with specialized agents (*EywaMAS*). We further investigate a planning-based orchestration framework in which a planner dynamically coordinates traditional agents and Eywa agents to solve complex tasks across heterogeneous data modalities (*EywaOrchestra*). We evaluate Eywa across a diverse set of scientific domains spanning physical, life, and social sciences. Through effective collaboration with specialized foundation models, *Eywa* improves performance on domain-specific tasks, while reducing reliance on language-based reasoning, and therefore reducing token consumption.

## 1 INTRODUCTION

Large language models (LLMs) have driven the emergence of agentic AI systems that spark paradigm shifts across industries (OpenAI, 2023; Llama Team, 2024; Wei et al., 2026). Large-scale natural language pretraining endows these systems with general capabilities in perception, planning, reasoning, and decision-making (Chang et al., 2023; Feng et al., 2024). However, real-world problems are not limited to natural language. In scientific tasks involving specialized data types such as symbolic representations (e.g., formulas) and structured data (e.g., time series) (Hu et al., 2025; Shojaee et al., 2025), relying on natural language as the universal interface can become a significant bottleneck. This mismatch between language-centric pretraining and specialized scientific downstream tasks poses a key challenge to scientific agentic systems (Wei et al., 2025; Ríos-García et al., 2026).

Meanwhile, this limitation has coincided with rapid progress in developing domain-specific foundation models (FMs) for specialized data and tasks (Bommasani et al., 2021; Chen et al., 2023a; Kottapalli et al., 2025). These models, while not always equipped with language interfaces, are typically pretrained to capture domain-specific patterns and offer strong predictive capabilities within their domains (Lam et al., 2022; Zhu et al., 2024; Hamann et al., 2024; Herzog et al., 2025). This creates an opportunity to extend language-centric systems by enabling these heterogeneous foundation models to participate directly in reasoning processes, motivating the following research question:

***Can heterogeneous foundation models collaborate within agentic systems?***

This challenge can be viewed as a limitation of communication. Unlike existing agentic collaboration systems that assume all agents communicate through language (Guo et al., 2024; Yan et al., 2025), many foundation models do not natively support language as an input or output modality. This discrepancy makes it challenging to directly incorporate such foundation models into agentic systems.



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.

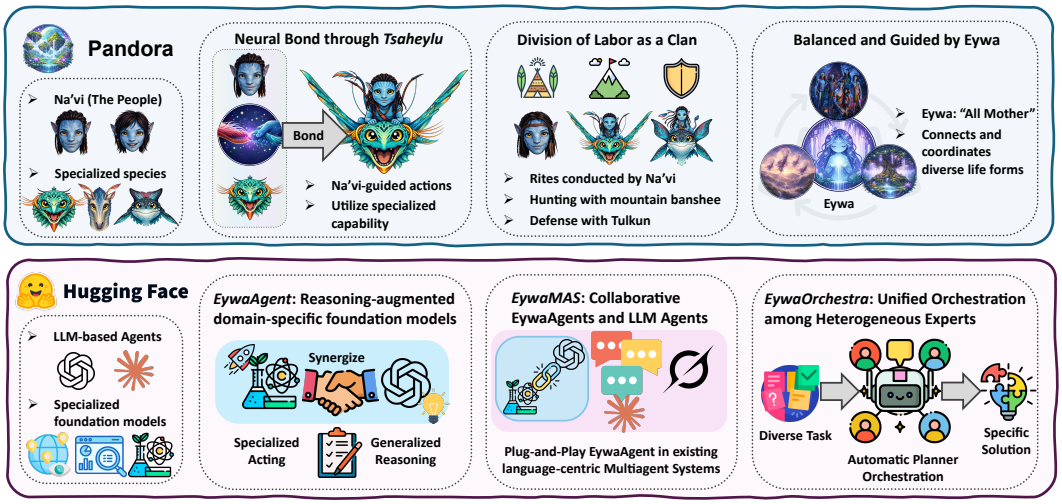



Figure 1: Analogy between Avatar Pandora and Agentic AI ecosystems. In Pandora, specialized species are coordinated by Na'vi through *Tsaheylu* (a neural bond for cross-species communication), collaborating under global guidance of *"All Mother"*. Inspired by this, we propose *Eywa*, a three-stage agentic framework: (1) *EywaAgent* builds an FM-LLM *Tsaheylu* interface and augments FMs with language-based reasoning interfaces; (2) *EywaMAS* enables collaboration between *EywaAgents* and LLM agents; and (3) *EywaOrchestra* dynamically orchestrates across heterogeneous experts.

To better illustrate this limitation and our motivation, in Figure 1, we draw an analogy from the movie *Avatar*[1]. On Pandora, many species possess highly specialized capabilities. Domain-specific foundation models are analogous to these specialized Pandora species. Yet such capabilities cannot be directly coordinated by Na'vi as they do not communicate through a shared symbolic language. Instead, the Na'vi establish connections through *Tsaheylu*[2], forming a direct neural interface that enables interaction even across fundamentally different biological systems.

In this paper, we introduce *Eywa*[3], a heterogeneous agentic framework to bridge language agents and domain-specific foundation models. Inspired by *Tsaheylu*, we propose to augment a domain-specific foundation model with a language model to create an *EywaAgent* (like a Na'vi warrior bonded with a Banshee). This design allows language agents to guide inference, planning, and decision-making of the foundation models over their specialized tasks. Building upon this primitive, we introduce *EywaMAS*, where *EywaAgents* can replace existing language agents in multi-agent systems. Moreover, we propose *EywaOrchestra*, a planning-based orchestration framework to dynamically coordinate both language agents and EywaAgents for complex tasks. Through this design, Eywa enables modality-native collaboration without requiring full translation into natural language.

We evaluate *Eywa* across a diverse set of scientific domains spanning physical, life, and social sciences. By effective modality-native collaboration, *Eywa* consistently improves the utility-cost trade-off over language-only baselines. *EywaAgent* and *EywaMAS* improve utility while reducing token and time usage in multi-agent settings. Moreover, *EywaOrchestra* dynamically orchestrates heterogeneous models and improves over single-agent and multi-agent baselines. Our major contributions are:

- We introduce *Eywa*, a heterogeneous agentic framework that enables modality-native collaboration by augmenting foundation models with language-model-based reasoning interfaces.
- We introduce three instantiations of the framework: *EywaAgent* (single-agent integration), *EywaMAS* (multi-agent extension), and *EywaOrchestra* (planning-based orchestration).
- We conduct extensive experiments across diverse scientific domains on our new benchmark *EywaBench* as well as other benchmarks, demonstrating improved performance by effectively integrating domain-specific foundation models into agentic systems.

[1] https://en.wikipedia.org/wiki/Avatar_(2009_film)
[2] https://james-camerons-avatar.fandom.com/wiki/Tsaheylu
[3] *Eywa*, a.k.a., "All Mother", is a planetary-scale network that connects and coordinates diverse life forms.

## 2 PRELIMINARY

**LLM Agent ("LLM").** An LLM agent (Wang et al., 2024a) is a policy $A_{\text{LLM}} : \mathcal{S} \to \Delta(\mathcal{M})$ where $\mathcal{S}$ is an internal state space, $\mathcal{M}$ is a response space and $\Delta(\mathcal{M})$ denotes a distribution over outputs.

**Domain-Specific Foundation Model ("FM").** While LLMs are often considered a class of foundation models, in this work we term *domain-specific foundation models* to refer to models that are primarily designed for specialized domains and do not necessarily provide a native language interface. An FM for domain $k$ is formulated as $F_k : \mathcal{X}_k \times \mathcal{U}_k \to \mathcal{O}_k$ where $\mathcal{X}_k$ is the input, $\mathcal{U}_k$ is a space of structured configuration arguments, and $\mathcal{O}_k$ is an output space.

**Multi-Agent Systems (MAS).** A multi-agent system is defined as a tuple $\mathcal{M} = (\mathcal{A}, \mathcal{G})$ where $\mathcal{A} = \{A_1, A_2, \ldots, A_n\}$ is a set of agents, and $\mathcal{G}$ denotes the communication topology. Each agent $A_i$ is a policy $A_i : \mathcal{S}_i \to \Delta(\mathcal{M}_i)$ where $\mathcal{S}_i$ is the local state space and $\mathcal{M}_i$ is the message space. Let $s_i^{(t)} \in \mathcal{S}_i$ denote the local state of agent $A_i$ at step $t$, and let $m_i^{(t)} \in \mathcal{M}_i$ denote the message produced by $A_i$ at step $t$; correspondingly, $m_{j,i}^{(t)}$ denotes the message from $A_j$ to $A_i$, and $m_{-i}^{(t)}$ collects all messages received by $A_i$. At each step, agent $A_i$ updates its state and produces a message:

$$s_i^{(t)} = \text{Update}_i\big(s_i^{(t-1)}, m_{-i}^{(t)}\big), \quad m_i^{(t)} \sim A_i\big(s_i^{(t)}\big). \tag{1}$$

The interaction proceeds iteratively until a final output $\hat{y}$ after a finite number of steps.

**Problem Formulation.** Let $\mathcal{T}$ denote a family of tasks. Each task instance is represented by $\tau = (q, x, y^\star, \ell)$ where $q \in \mathcal{Q}$ is a natural-language instruction, $x \in \mathcal{X}$ is the task input, $y^\star \in \mathcal{Y}$ is the desired output, and $\ell$ is a task-specific loss function. We make a reasonable assumption for scientific tasks that the input space factorizes as $\mathcal{X} = \mathcal{X}_{\text{lng}} \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_m$ where $\mathcal{X}_{\text{lng}}$ denotes language-observable context and each $\mathcal{X}_k$ denotes a domain-specific input. An agentic system $G$ produces an output $\hat{y}_G(\tau) = G(q, x)$ for task $\tau$. The objective is to minimize the expected task loss over the task distribution: $\min_G \ \mathbb{E}_{\tau \sim \mathcal{T}} \left[\ell\big(\hat{y}_G(\tau), y^\star\big)\right]$.

## 3 EYWAAGENT: REASONING FOUNDATION MODEL AGENTS

Our first step towards the Eywa agentic framework is to introduce *EywaAgent*, a unified abstraction that augments a foundation model with a language-based reasoning interface. The key idea is similar to "Tsaheylu" in Avatar to create a strong bond between a language model that performs high-level planning and control and a domain-specific foundation model that provides specialized capabilities.

### 3.1 FM-LLM "TSAHEYLU" BOND

The objective of the FM–LLM "Tsaheylu" is to establish a robust and stable communication channel between a domain-specific foundation model $F_k$ for domain $k$ and a language model $A_{\text{LLM}}$. The Tsaheylu interface is designed to ensure that: (1) the LLM can correctly configure the control input $\mathcal{U}_k$ conditioned on the task state, (2) specialized computation is delegated to the foundation model, and (3) the resulting output $\mathcal{O}_k$ can be faithfully reintegrated into the language reasoning process.

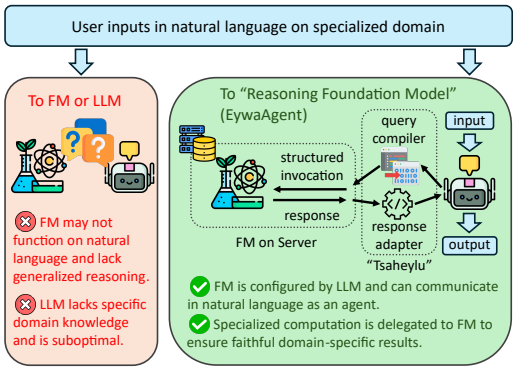



Figure 2: Reasoning FM Agent (EywaAgent) leverages both generalized reasoning and specialized acting through FM-LLM "Tsaheylu" Bond.

To this end, we formalize the FM–LLM Tsaheylu as a bidirectional communication interface. For each domain $k$, we define an interface pair $(\phi_k, \psi_k)$. $\phi_k : \mathcal{S} \to \mathcal{U}_k$ is a *query compiler* that translates the task state into a structured FM invocation; $\psi_k : \mathcal{O}_k \to \mathcal{Z}_k$ is a *response adapter* that converts the specialist output to be LLM-consumable, where $\mathcal{Z}_k$ denotes a structured context space compatible with language reasoning.

The resulting communication pipeline can be expressed as

$$\text{input task } \tau \xrightarrow{\text{interpretation by } A_{LLM}} s \xrightarrow{\phi_k} u_k \xrightarrow{F_k(\cdot)} o_k \xrightarrow{\psi_k} z_k \xrightarrow{\text{synthesis by } A_{LLM}} \text{output response } \hat{y}$$

which enables seamless integration of heterogeneous foundation models into the language-centric reasoning loop and forms the building block of our Eywa agentic framework.

**Tsaheylu Context Protocol.** Tsaheylu interface is implemented through *Tsaheylu Context Protocol* for structured FM-LLM interaction. Each foundation model $F_k$ is exposed as a *remote service* with a well-defined schema over its input space $\mathcal{U}_k$ and output space $\mathcal{O}_k$. (i) *Handshake*. LLM first sends a capability query to $F_k$, which returns FM specifications. (ii) *Invocation*. The query compiler $\phi_k$ specifies the invocation parameters and execution constraints. Upon invocation, the remote FM server executes the requested operation by applying the foundation model $F_k$ with configuration $u_k$, returning structured outputs $o_k$. (iii) *Adaptation*. The response adapter $\psi_k$ then transforms $o_k$ into a language-compatible representation $z_k$, which is appended to the subsequent reasoning.

### 3.2 EywaAgent for Generalized Reasoning and Specialized Acting

Building upon the FM-LLM Tsaheylu interface, we define *EywaAgent* as a unified agent that combines language-based reasoning with domain-specific computation. Unlike conventional language agents that operate purely in the linguistic space, EywaAgent dynamically adapts based on task requirements. Instead of a single standalone model, an *EywaAgent* is a coupled FM-LLM agentic unit. The two components are connected through the Tsaheylu interface and jointly define the agent's behavior.

**Definition 1** (EywaAgent)**.** *An EywaAgent is defined as a tuple*

$$A_{\text{eywa}} = (A_{\text{LLM}}, F, \phi, \psi, \mathcal{C}) \tag{2}$$

*where $A_{\text{LLM}} : \mathcal{S} \to \Delta(\mathcal{M})$ is a language model operating over the agent state space $\mathcal{S}$, $F : \mathcal{X} \times \mathcal{U} \to \mathcal{O}$ is a domain-specific foundation model, and $(\phi, \psi)$ defines a bidirectional communication interface between the language space and the domain space, with $\phi : \mathcal{S} \to \mathcal{U}$ and $\psi : \mathcal{O} \to \mathcal{Z}$. The control policy $\mathcal{C} : \mathcal{S} \to \{\texttt{invoke}, \texttt{skip}\}$ determines whether and how the foundation model is invoked. If not specified, $\mathcal{C}$ is induced by the language model through its reasoning over the current state.*

An illustration of EywaAgent is shown in Figure 2. At each reasoning step $t$, given state $s^{(t)}$, the control policy produces a decision $a^{(t)} \sim \mathcal{C}(s^{(t)}), \quad a^{(t)} \in \{\texttt{invoke}, \texttt{skip}\}$.

If $a^{(t)} = \texttt{skip}$, the agent reduces to a standard language-only reasoning step:

$$z^{(t)} = A_{\text{LLM}}(s^{(t)}). \tag{3}$$

If $a^{(t)} = \texttt{invoke}$, the agent executes the Tsaheylu pipeline:

$$u = \phi(s^{(t)}), \quad o = F(x, u), \quad z^{(t)} = \psi(o). \tag{4}$$

The updated state is given by $s^{(t+1)} = s^{(t)} \cup \{z^{(t)}\}$. This adaptive mechanism enables EywaAgent to seamlessly switch between generalized reasoning and specialized acting. From this perspective, EywaAgent subsumes language-only agents as a special case (by choosing $\mathcal{C}$ to always skip), while strictly expanding the space of computable functions through access to domain-specific foundation models. Consequently, as we show in Appendix Theorem 13, EywaAgent achieves both enhanced expressivity and improved task performance. Moreover, by delegating computation to the foundation model, *Eywa* avoids explicit token-level reasoning over structured data and reduces token usage.

## 4 Eywa Systems: Multi-Agent Composition and Orchestration

With EywaAgent being a plug-and-play building block for agentic AI systems, we naturally extend this paradigm to multi-agent settings to enable heterogeneous collaborations. We introduce two system-level abstractions. First, *EywaMAS* generalizes EywaAgent to a distributed multi-agent setting, allowing specialized agents to interact and collaborate. Second, *EywaOrchestra* introduces a global orchestration mechanism that dynamically coordinates agents through structured planning.

### 4.1 EywaMAS: Plug-and-Play Composition with EywaAgent

**Definition 2** (EywaMAS). *An EywaMAS is defined as a multi-agent system*

$$\mathcal{M}_{\text{Eywa}} = (\mathcal{A}, \mathcal{G}) \tag{5}$$

*where $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ is a set of heterogeneous agents, each of which is either an LLM agent or an EywaAgent, and $\mathcal{G}$ specifies the communication topology.*

*EywaMAS* is a multi-agent system that composes heterogeneous agents in a plug-and-play manner. Unlike traditional multi-agent systems that rely solely on language-based agents, EywaMAS enables seamless integration of both language-only agents and EywaAgents within a unified framework. In practice, constructing an EywaMAS requires minimal modification to existing multi-agent architectures. Specifically, one can replace a subset of language-only agents with EywaAgents while keeping the overall system structure unchanged.

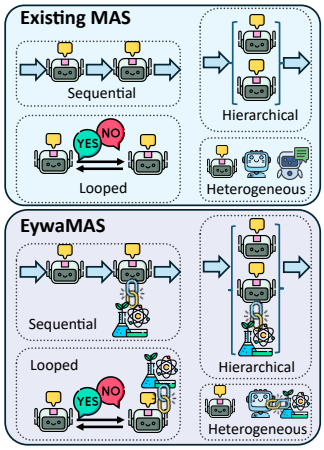



Figure 3: EywaMAS generalizes existing multi-agent systems with EywaAgents.

Each EywaAgent $\mathcal{A}_k$ internally follows the mechanism defined in Section 3. At the system level, EywaMAS follows the same communication and state update dynamics as standard MAS governed by the communication topology $\mathcal{G}$. It supports flexible composition across (1) different language models with varying scales or capabilities, (2) different foundation models across domains, and (3) mixed agent types (LLM Agents and EywaAgents). When $\mathcal{A} = \{\mathcal{A}_1, \ldots, \mathcal{A}_n\}$ are all LLM agents, EywaMAS degenerates to existing multi-agent systems. Similar to Theorem 13, we establish a theoretical result showing that EywaMAS strictly improves over language-only multi-agent systems in Appendix A.4.

### 4.2 EywaOrchestra: Dynamic Orchestration of Heterogeneous Experts

Real-world tasks are highly diverse. Different tasks may require different mixtures of language-only reasoning, domain-grounded prediction, and inter-agent collaboration patterns. To address this, we extend *Eywa* from fixed systems to a dynamic orchestration framework, termed *EywaOrchestra*.

**Definition 3** (EywaOrchestra). *Given candidate LLMs $\mathcal{M}_{\text{LLM}}$, candidate FMs $\mathcal{M}_{\text{FM}}$, and a topology pool $\Pi$, EywaOrchestra is a dynamically instantiated heterogeneous multi-agent system:*

$$\mathcal{O} = (\mathcal{C}, P) \tag{6}$$

*where $\mathcal{C}$ is the configuration space induced by $(\mathcal{M}_{\text{LLM}}, \mathcal{M}_{\text{FM}}, \Pi)$, $P$ is the conductor.*

Planning in *EywaOrchestra* is achieved by a *conductor* that, conditioned on the input task, dynamically decides: (i) the role and type of each agent, e.g., whether an agent should be a language-only agent or an EywaAgent; (ii) the backbone language model used by each agent; (iii) the domain-specific foundation model of each EywaAgent; and (iv) the communication topology of the overall multi-agent system. After the conductor configuration, the instantiated system is executed to solve the task.

The motivation of *EywaOrchestra* is to move beyond static designs by enabling task-adaptive system construction, jointly leveraging *model adaptivity*, through selecting language and domain-specific foundation models, and *structural adaptivity*, through selecting the communication topology itself. In Appendix Theorem 16, we prove the improvement of *EywaOrchestra* over static systems.

## 5 Experiments

### 5.1 EywaBench: A Scalable Multi-task Multi-domain Scientific Benchmark

Current scientific benchmarks are often limited to a narrow task family (Wu et al., 2017; M-A-P Team, 2025), a single domain (Dai et al., 2025; Qiu et al., 2025a), or one data format (Yin et al., 2025; Hasson et al., 2025; Wang et al., 2025a). More specifically, two important scientific modalities, time series and tabular data, are often ignored or poorly evaluated. To provide a holistic evaluation of agentic

Table 1: Overall performance comparison across scientific domains on *EywaBench*. We compare all methods on utility (↑), inference time (↓), and token consumption (↓). Best results are highlighted in bold and second-best results are underlined. Our proposed methods, EywaAgent, EywaMAS, and EywaOrchestra, achieve strong overall performance while maintaining competitive efficiency.

| Method | Metrics | Physical Science | | | Life Science | | | Social Science | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Material | Energy | Space | Biology | Clinic | Drug | Economy | Business | Infrastructure | Overall |
| ***Single-Agent Setting*** | | | | | | | | | | | |
| **Single-LLM-Agent** | Utility (↑) | 0.5616 | 0.8202 | 0.5235 | 0.3402 | 0.4582 | 0.6004 | 0.7689 | 0.6528 | 0.6758 | 0.6154 |
| | Time (↓) | 34.48 | 27.01 | 26.00 | 34.68 | 22.37 | 21.13 | 22.67 | 22.28 | 18.42 | 25.22 |
| | Tokens (↓) | 6367 | 4854 | 4512 | 6164 | 3618 | 3571 | 4097 | 3915 | 3327 | 4469 |
| **EywaAgent (Ours)** | Utility (↑) | 0.5871 | 0.8390 | 0.6123 | 0.3718 | 0.5085 | 0.6199 | **0.8048** | 0.7371 | 0.7060 | 0.6558 |
| | Time (↓) | 34.88 | 24.42 | 23.12 | 30.84 | 20.32 | 15.84 | 19.71 | 20.98 | 15.99 | 22.78 |
| | Tokens (↓) | 5040 | 3167 | 3329 | 4858 | 2333 | 2210 | 2791 | 2444 | 2248 | 3137 |
| ***Multi-Agent Setting*** | | | | | | | | | | | |
| **Refine MAS (2023)** | Utility (↑) | 0.5687 | 0.8667 | 0.6244 | 0.3623 | 0.4504 | 0.6215 | 0.7523 | 0.6880 | 0.6362 | 0.6294 |
| | Time (↓) | 72.76 | 64.22 | 79.65 | 75.21 | 51.89 | 50.63 | 62.33 | 48.54 | 47.49 | 60.59 |
| | Tokens (↓) | 11013 | 9009 | 10043 | 10497 | 7029 | 7498 | 8924 | 6997 | 7438 | 8673 |
| **Debate MAS (2024)** | Utility (↑) | 0.5602 | 0.8656 | 0.6543 | 0.3438 | 0.4738 | 0.6198 | 0.7729 | 0.6907 | 0.7237 | 0.6460 |
| | Time (↓) | 82.06 | 79.46 | 74.75 | 101.64 | 78.19 | 63.98 | 92.72 | 72.46 | 60.73 | 78.22 |
| | Tokens (↓) | 16652 | 14278 | 13614 | 17007 | 11159 | 10447 | 14694 | 10953 | 10311 | 13216 |
| **MoA (2025b)** | Utility (↑) | 0.5909 | 0.8069 | 0.5863 | 0.3580 | 0.4722 | 0.5686 | 0.7499 | 0.7004 | 0.6938 | 0.6273 |
| | Time (↓) | 90.15 | 56.95 | 69.32 | 59.10 | 46.53 | 44.31 | 57.35 | 48.29 | 47.34 | 57.75 |
| | Tokens (↓) | 25327 | 16453 | 17332 | 15980 | 11014 | 10344 | 16114 | 11690 | 12365 | 15317 |
| **X-MAS (2025)** | Utility (↑) | 0.5831 | 0.8057 | 0.5723 | 0.3737 | 0.4490 | 0.6211 | 0.6923 | 0.6390 | 0.7180 | 0.6188 |
| | Time (↓) | 104.48 | 86.63 | 79.06 | 88.20 | 67.94 | 59.76 | 75.50 | 72.82 | 62.95 | 77.42 |
| | Tokens (↓) | 24149 | 19808 | 16584 | 18451 | 12549 | 11907 | 16499 | 14007 | 14056 | 16537 |
| **EywaMAS (Ours)** | Utility (↑) | **0.6381** | **0.8742** | 0.6899 | **0.3798** | 0.5086 | 0.6248 | 0.7959 | 0.7284 | **0.7406** | **0.6761** |
| | Time (↓) | 77.25 | 75.96 | 72.51 | 111.92 | 59.97 | 59.23 | 68.40 | 58.11 | 46.49 | 72.11 |
| | Tokens (↓) | 14529 | 11709 | 11787 | 16502 | 9407 | 8078 | 11044 | 9470 | 8912 | 11214 |
| ***Dynamic Orchestration*** | | | | | | | | | | | |
| **EywaOrchestra (Ours)** | Utility (↑) | 0.6249 | 0.8711 | **0.7187** | 0.3682 | **0.5159** | **0.6319** | 0.7830 | **0.7388** | 0.7298 | 0.6746 |
| | Time (↓) | 61.78 | 39.92 | 75.47 | 67.88 | 45.38 | 45.94 | 49.13 | 34.18 | 28.80 | 48.16 |
| | Tokens (↓) | 11535 | 7723 | 10810 | 11315 | 7050 | 6495 | 7117 | 7264 | 6892 | 8335 |

systems in scientific settings, we introduce *EywaBench*, a scalable benchmark for multi-task and multi-domain scientific reasoning across heterogeneous modalities. *EywaBench* is constructed from a collection of datasets, including but not limited to DeepPrinciple (Song et al., 2025), MMLU-Pro (Wang et al., 2024b), fev-bench (Shchur et al., 2025), and TabArena (Erickson et al., 2025).

**Multi-task and multi-domain coverage.** *EywaBench* includes tasks spanning natural language, time series, and tabular data. The samples are organized into three domains: physical science, life science, and social science. Details of domains and sub-domains are provided in Appendix C.

**Scale and evaluation metric.** We provide two variants: (i) *EywaBench-full* with 5,002 instances and (ii) *EywaBench-mini* for cost-efficient evaluation. In Appendix C.4, we show that EywaBench-mini is a representative subset for accurate and efficient evaluation. *EywaBench* uses a unified utility score $u \in [0, 1]$ across all tasks. For natural-language-centered tasks, utility is computed with a soft-match score between predictions and references; for time-series and tabular tasks, utility is derived from normalized prediction errors. Detailed metric definitions are provided in the Appendix C.

### 5.2 Experimental Setup

**Language models and foundation models.** We use gpt-5-nano as the default language model and we also evaluate other models from and beyond the GPT family in our later experiments. In our main experiment, we have two foundation models to build EywaAgents. Chronos (Ansari et al., 2024; 2025) is a general-purpose foundation model for time series. TabPFN (Hollmann et al., 2023) is a transformer-based foundation model for tabular prediction problems. Neither foundation model provides a native language interface. In Appendix D.3, we further extend the pool with five additional time-series and tabular foundation models.

**Baseline methods.** We compare *Eywa* against three baseline groups: (a) *single-agent LLM baselines* built on the same backbone; (b) *homogeneous LLM-based multi-agent baselines*, including Refine (Madaan et al., 2023) and Debate (Du et al., 2024), instantiated with the same backbone model; (c)

*heterogeneous LLM-based multi-agent baselines*, including Mixture-of-Agents (MoA) (Wang et al., 2025b) and X-MAS (Ye et al., 2025), which combine multiple heterogeneous language models.

**Device details.** For all runs, we record wall-clock latency and token usage. The results are averaged over multiple runs on a 13th Gen Intel(R) Core(TM) i9-13900H CPU with 64GB RAM.

### 5.3 Main Results

We evaluate all methods under a unified protocol. For *EywaAgent* and *EywaMAS*, which do not perform dynamic orchestration, human experts assign the foundation model and annotate the *EywaMAS* topology based on the task description, with debate as the default. Human experts never observe model outputs or ground-truth labels. For *EywaOrchestra*, the conductor automatically selects the foundation model and topology conditioned on the sample.

Table 1 reports the performance of all methods on *EywaBench-mini* in scientific settings. We highlight the following observations: (a) *EywaAgent* improves both quality and efficiency under the same backbone. Compared with the corresponding single-agent baseline, *EywaAgent* increases average utility by 6.6%, while reducing latency and cutting token usage by nearly 30% through delegation to domain-specific foundation models. (b) *EywaMAS* outperforms homogeneous MAS baselines in scientific settings. *EywaMAS* achieves the best overall utility and outperforms homogeneous multi-agent baselines. Compared with *Refine*, *EywaMAS* delivers stronger utility. Compared with *Debate*, *EywaMAS* not only achieves better utility but also requires fewer tokens.

(c) LLM-only heterogeneity is insufficient for scientific tasks. Heterogeneous LLM-only MAS methods do not consistently outperform strong homogeneous MAS baselines. This suggests that, for scientific workloads, cross-modality heterogeneity is more critical than only combining heterogeneous language models. (d) Not every domain benefits equally from heavier multi-agent computation. In domains such as economy and business, single-agent *EywaAgent* is already highly competitive, indicating that always using complex multi-agent topologies is not necessarily optimal. This observation motivates adaptive orchestration conditioned on task and domain characteristics. (e) *EywaOrchestra* approaches EywaMAS with lower cost and automation. *EywaOrchestra* uses no expert configuration and instead lets the conductor automatically construct the system for each sample. Despite this, *EywaOrchestra* reaches utility close to expert-designed *EywaMAS*, and even surpasses it on several sub-domains. At the same time, dynamic orchestration substantially reduces inference cost in both latency and token usage compared to fixed multi-agent systems.

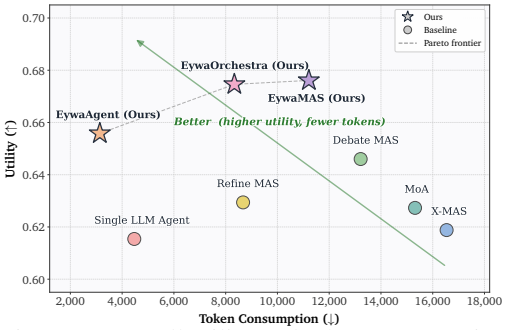


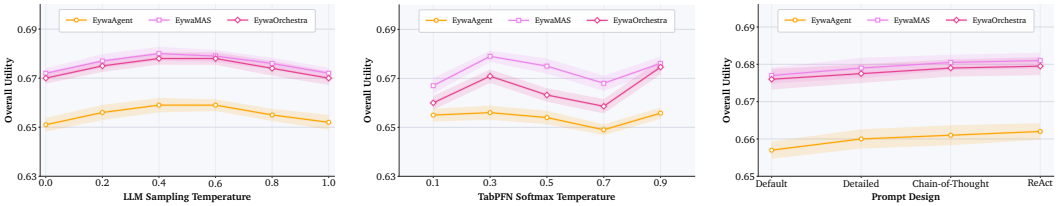

Figure 4: Overall utility and token consumption across methods. Full results in Figure 14.

### 5.4 Further Analysis

**Hyperparameter sensitivity.** We perform hyperparameter studies of both language models and foundation models to evaluate the robustness of *Eywa*. Specifically, we vary the LLM sampling temperature, the foundation model temperature in TabPFN, and the prompt design used by the language-model interface. As shown in Figure 5, *Eywa* remains stable across a broad range of configurations. In particular, the performance of *EywaAgent*, *EywaMAS*, and *EywaOrchestra* stays robust as the LLM sampling temperature varies, while consistently peaking around a moderate temperature. Ablation on varying the TabPFN softmax temperature also suggests that the benefit of integrating domain-specific foundation models is robust to foundation-model calibration. We further compare several prompt designs. The *Detailed* prompt provides more comprehensive task descriptions and general guidance, *Chain-of-Thought* (Wei et al., 2022) encourages the model to

Table 2: Ablation results of *EywaAgent* with different LLM backends. *Eywa* benefits from more powerful LLMs.

| Method | Metrics | Physical | Life | Social | Overall |
|---|---|---|---|---|---|
| gpt-4.1-nano | Utility (↑) | 0.6547 | 0.4010 | 0.6269 | 0.5680 |
| | Time (↓) | 14.25 | 28.66 | 16.89 | 19.61 |
| | Tokens (↓) | 1314 | 927 | 1160 | 1139 |
| gpt-5-nano | Utility (↑) | 0.6914 | 0.5001 | 0.7488 | 0.6558 |
| | Time (↓) | 28.04 | 22.33 | 18.71 | 22.78 |
| | Tokens (↓) | 3907 | 3134 | 2491 | 3137 |
| gpt-5-mini | Utility (↑) | 0.7191 | 0.5035 | 0.7444 | 0.6640 |
| | Time (↓) | 23.18 | 30.70 | 18.51 | 23.63 |
| | Tokens (↓) | 2875 | 2617 | 1949 | 2444 |




(a) LLM Temperature Ablation. (b) FM Temperature Ablation. (c) Prompt Design Ablation.

Figure 5: Hyperparameter sensitivity analysis of *Eywa*. We evaluate *EywaAgent*, *EywaMAS*, and *EywaOrchestra* under different ablation configurations. **(a)** LLM temperature ablation: *Eywa* maintains stable performance across varying LLM sampling temperatures. **(b)** FM temperature ablation: *Eywa* remains robust under different TabPFN softmax temperatures. **(c)** Prompt design ablation: *Eywa* remains effective across different prompting strategies, and generally benefits from more structured prompt designs. Overall, *Eywa* is compatible and robust to a broad range of design choices.

reason step by step, and *ReAct* (Yao et al., 2023) further interleaves reasoning with actions, enabling the model to decide more explicitly when to invoke tools or domain-specific foundation models during problem solving. While *Eywa* performs well under all prompting strategies, more structured prompts generally lead to slightly better utility. Overall, these results demonstrate that the gains of *Eywa* are not tied to a particular hyperparameter choice or prompt template. Additional ablation studies are provided in Appendix D.1.

**Ablation on *Eywa* with different LLM backends.** To evaluate the impact of LLM backbone choice, we instantiate *EywaAgent* with three backends from OpenAI GPT family. As shown in Table 2, *Eywa* remains consistently effective across physical, life, and social science domains under all three backbones. We observe an upward trend in overall utility as backend capability increases, indicating that *Eywa* is robust to backbone selection while still benefiting from stronger LLM priors. Complete backbone ablations for *EywaMAS* and *EywaOrchestra* are reported in Appendix Table 10 and Table 14. *EywaOrchestra* consistently outperforms the single-agent baseline with various LLM backbones.

***Eywa* benefits from stronger domain-specific model.** Our experiments replacing Chronos in *EywaOrchestra* with weaker models indicate that *Eywa* benefits from stronger pretrained domain-specific models. Details are provided in Appendix D.2.

**Case study.** In Appendix D.5, we provide qualitative case studies to further illustrate how *Eywa* coordinates language models and domain-specific foundation models. Case Study A compares a language-only LLM agent with *EywaAgent* on the same task over structured financial signals. Case Study B further illustrates the role of *EywaOrchestra*.

### 5.5 Eywa on Downstream Scientific Applications

We further apply *Eywa* to three downstream scientific applications from PFMBench (Gao et al., 2025): (i) *metal-ion binding* (Hu et al., 2022), a binary classification task on whether a protein binds metal ions; (ii) *protein-ligand binding* (Liu et al., 2007), a regression task that predicts the binding affinity between a protein and a ligand; and (iii) *enzyme catalytic efficiency* (Li et al., 2022), a regression task that predicts the catalytic efficiency of an enzyme toward a substrate. The latter two tasks take both a protein sequence and a molecule in SMILES format as input. These applications introduce two new modalities, amino acid sequences and SMILES strings, together with two new foundation models, ESM-2 (Lin et al., 2023) for proteins and ChemBERTa (Chithrananda et al., 2020) for molecules.

Table 3: Results on downstream scientific applications. We report AUROC for metal-ion binding and Spearman's $\rho$ for the other two tasks.

| Method | Metal-ion | Ligand | Enzyme Efficiency |
|---|---|---|---|
| gpt-5-nano | 0.5251 | 0.0026 | 0.0032 |
| Inkling-975B | 0.5468 | 0.0083 | 0.0179 |
| *EywaOrchestra* (Ours) | **0.6094** | **0.0572** | **0.2581** |

Each test instance contains labeled examples from the training split and query samples from the test split. The test sets contain 47, 56, and 64 instances, respectively. We compare with language-only single agents built on gpt-5-nano and Inkling-975B, which receive labeled examples in the prompt. We report AUROC for metal-ion binding and Spearman's $\rho$ for the two regression tasks. As shown in

Table 3, language-only agents perform close to chance (AUROC near 0.5 and $\rho$ near 0), even with the frontier-scale Inkling-975B, as amino acid sequences and SMILES strings carry little usable signal when read as text. In contrast, *EywaOrchestra* improves over both baselines on all three applications.

## 6 Related Work

In this section, we review the key related works on the topics that are closely related to this work. We put more related works and discussions in the Appendix B.1 to keep the main text concise.

**Scientific Large Language Models and Foundation Models.** Scientific large language models have emerged for scientific knowledge understanding and domain-specific question answering (Hu et al., 2025; Zhang et al., 2024c). Existing scientific LLMs are developed through the following paradigms: general scientific pretraining on large-scale scientific corpora (Taylor et al., 2022; Lewkowycz et al., 2022); domain-specific adaptation to specialized disciplines (Luo et al., 2022; Singhal et al., 2023; Zhang et al., 2024b); and scientific agentic workflows that enhance multi-step reasoning and problem solving (Zhang et al., 2024a; Ghafarollahi & Buehler, 2025). However, these scientific LLMs remain language-centric, where scientific data are interpreted as descriptions or symbolic sequences, which might be suboptimal for structured information (Jin et al., 2023; Sui et al., 2024; Sadeghi et al., 2024). Therefore, *Eywa* takes a complementary direction by using LLMs as reasoning interfaces to coordinate domain-specific models that operate on native scientific representations.

Alongside language models, domain-specific foundation models have rapidly advanced across scientific disciplines (Menon et al., 2026; Bommasani et al., 2021). Time series foundation models (Ansari et al., 2024; Woo et al., 2024) pretrained on large-scale temporal corpora can achieve competitive zero-shot forecasting (Kottapalli et al., 2025). For tabular data, TabPFN (Hollmann et al., 2023) leverages in-context learning to solve prediction tasks. The broader ecosystem further extends to more specialized scientific representations. In materials science, universal machine-learned interatomic potentials are trained on broad crystallographic databases (Merchant et al., 2023; Batatia et al., 2025; Deng et al., 2023). In weather and climate, GraphCast (Lam et al., 2022), Aurora (Bodnar et al., 2025), and GenCast (Price et al., 2025) show strong potential for medium-range forecasting. In life sciences, AlphaFold (Jumper et al., 2021) and the ESM family (Lin et al., 2023; Hayes et al., 2025) have transformed protein structure prediction and design. Despite their strong domain-specific capabilities, these models typically operate over domain-specific representations and do not expose native language interfaces, making their direct integration into language-centric agentic systems nontrivial. Bridging this interface gap is a central motivation of *Eywa*.

**Agentic Systems in Scientific Settings.** In scientific workflows, agentic systems have been applied to tasks such as hypothesis generation, literature synthesis, and experimental design, often under fixed agent topologies (Wei et al., 2025; Ghafarollahi & Buehler, 2025; Liu et al., 2025). More recently, end-to-end agentic systems have been explored for automating the AI research process itself (Lu et al., 2026). Another line of work augments agents with domain-specific tools by exposing complex simulators, solvers, or expert systems through external APIs when available (Kim et al., 2025b; Inoue et al., 2025). However, such systems often rely on human experts to predefine the task-solving procedure and implement well-specified tools before agent execution (Yuan et al., 2025). As a result, this solution is plausible for specific tasks with predefined workflows (Varambally et al., 2025), but is difficult to generalize across scientific domains or support dynamic model collaboration. *Eywa* addresses this gap by enabling on-the-fly scientific computation through active modality-native collaboration between LLMs and domain-specific foundation models.

## 7 Conclusion

Language-centric agentic systems are strong general problem solvers, but their reliance on textual interfaces limits their applicability to scientific tasks with structured, non-linguistic data. In this work, we propose *Eywa*, a heterogeneous framework that connects language reasoning and domain-specific foundation models through the FM–LLM "Tsaheylu" interface, and instantiate it as *EywaAgent*, *EywaMAS*, and *EywaOrchestra*. We further introduce *EywaBench* for multi-task, multi-domain evaluation in scientific settings. Experiments across physical, life, and social science tasks show that *Eywa* improves utility while reducing token usage and inference cost, demonstrating the value of modality-native collaboration in agentic systems.

## AI USE STATEMENT

In this work, we used generative AI tools to assist with writing. We have not used generative AI tools to generate research ideas or design the method. Note that large language models are also studied as components of the proposed agentic systems, which is part of the research itself rather than assistance in preparing this work. We have reviewed and verified all AI-assisted content, and we take full responsibility for the final content of this work.

## ETHICS STATEMENT

Our work is solely focused on the technical challenge of collaboration between language models and domain-specific foundation models, uses only publicly available datasets, and does not involve any elements that could pose ethical risks to the best of our knowledge.

## REPRODUCIBILITY STATEMENT

We provide the details needed to reproduce our results, including the experimental setup and hyperparameters in Section 5, and the construction and metrics of *EywaBench* in Appendix C. Our code is provided in the supplementary material.

## REFERENCES


Abdul Fatir Ansari, Lorenzo Stella, Ali Caner Türkmen, Xiyuan Zhang, Pedro Mercado, Huibin Shen, Oleksandr Shchur, Syama Sundar Rangapuram, Sebastian Pineda-Arango, Shubham Kapoor, Jasper Zschiegner, Danielle C. Maddix, Hao Wang, Michael W. Mahoney, Kari Torkkola, Andrew Gordon Wilson, Michael Bohlke-Schneider, and Bernie Wang. Chronos: Learning the language of time series. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=gerNCVqqtR.

Abdul Fatir Ansari, Oleksandr Shchur, Jaris Küken, Andreas Auer, Boran Han, Pedro Mercado, Syama Sundar Rangapuram, Huibin Shen, Lorenzo Stella, Xiyuan Zhang, Mononito Goswami, Shubham Kapoor, Danielle C. Maddix, Pablo Guerron, Tony Hu, Junming Yin, Nick Erickson, Prateek Mutalik Desai, Hao Wang, Huzefa Rangwala, George Karypis, Yuyang Wang, and Michael Bohlke-Schneider. Chronos-2: From univariate to universal forecasting. *CoRR*, abs/2510.15821, 2025. doi: 10.48550/ARXIV.2510.15821. URL https://doi.org/10.48550/arXiv.2510.15821.

Alessia Antelmi, Gennaro Cordasco, Mirko Polato, Vittorio Scarano, Carmine Spagnuolo, and Dingqi Yang. A survey on hypergraph representation learning. *ACM Computing Surveys*, 56(1):1–38, 2023.

Anthropic. Claude family models. https://platform.claude.com/docs/en/about-claude/models/overview, 2025.

Ilyes Batatia et al. A foundation model for atomistic materials chemistry. *The Journal of Chemical Physics*, 163(18):184110, 2025. doi: 10.1063/5.0297006.

Cristian Bodnar, Wessel P. Bruinsma, Ana Lucic, Megan Stanley, Anna Allen, Johannes Brandstetter, Patrick Garvan, Maik Riechert, Jonathan A. Weyn, Haiyu Dong, Jayesh K. Gupta, Kit Thambiratnam, Alexander T. Archibald, Chun-Chieh Wu, Elizabeth Heider, Max Welling, Richard E. Turner, and Paris Perdikaris. A foundation model for the earth system. *Nat.*, 641(8065):1180–1187, 2025. doi: 10.1038/S41586-025-09005-Y. URL https://doi.org/10.1038/s41586-025-09005-y.

Rishi Bommasani, Drew A. Hudson, Ehsan Adeli, Russ B. Altman, Simran Arora, Sydney von Arx, Michael S. Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, Erik Brynjolfsson, Shyamal Buch, Dallas Card, Rodrigo Castellon, Niladri S. Chatterji, Annie S. Chen, Kathleen Creel, Jared Quincy Davis, Dorottya Demszky, Chris Donahue, Moussa Doumbouya, Esin Durmus, Stefano Ermon, John Etchemendy, Kawin Ethayarajh, Li Fei-Fei, Chelsea Finn, Trevor Gale, Lauren E. Gillespie, Karan Goel, Noah D. Goodman, Shelby Grossman, Neel Guha, Tatsunori Hashimoto, Peter Henderson, John Hewitt, Daniel E. Ho, Jenny Hong, Kyle Hsu, Jing Huang, Thomas Icard, Saahil Jain, Dan Jurafsky, Pratyusha Kalluri, Siddharth Karamcheti, Geoff Keeling, Fereshte Khani, Omar Khattab, Pang Wei Koh, Mark S. Krass, Ranjay Krishna, Rohith Kuditipudi, and et al. On the opportunities and risks of foundation models. *CoRR*, abs/2108.07258, 2021. URL https://arxiv.org/abs/2108.07258.

Vadim Borisov, Tobias Leemann, Kathrin Seßler, Johannes Haug, Martin Pawelczyk, and Gjergji Kasneci. Deep neural networks and tabular data: A survey. *IEEE transactions on neural networks and learning systems*, 35(6):7499–7519, 2022.

George E. P. Box, Gwilym M. Jenkins, Gregory C. Reinsel, and Greta M. Ljung. *Time Series Analysis: Forecasting and Control*. John Wiley & Sons, 5th edition, 2015.

Yupeng Chang, Xu Wang, Jindong Wang, Yuan Wu, Kaijie Zhu, Hao Chen, Linyi Yang, Xiaoyuan Yi, Cunxiang Wang, Yidong Wang, Wei Ye, Yue Zhang, Yi Chang, Philip S. Yu, Qiang Yang, and Xing Xie. A survey on evaluation of large language models. *CoRR*, abs/2307.03109, 2023. doi: 10.48550/ARXIV.2307.03109. URL https://doi.org/10.48550/arXiv.2307.03109.

Qiguang Chen, Ming-Hsuan Yang, Libo Qin, Jinhao Liu, Zheng Yan, Jiannan Guan, Dengyun Peng, Yiyan Ji, Hanjing Li, Mengkang Hu, Yimeng Zhang, Yihao Liang, Yu Zhou, Jiaqi Wang, Zhi Chen, and Wanxiang Che. Ai4research: A survey of artificial intelligence for scientific research. *CoRR*, abs/2507.01903, 2025. doi: 10.48550/ARXIV.2507.01903. URL https://doi.org/10.48550/arXiv.2507.01903.

Shengchao Chen, Guodong Long, Jing Jiang, Dikai Liu, and Chengqi Zhang. Foundation models for weather and climate data understanding: A comprehensive survey. *CoRR*, abs/2312.03014, 2023a. doi: 10.48550/ARXIV.2312.03014. URL https://doi.org/10.48550/arXiv.2312.03014.

Zhikai Chen, Haitao Mao, Hang Li, Wei Jin, Hongzhi Wen, Xiaochi Wei, Shuaiqiang Wang, Dawei Yin, Wenqi Fan, Hui Liu, and Jiliang Tang. Exploring the potential of large language models (llms)in learning on graphs. *SIGKDD Explor.*, 25(2):42–61, 2023b. doi: 10.1145/3655103.3655110. URL https://doi.org/10.1145/3655103.3655110.

Seyone Chithrananda, Gabriel Grand, and Bharath Ramsundar. Chemberta: large-scale self-supervised pretraining for molecular property prediction. *arXiv preprint arXiv:2010.09885*, 2020.

Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261*, 2025.

Song Dai, Yibo Yan, Jiamin Su, Dongfang Zihao, Yubo Gao, Yonghua Hei, Jungang Li, Junyan Zhang, Sicheng Tao, Zhuoran Gao, and Xuming Hu. Physicsarena: The first multimodal physics reasoning benchmark exploring variable, process, and solution dimensions. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 17290–17316. Association for Computational Linguistics, 2025. URL https://aclanthology.org/2025.findings-emnlp.937/.

Yufan Dang, Chen Qian, Xueheng Luo, Jingru Fan, Zihao Xie, Ruijie Shi, Weize Chen, Cheng Yang, Xiaoyin Che, Ye Tian, Xuantang Xiong, Lei Han, Zhiyuan Liu, and Maosong Sun. Multi-agent collaboration via evolving orchestration. *CoRR*, abs/2505.19591, 2025. doi: 10.48550/ARXIV.2505.19591. URL https://doi.org/10.48550/arXiv.2505.19591.

Abhimanyu Das, Weihao Kong, Rajat Sen, and Yichen Zhou. A decoder-only foundation model for time-series forecasting. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 10148–10167. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/das24c.html.

DeepSeek-AI. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *CoRR*, abs/2501.12948, 2025. doi: 10.48550/ARXIV.2501.12948. URL https://doi.org/10.48550/arXiv.2501.12948.

Bowen Deng, Peichen Zhong, KyuJung Jun, Janosh Riebesell, Kevin Han, Christopher J. Bartel, and Gerbrand Ceder. Chgnet as a pretrained universal neural network potential for charge-informed atomistic modelling. *Nat. Mac. Intell.*, 5(9):1031–1041, 2023. doi: 10.1038/S42256-023-00716-3. URL https://doi.org/10.1038/s42256-023-00716-3.

Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving factuality and reasoning in language models through multiagent debate. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, volume 235 of *Proceedings of Machine Learning Research*, pp. 11733–11763. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/du24e.html.

Vijay Ekambaram, Arindam Jati, Pankaj Dayama, Sumanta Mukherjee, Nam H Nguyen, Wesley M Gifford, Chandra Reddy, and Jayant Kalagnanam. Tiny time mixers (ttms): Fast pre-trained models for enhanced zero/few-shot forecasting of multivariate time series. *Advances in Neural Information Processing Systems*, 37:74147–74181, 2024.

Nick Erickson, Lennart Purucker, Andrej Tschalzev, David Holzmüller, Prateek Mutalik Desai, David Salinas, and Frank Hutter. Tabarena: A living benchmark for machine learning on tabular data.

*CoRR*, abs/2506.16791, 2025. doi: 10.48550/ARXIV.2506.16791. URL https://doi.org/10.48550/arXiv.2506.16791.

Tao Feng, Chuanyang Jin, Jingyu Liu, Kunlun Zhu, Haoqin Tu, Zirui Cheng, Guanyu Lin, and Jiaxuan You. How far are we from AGI: are llms all we need? *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=H2ZKqfNd0U.

Dongqi Fu. Investigating natural and artificial dynamics in graph data mining and machine learning. In *Proceedings of the 32nd ACM International Conference on Information and Knowledge Management*, pp. 5173–5176, 2023.

Zhangyang Gao, Hao Wang, Cheng Tan, Chenrui Xu, Mengdi Liu, Bozhen Hu, Linlin Chao, Xiaoming Zhang, and Stan Z Li. Pfmbench: Protein foundation model benchmark. *arXiv preprint arXiv:2506.14796*, 2025.

Gemma Team. Gemma 4 technical report. *arXiv preprint arXiv:2607.02770*, 2026.

Bogdan Georgiev, Javier Gómez-Serrano, Terence Tao, and Adam Zsolt Wagner. Mathematical exploration and discovery at scale. *CoRR*, abs/2511.02864, 2025. doi: 10.48550/ARXIV.2511.02864. URL https://doi.org/10.48550/arXiv.2511.02864.

Alireza Ghafarollahi and Markus J Buehler. Sciagents: automating scientific discovery through bioinspired multi-agent intelligent graph reasoning. *Advanced Materials*, 37(22):2413523, 2025.

Nimisha Ghosh, Daniele Santoni, Debaleena Nawn, Eleonora Ottaviani, and Giovanni Felici. A comprehensive review of transformer-based language models for protein sequence analysis and design. *CoRR*, abs/2507.13646, 2025. doi: 10.48550/ARXIV.2507.13646. URL https://doi.org/10.48550/arXiv.2507.13646.

Mononito Goswami, Konrad Szafer, Arjun Choudhry, Yifu Cai, Shuo Li, and Artur Dubrawski. Moment: A family of open time-series foundation models. *arXiv preprint arXiv:2402.03885*, 2024.

Taicheng Guo, Xiuying Chen, Yaqi Wang, Ruidi Chang, Shichao Pei, Nitesh V. Chawla, Olaf Wiest, and Xiangliang Zhang. Large language model based multi-agents: A survey of progress and challenges. In *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI 2024, Jeju, South Korea, August 3-9, 2024*, pp. 8048–8057. ijcai.org, 2024. URL https://www.ijcai.org/proceedings/2024/890.

Hendrik F Hamann, Blazhe Gjorgiev, Thomas Brunschwiler, Leonardo SA Martins, Alban Puech, Anna Varbella, Jonas Weiss, Juan Bernabe-Moreno, Alexandre Blondin Massé, Seong Lok Choi, et al. Foundation models for the electric power grid. *Joule*, 8(12):3245–3258, 2024.

Jiawei Han, Jian Pei, and Hanghang Tong. *Data mining: concepts and techniques*. Morgan kaufmann, 2022.

Yana Hasson, Pauline Luc, Liliane Momeni, Maks Ovsjanikov, Guillaume Le Moing, Alina Kuznetsova, Ira Ktena, Jennifer J. Sun, Skanda Koppula, Dilara Gokay, Joseph Heyward, Etienne Pot, and Andrew Zisserman. Scivid: Cross-domain evaluation of video models in scientific applications. *CoRR*, abs/2507.03578, 2025. doi: 10.48550/ARXIV.2507.03578. URL https://doi.org/10.48550/arXiv.2507.03578.

Thomas Hayes, Roshan Rao, Halil Akin, Nicholas J. Sofroniew, Deniz Oktay, Zeming Lin, Robert Verkuil, Vincent Q. Tran, Jonathan Deaton, Marius Wiggert, Rohil Badkundri, Irhum Shafkat, Jun Gong, Alexander Derry, Raul S. Molina, Neil Thomas, Yousuf A. Khan, Chetan Mishra, Carolyn Kim, Liam J. Bartie, Matthew Nemeth, Patrick D. Hsu, Tom Sercu, Salvatore Candido, and Alexander Rives. Simulating 500 million years of evolution with a language model. *Science*, 387(6736):850–858, 2025. doi: 10.1126/science.ads0018.

Henry Herzog, Favyen Bastani, Yawen Zhang, Gabriel Tseng, Joseph Redmon, Hadrien Sablon, Ryan Park, Jacob Morrison, Alexandra Buraczynski, Karen Farley, et al. Olmoearth: Stable latent image modeling for multimodal earth observation. *arXiv preprint arXiv:2511.13655*, 2025.

Noah Hollmann, Samuel Müller, Katharina Eggensperger, and Frank Hutter. Tabpfn: A transformer that solves small tabular classification problems in a second. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=cp5PvcI6w8_.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, Liyang Zhou, Chenyu Ran, Lingfeng Xiao, Chenglin Wu, and Jürgen Schmidhuber. Metagpt: Meta programming for A multi-agent collaborative framework. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. URL https://openreview.net/forum?id=VtmBAGCN7o.

Ming Hu, Chenglong Ma, Wei Li, Wanghan Xu, Jiamin Wu, Jucheng Hu, Tianbin Li, Guohang Zhuang, Jiaqi Liu, Yingzhou Lu, et al. A survey of scientific large language models: From data foundations to agent frontiers. *arXiv preprint arXiv:2508.21148*, 2025.

Mingyang Hu, Fajie Yuan, Kevin Yang, Fusong Ju, Jin Su, Hui Wang, Fei Yang, and Qiuyang Ding. Exploring evolution-aware &-free protein language models as protein function predictors. *Advances in Neural Information Processing Systems*, 35:38873–38884, 2022.

Yoshitaka Inoue, Tianci Song, Xinling Wang, Augustin Luna, and Tianfan Fu. Drugagent: Multi-agent large language model-based reasoning for drug-target interaction prediction. *ArXiv*, pp. arXiv–2408, 2025.

Jinhao Jiang, Kun Zhou, Zican Dong, Keming Ye, Xin Zhao, and Ji-Rong Wen. Structgpt: A general framework for large language model to reason over structured data. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pp. 9237–9251. Association for Computational Linguistics, 2023. doi: 10.18653/V1/2023.EMNLP-MAIN.574. URL https://doi.org/10.18653/v1/2023.emnlp-main.574.

Bowen Jin, Gang Liu, Chi Han, Meng Jiang, Heng Ji, and Jiawei Han. Large language models on graphs: A comprehensive survey. *IEEE Trans. Knowl. Data Eng.*, 36(12):8622–8642, 2024. doi: 10.1109/TKDE.2024.3469578. URL https://doi.org/10.1109/TKDE.2024.3469578.

Ming Jin, Shiyu Wang, Lintao Ma, Zhixuan Chu, James Y Zhang, Xiaoming Shi, Pin-Yu Chen, Yuxuan Liang, Yuan-Fang Li, Shirui Pan, et al. Time-llm: Time series forecasting by reprogramming large language models. *arXiv preprint arXiv:2310.01728*, 2023.

Baoyu Jing, Sanhorn Chen, Lecheng Zheng, Boyu Liu, Zihao Li, Jiaru Zou, Tianxin Wei, Zhining Liu, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Yuchen Yan, Dongqi Fu, Jingchao Ni, Jingrui He, and Hanghang Tong. TSAQA: time series analysis question and answering benchmark. *CoRR*, abs/2601.23204, 2026. doi: 10.48550/ARXIV.2601.23204. URL https://doi.org/10.48550/arXiv.2601.23204.

John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Dongki Kim, Wonbin Lee, and Sung Ju Hwang. Mol-llama: Towards general understanding of molecules in large molecular language model. *CoRR*, abs/2502.13449, 2025a. doi: 10.48550/ARXIV.2502.13449. URL https://doi.org/10.48550/arXiv.2502.13449.

Hyomin Kim, Yunhui Jang, and Sungsoo Ahn. MT-mol: Multi agent system with tool-based reasoning for molecular optimization. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2025*, pp. 11544–11573, Suzhou, China, November 2025b. Association for Computational Linguistics. ISBN 979-8-89176-335-7. doi: 10.18653/v1/2025.findings-emnlp.619. URL https://aclanthology.org/2025.findings-emnlp.619/.

Siva Rama Krishna Kottapalli, Karthik Hubli, Sandeep Chandrashekhara, Garima Jain, Sunayana Hubli, Gayathri Botla, and Ramesh Doddaiah. Foundation models for time series: A survey. *CoRR*, abs/2504.04011, 2025. doi: 10.48550/ARXIV.2504.04011. URL https://doi.org/10.48550/arXiv.2504.04011.

Rémi Lam, Alvaro Sanchez-Gonzalez, Matthew Willson, Peter Wirnsberger, Meire Fortunato, Alexander Pritzel, Suman V. Ravuri, Timo Ewalds, Ferran Alet, Zach Eaton-Rosen, Weihua Hu, Alexander Merose, Stephan Hoyer, George Holland, Jacklynn Stott, Oriol Vinyals, Shakir Mohamed, and Peter W. Battaglia. Graphcast: Learning skillful medium-range global weather forecasting. *CoRR*, abs/2212.12794, 2022. doi: 10.48550/ARXIV.2212.12794. URL https://doi.org/10.48550/arXiv.2212.12794.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, et al. Solving quantitative reasoning problems with language models. *Advances in neural information processing systems*, 35:3843–3857, 2022.

Feiran Li, Le Yuan, Hongzhong Lu, Gang Li, Yu Chen, Martin KM Engqvist, Eduard J Kerkhoven, and Jens Nielsen. Deep learning-based k cat prediction enables improved enzyme-constrained model reconstruction. *Nature catalysis*, 5(8):662–672, 2022.

Zihao Li, Xiao Lin, Zhining Liu, Jiaru Zou, Ziwei Wu, Lecheng Zheng, Dongqi Fu, Yada Zhu, Hendrik F. Hamann, Hanghang Tong, and Jingrui He. Language in the flow of time: Time-series-paired texts weaved into a unified temporal narrative. *CoRR*, abs/2502.08942, 2025a. doi: 10.48550/ARXIV.2502.08942. URL https://doi.org/10.48550/arXiv.2502.08942.

Zihao Li, Lecheng Zheng, Bowen Jin, Dongqi Fu, Baoyu Jing, Yikun Ban, Jingrui He, and Jiawei Han. Can graph neural networks learn language with extremely weak text supervision? In Wanxiang Che, Joyce Nabende, Ekaterina Shutova, and Mohammad Taher Pilehvar (eds.), *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2025, Vienna, Austria, July 27 - August 1, 2025*, pp. 11138–11165. Association for Computational Linguistics, 2025b. URL https://aclanthology.org/2025.acl-long.545/.

Bryan Lim and Stefan Zohren. Time-series forecasting with deep learning: a survey. *Philosophical transactions of the royal society a: mathematical, physical and engineering sciences*, 379(2194), 2021.

Zeming Lin, Halil Akin, Roshan Rao, Brian Hie, Zhongkai Zhu, Wenting Lu, Nikita Smetanin, Robert Verkuil, Ori Kabeli, Yaniv Shmueli, Allan dos Santos Costa, Maryam Fazel-Zarandi, Tom Sercu, Salvatore Candido, and Alexander Rives. Evolutionary-scale prediction of atomic-level protein structure with a language model. *Science*, 379(6637):1123–1130, 2023. doi: 10.1126/science.ade2574.

Geng Liu, Fei Zhu, Rong Feng, Changyi Ma, Shiqi Wang, and Gaofeng Meng. Intent mismatch causes llms to get lost in multi-turn conversation. *CoRR*, abs/2602.07338, 2026. doi: 10.48550/ARXIV.2602.07338. URL https://doi.org/10.48550/arXiv.2602.07338.

Haoyang Liu, Yijiang Li, and Haohan Wang. Genomas: A multi-agent framework for scientific discovery via code-driven gene expression analysis. *arXiv preprint arXiv:2507.21035*, 2025.

Tiqing Liu, Yuhmei Lin, Xin Wen, Robert N Jorissen, and Michael K Gilson. Bindingdb: a web-accessible database of experimentally determined protein–ligand binding affinities. *Nucleic acids research*, 35(suppl_1):D198–D201, 2007.

Llama Team. The llama 3 herd of models. *CoRR*, abs/2407.21783, 2024. doi: 10.48550/ARXIV.2407.21783. URL https://doi.org/10.48550/arXiv.2407.21783.

Chris Lu, Cong Lu, Robert Tjarko Lange, Yutaro Yamada, Shengran Hu, Jakob Foerster, David Ha, and Jeff Clune. Towards end-to-end automation of ai research. *Nature*, 651(8107):914–919, 2026.

Wei Lu, Rachel K. Luu, and Markus J. Buehler. Fine-tuning large language models for domain adaptation: Exploration of training strategies, scaling, model merging and synergistic capabilities. *CoRR*, abs/2409.03444, 2024. doi: 10.48550/ARXIV.2409.03444. URL https://doi.org/10.48550/arXiv.2409.03444.

Daqin Luo, Chengjian Feng, Yuxuan Nong, and Yiqing Shen. Autom$^3$l: An automated multimodal machine learning framework with large language models. In Jianfei Cai, Mohan S. Kankanhalli, Balakrishnan Prabhakaran, Susanne Boll, Ramanathan Subramanian, Liang Zheng, Vivek K.

Singh, Pablo César, Lexing Xie, and Dong Xu (eds.), *Proceedings of the 32nd ACM International Conference on Multimedia, MM 2024, Melbourne, VIC, Australia, 28 October 2024 - 1 November 2024*, pp. 8586–8594. ACM, 2024. doi: 10.1145/3664647.3680665. URL https://doi.org/10.1145/3664647.3680665.

Junyu Luo, Weizhi Zhang, Ye Yuan, Yusheng Zhao, Junwei Yang, Yiyang Gu, Bohan Wu, Binqi Chen, Ziyue Qiao, Qingqing Long, Rongcheng Tu, Xiao Luo, Wei Ju, Zhiping Xiao, Yifan Wang, Meng Xiao, Chenwu Liu, Jingyang Yuan, Shichang Zhang, Yiqiao Jin, Fan Zhang, Xian Wu, Hanqing Zhao, Dacheng Tao, Philip S. Yu, and Ming Zhang. Large language model agent: A survey on methodology, applications and challenges. *CoRR*, abs/2503.21460, 2025. doi: 10.48550/ARXIV.2503.21460. URL https://doi.org/10.48550/arXiv.2503.21460.

Renqian Luo, Liai Sun, Yingce Xia, Tao Qin, Sheng Zhang, Hoifung Poon, and Tie-Yan Liu. Biogpt: generative pre-trained transformer for biomedical text generation and mining. *Briefings in bioinformatics*, 23(6):bbac409, 2022.

M-A-P Team. Supergpqa: Scaling LLM evaluation across 285 graduate disciplines. *CoRR*, abs/2502.14739, 2025. doi: 10.48550/ARXIV.2502.14739. URL https://doi.org/10.48550/arXiv.2502.14739.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative refinement with self-feedback. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/91edff07232fb1b55a505a9e9f6c0ff3-Abstract-Conference.html.

Sidharth S. Menon, Trishit Mondal, Shuvayan Brahmachary, Aniruddha Panda, Subodh M. Joshi, Kaushic Kalyanaraman, and Ameya D. Jagtap. On scientific foundation models: Rigorous definitions, key applications, and a comprehensive survey. *Neural Networks*, 198:108567, 2026. doi: 10.1016/J.NEUNET.2026.108567. URL https://doi.org/10.1016/j.neunet.2026.108567.

Amil Merchant, Simon Batzner, Samuel S Schoenholz, Muratahan Aykol, Gowoon Cheon, and Ekin Dogus Cubuk. Scaling deep learning for materials discovery. *Nature*, 624(7990):80–85, 2023.

Mike A. Merrill, Alexander Glenn Shaw, Nicholas Carlini, Boxuan Li, Harsh Raj, Ivan Bercovich, Lin Shi, Jeong Yeon Shin, Thomas Walshe, Estefany Kelly Buchanan, Junhong Shen, Guanghao Ye, Haowei Lin, Jason Poulos, Maoyu Wang, Marianna Nezhurina, Jenia Jitsev, Di Lu, Orfeas Menis-Mastromichalakis, Zhiwei Xu, Zizhao Chen, Yue Liu, Robert Zhang, Leon Liangyu Chen, Anurag Kashyap, Jan-Lucas Uslu, Jeffrey Li, Jianbo Wu, Minghao Yan, Song Bian, Vedang Sharma, Ke Sun, Steven Dillmann, Akshay Anand, Andrew Lanpouthakoun, Bardia Koopah, Changran Hu, Etash Kumar Guha, Gabriel H. S. Dreiman, Jiacheng Zhu, Karl Krauth, Li Zhong, Niklas Muennighoff, Robert Amanfu, Shangyin Tan, Shreyas Pimpalgaonkar, Tushar Aggarwal, Xiangning Lin, Xin Lan, Xuandong Zhao, Yiqing Liang, Yuanli Wang, Zilong Wang, Changzhi Zhou, David Heineman, Hange Liu, Harsh Trivedi, John Yang, Junhong Lin, Manish Shetty, Michael Yang, Nabil Omi, Negin Raoof, Shanda Li, Terry Yue Zhuo, Wuwei Lin, Yiwei Dai, Yuxin Wang, Wenhao Chai, Shang Zhou, Dariush Wahdany, Ziyu She, Jiaming Hu, Zhikang Dong, Yuxuan Zhu, Sasha Cui, Ahson Saiyed, Arinbjörn Kolbeinsson, Jesse Hu, Christopher Michael Rytting, Ryan Marten, Yixin Wang, Alex Dimakis, Andy Konwinski, and Ludwig Schmidt. Terminal-bench: Benchmarking agents on hard, realistic tasks in command line interfaces. *CoRR*, abs/2601.11868, 2026. doi: 10.48550/ARXIV.2601.11868. URL https://doi.org/10.48550/arXiv.2601.11868.

Xuying Ning, Dongqi Fu, Tianxin Wei, Mengting Ai, Jiaru Zou, Ting-Wei Li, Hanghang Tong, Yada Zhu, Hendrik Hamann, and Jingrui He. Mc-search: Evaluating and enhancing multimodal agentic search with structured long reasoning chains. In *The Fourteenth International Conference on Learning Representations*, 2026.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL https://doi.org/10.48550/arXiv.2303.08774.

OpenAI. Openai GPT-5 system card. *CoRR*, abs/2601.03267, 2026. doi: 10.48550/ARXIV.2601.03267. URL https://doi.org/10.48550/arXiv.2601.03267.

Boris N. Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*, 2020.

Bryan Perozzi, Bahare Fatemi, Dustin Zelle, Anton Tsitsulin, Seyed Mehran Kazemi, Rami Al-Rfou, and Jonathan Halcrow. Let your graph do the talking: Encoding structured data for llms. *CoRR*, abs/2402.05862, 2024. doi: 10.48550/ARXIV.2402.05862. URL https://doi.org/10.48550/arXiv.2402.05862.

Ilan Price, Alvaro Sanchez-Gonzalez, Ferran Alet, Tom R. Andersson, Andrew El-Kadi, Dominic Masters, Timo Ewalds, Jacklynn Stott, Shakir Mohamed, Peter W. Battaglia, Rémi R. Lam, and Matthew Willson. Probabilistic weather forecasting with machine learning. *Nat.*, 637(8044): 84–90, 2025. doi: 10.1038/S41586-024-08252-9. URL https://doi.org/10.1038/s41586-024-08252-9.

Shi Qiu, Shaoyang Guo, Zhuo-Yang Song, Yunbo Sun, Zeyu Cai, Jiashen Wei, Tianyu Luo, Yixuan Yin, Haoxu Zhang, Yi Hu, Chenyang Wang, Chencheng Tang, Haoling Chang, Qi Liu, Ziheng Zhou, Tianyu Zhang, Jingtian Zhang, Zhangyi Liu, Minghao Li, Yuku Zhang, Boxuan Jing, Xianqi Yin, Yutong Ren, Zizhuo Fu, Weike Wang, Xudong Tian, Anqi Lv, Laifu Man, Jianxiang Li, Feiyu Tao, Qihua Sun, Zhou Liang, Yushu Mu, Zhongxuan Li, Jing-Jun Zhang, Shutao Zhang, Xiaotian Li, Xingqi Xia, Jiawei Lin, Zheyu Shen, Jiahang Chen, Qiuhao Xiong, Binran Wang, Fengyuan Wang, Ziyang Ni, Bohan Zhang, Fan Cui, Changkun Shao, Qing-Hong Cao, Ming-xing Luo, Muhan Zhang, and Hua Xing Zhu. Phybench: Holistic evaluation of physical perception and reasoning in large language models. *CoRR*, abs/2504.16074, 2025a. doi: 10.48550/ARXIV.2504.16074. URL https://doi.org/10.48550/arXiv.2504.16074.

Zihan Qiu, Zekun Wang, Bo Zheng, Zeyu Huang, Kaiyue Wen, Songlin Yang, Rui Men, Le Yu, Fei Huang, Suozhi Huang, Dayiheng Liu, Jingren Zhou, and Junyang Lin. Gated attention for large language models: Non-linearity, sparsity, and attention-sink-free. *CoRR*, abs/2505.06708, 2025b. doi: 10.48550/ARXIV.2505.06708. URL https://doi.org/10.48550/arXiv.2505.06708.

Qwen Team. Qwen2.5 technical report. *arXiv preprint arXiv:2412.15115*, 2024.

Qwen Team. Qwen3-omni technical report. *CoRR*, abs/2509.17765, 2025. doi: 10.48550/ARXIV.2509.17765. URL https://doi.org/10.48550/arXiv.2509.17765.

Martiño Ríos-García, Nawaf Alampara, Chandan Gupta, Indrajeet Mandal, Sajid Mannan, Ali Asghar Aghajani, NM Krishnan, and Kevin Maik Jablonka. Ai scientists produce results without reasoning scientifically. *arXiv preprint arXiv:2604.18805*, 2026.

Shaghayegh Sadeghi, Alan Bui, Ali Forooghi, Jianguo Lu, and Alioune Ngom. Can large language models understand molecules? *BMC bioinformatics*, 25(1):225, 2024.

ByteDance Seed. Seed1.5-vl technical report. *CoRR*, abs/2505.07062, 2025. doi: 10.48550/ARXIV.2505.07062. URL https://doi.org/10.48550/arXiv.2505.07062.

Oleksandr Shchur, Abdul Fatir Ansari, Caner Turkmen, Lorenzo Stella, Nick Erickson, Pablo Guerron, Michael Bohlke-Schneider, and Yuyang Wang. fev-bench: A realistic benchmark for time series forecasting. *CoRR*, abs/2509.26468, 2025. doi: 10.48550/ARXIV.2509.26468. URL https://doi.org/10.48550/arXiv.2509.26468.

Yongliang Shen, Kaitao Song, Xu Tan, Dongsheng Li, Weiming Lu, and Yueting Zhuang. Hugginggpt: Solving AI tasks with chatgpt and its friends in hugging face. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 -*

*16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/77c33e6a367922d003ff102ffb92b658-Abstract-Conference.html.

Feifei Shi, Xueyan Yin, Kang Wang, Wanyu Tu, Qifu Sun, and Huansheng Ning. Large language models for time series analysis: Techniques, applications, and challenges. *CoRR*, abs/2506.11040, 2025. doi: 10.48550/ARXIV.2506.11040. URL https://doi.org/10.48550/arXiv.2506.11040.

Parshin Shojaee, Ngoc-Hieu Nguyen, Kazem Meidani, Amir Barati Farimani, Khoa D. Doan, and Chandan K. Reddy. Llm-srbench: A new benchmark for scientific equation discovery with large language models. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/shojaee25a.html.

Karan Singhal, Shekoofeh Azizi, Tao Tu, S Sara Mahdavi, Jason Wei, Hyung Won Chung, Nathan Scales, Ajay Tanwani, Heather Cole-Lewis, Stephen Pfohl, et al. Large language models encode clinical knowledge. *Nature*, 620(7972):172–180, 2023.

Zhangde Song, Jieyu Lu, Yuanqi Du, Botao Yu, Thomas M Pruyn, Yue Huang, Kehan Guo, Xiuzhe Luo, Yuanhao Qu, Yi Qu, et al. Evaluating large language models in scientific discovery. *arXiv preprint arXiv:2512.15567*, 2025.

Chen Su, Yuanhe Tian, Yan Song, and Yongdong Zhang. Text reinforcement for multimodal time series forecasting. *CoRR*, abs/2509.00687, 2025a. doi: 10.48550/ARXIV.2509.00687. URL https://doi.org/10.48550/arXiv.2509.00687.

Guangxin Su, Hanchen Wang, Jianwei Wang, Wenjie Zhang, Ying Zhang, and Jian Pei. Large language models meet text-attributed graphs: A survey of integration frameworks and applications. *CoRR*, abs/2510.21131, 2025b. doi: 10.48550/ARXIV.2510.21131. URL https://doi.org/10.48550/arXiv.2510.21131.

Jinwei Su, Yinghui Xia, Qizhen Lan, Xinyuan Song, Chen Chen, JingSong Yang, Lewei He, and Tianyu Shi. Difficulty-aware agent orchestration in llm-powered workflows. *CoRR*, abs/2509.11079, 2025c. doi: 10.48550/ARXIV.2509.11079. URL https://doi.org/10.48550/arXiv.2509.11079.

Yuan Sui, Mengyu Zhou, Mingjie Zhou, Shi Han, and Dongmei Zhang. Table meets llm: Can large language models understand structured table data? a benchmark and empirical study. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*, pp. 645–654, 2024.

Synthefy, Po-han Li, Aditya Narayanan, Sai Shankar Narasimhan, Raghav Mallampalli, Aahan Agrawal, Bekzat Ajan, Raimi Shah, and Shubhankar Agarwal. Synthefy nori: Tabular foundation model for regression, 2026. URL https://doi.org/10.5281/zenodo.20710462.

Xiaoyu Tan, Haoyu Wang, Xihe Qiu, Yuan Cheng, Yinghui Xu, Wei Chu, and Yuan Qi. Struct-x: Enhancing large language models reasoning with structured data. *CoRR*, abs/2407.12522, 2024. doi: 10.48550/ARXIV.2407.12522. URL https://doi.org/10.48550/arXiv.2407.12522.

Jiabin Tang, Yuhao Yang, Wei Wei, Lei Shi, Lixin Su, Suqi Cheng, Dawei Yin, and Chao Huang. Graphgpt: Graph instruction tuning for large language models. In Grace Hui Yang, Hongning Wang, Sam Han, Claudia Hauff, Guido Zuccon, and Yi Zhang (eds.), *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2024, Washington DC, USA, July 14-18, 2024*, pp. 491–500. ACM, 2024. doi: 10.1145/3626772.3657775. URL https://doi.org/10.1145/3626772.3657775.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. Galactica: A large language model for science. *arXiv preprint arXiv:2211.09085*, 2022.

Thinking Machines Lab. Introducing Inkling. https://thinkingmachines.ai/news/introducing-inkling/, July 2026.

Aaron Xuxiang Tian, Ruofan Zhang, Jiayao Tang, Young Min Cho, Xueqian Li, Qiang Yi, Ji Wang, Zhunping Zhang, Danrui Qi, Zekun Li, Xingyu Xiang, Sharath Chandra Guntuku, Lyle H. Ungar, Tianyu Shi, and Chi Wang. Beyond the strongest LLM: multi-turn multi-agent orchestration vs. single llms on benchmarks. *CoRR*, abs/2509.23537, 2025. doi: 10.48550/ARXIV.2509.23537. URL https://doi.org/10.48550/arXiv.2509.23537.

Shengbang Tong, David Fan, John Nguyen, Ellis Brown, Gaoyue Zhou, Shengyi Qian, Boyang Zheng, Théophane Vallaeys, Junlin Han, Rob Fergus, Naila Murray, Marjan Ghazvininejad, Mike Lewis, Nicolas Ballas, Amir Bar, Michael Rabbat, Jakob Verbeek, Luke Zettlemoyer, Koustuv Sinha, Yann LeCun, and Saining Xie. Beyond language modeling: An exploration of multimodal pretraining. *CoRR*, abs/2603.03276, 2026. doi: 10.48550/ARXIV.2603.03276. URL https://doi.org/10.48550/arXiv.2603.03276.

Alexander Tornede, Difan Deng, Theresa Eimer, Joseph Giovanelli, Aditya Mohan, Tim Ruhkopf, Sarah Segel, Daphne Theodorakopoulos, Tanja Tornede, Henning Wachsmuth, and Marius Lindauer. Automl in the age of large language models: Current challenges, future opportunities and risks. *Trans. Mach. Learn. Res.*, 2024, 2024. URL https://openreview.net/forum?id=cAthubStyG.

Patara Trirat, Wonyong Jeong, and Sung Ju Hwang. Automl-agent: A multi-agent LLM framework for full-pipeline automl. In Aarti Singh, Maryam Fazel, Daniel Hsu, Simon Lacoste-Julien, Felix Berkenkamp, Tegan Maharaj, Kiri Wagstaff, and Jerry Zhu (eds.), *Forty-second International Conference on Machine Learning, ICML 2025, Vancouver, BC, Canada, July 13-19, 2025*, Proceedings of Machine Learning Research. PMLR / OpenReview.net, 2025. URL https://proceedings.mlr.press/v267/trirat25a.html.

Sumanth Varambally, Marshall Fisher, Jas Thakker, Yiwei Chen, Zhirui Xia, Yasaman Jafari, Ruijia Niu, Manas Jain, Veeramakali Vignesh Manivannan, Zachary Novack, Luyu Han, Srikar Eranky, Salva Rühling Cachay, Taylor Berg-Kirkpatrick, Duncan Watson-Parris, Yi-An Ma, and Rose Yu. Zephyrus: An agentic framework for weather science. *CoRR*, abs/2510.04017, 2025. doi: 10.48550/ARXIV.2510.04017. URL https://doi.org/10.48550/arXiv.2510.04017.

Fengxiang Wang, Hongzhen Wang, Zonghao Guo, Di Wang, Yulin Wang, Mingshuo Chen, Qiang Ma, Long Lan, Wenjing Yang, Jing Zhang, Zhiyuan Liu, and Maosong Sun. Xlrs-bench: Could your multimodal llms understand extremely large ultra-high-resolution remote sensing imagery? In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2025, Nashville, TN, USA, June 11-15, 2025*, pp. 14325–14336. Computer Vision Foundation / IEEE, 2025a. doi: 10.1109/CVPR52734.2025.01336. URL https://openaccess.thecvf.com/content/CVPR2025/html/Wang_XLRS-Bench_Could_Your_Multimodal_LLMs_Understand_Extremely_Large_Ultra-High-Resolution_Remote_CVPR_2025_paper.html.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. Mixture-of-agents enhances large language model capabilities. In *The Thirteenth International Conference on Learning Representations, ICLR 2025, Singapore, April 24-28, 2025*. OpenReview.net, 2025b. URL https://openreview.net/forum?id=h0ZfDIrj7T.

Lei Wang, Chen Ma, Xueyang Feng, Zeyu Zhang, Hao Yang, Jingsen Zhang, Zhiyuan Chen, Jiakai Tang, Xu Chen, Yankai Lin, Wayne Xin Zhao, Zhewei Wei, and Jirong Wen. A survey on large language model based autonomous agents. *Frontiers Comput. Sci.*, 18(6):186345, 2024a. doi: 10.1007/S11704-024-40231-1. URL https://doi.org/10.1007/s11704-024-40231-1.

Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyan Jiang, et al. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark. *Advances in Neural Information Processing Systems*, 37: 95266–95290, 2024b.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language

models. In Sanmi Koyejo, S. Mohamed, A. Agarwal, Danielle Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*, 2022. URL http://papers.nips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Jiaqi Wei, Yuejin Yang, Xiang Zhang, Yuhan Chen, Xiang Zhuang, Zhangyang Gao, Dongzhan Zhou, Guangshuai Wang, Zhiqiang Gao, Juntai Cao, et al. From ai for science to agentic science: A survey on autonomous scientific discovery. *arXiv preprint arXiv:2508.14111*, 2025.

Tianxin Wei, Ting-Wei Li, Zhining Liu, Xuying Ning, Ze Yang, Jiaru Zou, Zhichen Zeng, Ruizhong Qiu, Xiao Lin, Dongqi Fu, Zihao Li, Mengting Ai, Duo Zhou, Wenxuan Bao, Yunzhe Li, Gaotang Li, Cheng Qian, Yu Wang, Xiangru Tang, Yin Xiao, Liri Fang, Hui Liu, Xianfeng Tang, Yuji Zhang, Chi Wang, Jiaxuan You, Heng Ji, Hanghang Tong, and Jingrui He. Agentic reasoning for large language models. *CoRR*, abs/2601.12538, 2026. doi: 10.48550/ARXIV.2601.12538. URL https://doi.org/10.48550/arXiv.2601.12538.

Gerald Woo, Chenghao Liu, Akshat Kumar, Caiming Xiong, Silvio Savarese, and Doyen Sahoo. Unified training of universal time series forecasting transformers. In Ruslan Salakhutdinov, Zico Kolter, Katherine A. Heller, Adrian Weller, Nuria Oliver, Jonathan Scarlett, and Felix Berkenkamp (eds.), *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*, Proceedings of Machine Learning Research, pp. 53140–53164. PMLR / OpenReview.net, 2024. URL https://proceedings.mlr.press/v235/woo24a.html.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Shaokun Zhang, Erkang Zhu, Beibin Li, Li Jiang, Xiaoyun Zhang, and Chi Wang. Autogen: Enabling next-gen LLM applications via multi-agent conversation framework. *CoRR*, abs/2308.08155, 2023. doi: 10.48550/ARXIV.2308.08155. URL https://doi.org/10.48550/arXiv.2308.08155.

Zhenqin Wu, Bharath Ramsundar, Evan N. Feinberg, Joseph Gomes, Caleb Geniesse, Aneesh S. Pappu, Karl Leswing, and Vijay S. Pande. Moleculenet: A benchmark for molecular machine learning. *CoRR*, abs/1703.00564, 2017. URL http://arxiv.org/abs/1703.00564.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *IEEE transactions on neural networks and learning systems*, 32(1):4–24, 2020.

Bingyu Yan, Xiaoming Zhang, Litian Zhang, Lian Zhang, Ziyi Zhou, Dezhuang Miao, and Chaozhuo Li. Beyond self-talk: A communication-centric survey of llm-based multi-agent systems. *CoRR*, abs/2502.14321, 2025. doi: 10.48550/ARXIV.2502.14321. URL https://doi.org/10.48550/arXiv.2502.14321.

Hao Yan, Chaozhuo Li, Ruosong Long, Chao Yan, Jianan Zhao, Wenwen Zhuang, Jun Yin, Peiyan Zhang, Weihao Han, Hao Sun, Weiwei Deng, Qi Zhang, Lichao Sun, Xing Xie, and Senzhang Wang. A comprehensive study on text-attributed graphs: Benchmarking and rethinking. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine (eds.), *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/37d00f567a18b478065f1a91b95622a0-Abstract-Datasets_and_Benchmarks.html.

Yuhao Yang, Jiabin Tang, Lianghao Xia, Xingchen Zou, Yuxuan Liang, and Chao Huang. Graphagent: Agentic graph language assistant. In Christos Christodoulopoulos, Tanmoy Chakraborty, Carolyn Rose, and Violet Peng (eds.), *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing, EMNLP 2025, Suzhou, China, November 4-9, 2025*, pp. 26349–26368. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.EMNLP-MAIN.1339. URL https://doi.org/10.18653/v1/2025.emnlp-main.1339.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=WE_vluYUL-X.

Rui Ye, Xiangrui Liu, Qimin Wu, Xianghe Pang, Zhenfei Yin, Lei Bai, and Siheng Chen. X-MAS: towards building multi-agent systems with heterogeneous llms. *CoRR*, abs/2505.16997, 2025. doi: 10.48550/ARXIV.2505.16997. URL https://doi.org/10.48550/arXiv.2505.16997.

Ming Yin, Yuanhao Qu, Dyllan Liu, Ling Yang, Le Cong, and Mengdi Wang. Genome-bench: A scientific reasoning benchmark from real-world expert discussions. *CoRR*, abs/2505.19501, 2025. doi: 10.48550/ARXIV.2505.19501. URL https://doi.org/10.48550/arXiv.2505.19501.

Siyu Yuan, Kaitao Song, Jiangjie Chen, Xu Tan, Yongliang Shen, Kan Ren, Dongsheng Li, and Deqing Yang. EASYTOOL: enhancing llm-based agents with concise tool instruction. In Luis Chiruzzo, Alan Ritter, and Lu Wang (eds.), *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2025 - Volume 1: Long Papers, Albuquerque, New Mexico, USA, April 29 - May 4, 2025*, pp. 951–972. Association for Computational Linguistics, 2025. doi: 10.18653/V1/2025.NAACL-LONG.44. URL https://doi.org/10.18653/v1/2025.naacl-long.44.

Dan Zhang, Ziniu Hu, Sining Zhoubian, Zhengxiao Du, Kaiyu Yang, Zihan Wang, Yisong Yue, Yuxiao Dong, and Jie Tang. Sciglm: Training scientific language models with self-reflective instruction annotation and tuning. *arXiv preprint arXiv:2401.07950*, 4, 2024a.

Di Zhang, Wei Liu, Qian Tan, Jingdan Chen, Hang Yan, Yuliang Yan, Jiatong Li, Weiran Huang, Xiangyu Yue, Wanli Ouyang, et al. Chemllm: A chemical large language model. *arXiv preprint arXiv:2402.06852*, 2024b.

Wentao Zhang, Liang Zeng, Yuzhen Xiao, Yongcong Li, Ce Cui, Yilei Zhao, Rui Hu, Yang Liu, Yahui Zhou, and Bo An. Agentorchestra: Orchestrating multi-agent intelligence with the tool-environment-agent (tea) protocol. *arXiv preprint arXiv:2506.12508*, 2025a.

Xiyuan Zhang, Danielle Maddix Robinson, Junming Yin, Nick Erickson, Abdul Fatir Ansari, Boran Han, Shuai Zhang, Leman Akoglu, Christos Faloutsos, Michael Mahoney, et al. Mitra: Mixed synthetic priors for enhancing tabular foundation models. *Advances in neural information processing systems*, 38:15795–15840, 2026.

Yu Zhang, Xiusi Chen, Bowen Jin, Sheng Wang, Shuiwang Ji, Wei Wang, and Jiawei Han. A comprehensive survey of scientific large language models and their applications in scientific discovery. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 8783–8817, 2024c.

Yu Zhang, Zongyu Lin, Xingcheng Yao, Jiaxi Hu, Fanqing Meng, Chengyin Liu, Xin Men, Songlin Yang, Zhiyuan Li, Wentao Li, Enzhe Lu, Weizhou Liu, Yanru Chen, Weixin Xu, Longhui Yu, Yejie Wang, Yu Fan, Longguang Zhong, Enming Yuan, Dehao Zhang, Yizhi Zhang, T. Y. Liu, Haiming Wang, Shengjun Fang, Weiran He, Shaowei Liu, Yiwei Li, Jianlin Su, Jiezhong Qiu, Bo Pang, Junjie Yan, Zhejun Jiang, Weixiao Huang, Bohong Yin, Jiacheng You, Chu Wei, Zhengtao Wang, Chao Hong, Yutian Chen, Guanduo Chen, Yucheng Wang, Huabin Zheng, Feng Wang, Yibo Liu, Mengnan Dong, Zheng Zhang, Siyuan Pan, Wenhao Wu, Yuhao Wu, Longyu Guan, Jiawen Tao, Guohong Fu, Xinran Xu, Yuzhi Wang, Guokun Lai, Yuxin Wu, Xinyu Zhou, Zhilin Yang, and Yulun Du. Kimi linear: An expressive, efficient attention architecture. *CoRR*, abs/2510.26692, 2025b. doi: 10.48550/ARXIV.2510.26692. URL https://doi.org/10.48550/arXiv.2510.26692.

Haokun Zhao, Xiang Zhang, Jiaqi Wei, Yiwei Xu, Yuting He, Siqi Sun, and Chenyu You. Timeseriesscientist: A general-purpose AI agent for time series analysis. *CoRR*, abs/2510.01538, 2025. doi: 10.48550/ARXIV.2510.01538. URL https://doi.org/10.48550/arXiv.2510.01538.

Xiao Xiang Zhu, Zhitong Xiong, Yi Wang, Adam J. Stewart, Konrad Heidler, Yuanyuan Wang, Zhenghang Yuan, Thomas Dujardin, Qingsong Xu, and Yilei Shi. On the foundations of earth and climate foundation models. *CoRR*, abs/2405.04285, 2024. doi: 10.48550/ARXIV.2405.04285. URL https://doi.org/10.48550/arXiv.2405.04285.

Yihan Zhu, Gang Liu, Eric Inae, and Meng Jiang. Moltextnet: A two-million molecule-text dataset for multimodal molecular learning. *CoRR*, abs/2506.00009, 2025. doi: 10.48550/ARXIV.2506.00009. URL https://doi.org/10.48550/arXiv.2506.00009.

# Appendix

**Roadmap.** In this appendix, we provide a detailed overview of our methodology and experimental setup. Appendix A presents the full theoretical analysis of *Eywa*, covering the language interface bottleneck, the expressivity and solvability guarantees of *EywaAgent*, the generalization to *EywaMAS*, the theory of *EywaOrchestra* as adaptive orchestration, and a token-complexity analysis. Appendix B provides further discussions on additional related work, potential extensions, and the design choices of *Eywa*. Appendix C describes *EywaBench* in detail, including domain coverage, modality composition, source datasets, construction pipeline, evaluation metrics, and a statistical analysis of the resulting benchmark. Finally, Appendix D reports additional experimental results, including more detailed LLM-backbone ablations, qualitative case studies, and per-domain utility-token trade-off analyses across all nine scientific sub-domains. Appendix E contains LLM prompt examples. The table of contents is provided below for quick navigation.

## Table of Contents


**A Theoretical Analysis** **24**

A.1 Notation, Problem Setup, and Assumptions . . . 24
A.2 Information-Theoretic Analysis of the Language Interface Bottleneck . . . 26
A.3 Expressivity and Solvability of EywaAgent . . . 28
A.4 Multi-Agent Propagation: EywaMAS . . . 29
A.5 Adaptive Orchestration: EywaOrchestra . . . 29
A.6 Efficiency and Token Complexity . . . 30
A.7 Summary of Theoretical Results . . . 31

**B Further Discussions** **33**

B.1 More Related Works . . . 33
B.2 Future Directions . . . 34
B.3 Design Choices . . . 35

**C EywaBench Details** **36**

C.1 Source Datasets . . . 36
C.2 Data Schema . . . 37
C.3 Composition and Coverage Analysis . . . 38
C.4 Representativeness of EywaBench-mini . . . 40
C.5 Metrics . . . 40

**D Experiment Details** **41**

D.1 More Detailed Ablations . . . 41
D.2 Replacing the Time-Series Foundation Model . . . 45
D.3 Extending the Foundation Model Pool . . . 45
D.4 Deployment of Foundation Models . . . 45
D.5 Case Study . . . 46
D.6 Detailed Utility and Token Consumption . . . 46

**E Prompt Templates** **49**

# A THEORETICAL ANALYSIS

In this section, we conduct a detailed theoretical analysis of our *Eywa* framework. We first unify the notation used across the main paper and the appendix, and collect a complete list of assumptions (Section A.1). We then develop the theory along four complementary axes. Section A.2 establishes an information-theoretic foundation that characterizes the *language interface bottleneck*, providing a first-principles justification for the Domain Advantage assumption of the main paper. Building on this foundation, Section A.3 proves the expressivity and solvability guarantees of *EywaAgent*. Section A.4 generalizes the analysis to *EywaMAS*, formalizing how the single-agent advantage propagates through communication graphs. Section A.5 develops the theory of *EywaOrchestra* as adaptive orchestration over heterogeneous experts. Finally, Section A.6 provides a token-complexity analysis that corroborates our empirical efficiency gains. Table 5 at the end of this section provides a consolidated summary of all theoretical results and their dependencies.

## A.1 NOTATION, PROBLEM SETUP, AND ASSUMPTIONS

**Notations.** We collect, in Table 4, the notation used throughout the main text and this appendix.

We first formalize the function classes that appear throughout this section. A function class is simply a set of input-output mappings. Once we specify what components a system may use and how they may interact, we obtain the collection of all functions $f : \mathcal{X} \to \mathcal{Y}$ that can be implemented by such a system. This plays a similar role as expressive families in neural networks: **a larger function class means the system can represent a broader range of behaviors**.

**Definition 4** (Induced Function Classes)**.** *Given candidate LLM pool $\mathcal{M}_{\text{LLM}}$, candidate FM pool $\mathcal{M}_{\text{FM}}$, topology pool $\Pi$, and a family of admissible interface pairs $\{(\phi_k, \psi_k)\}$, we define:*

- $\mathcal{F}_{\text{LLM}} = \{f : \mathcal{X} \to \mathcal{Y} \mid f(x) = A_{\text{LLM}}(\text{serialize}(x)),\ A_{\text{LLM}} \in \mathcal{M}_{\text{LLM}}\}$;
- $\mathcal{F}_{\text{Eywa}} = \{f : \mathcal{X} \to \mathcal{Y} \mid f$ *is implemented by some* $A_{\text{eywa}} = (A_{\text{LLM}}, F, \phi, \psi, \mathcal{C})\}$;
- $\mathcal{F}_{\text{LLM-MAS}}, \mathcal{F}_{\text{Eywa-MAS}}$ *are analogously defined at the system level over topologies* $\mathcal{G} \in \Pi$;
- $\mathcal{F}_{\text{Orch}} = \{f : \mathcal{X} \to \mathcal{Y} \mid f(x) = F_c(q, x),\ c \sim P(\cdot \mid q, x)$ *for some conductor* $P\}$.

By construction, $\mathcal{F}_{\text{LLM}} \subseteq \mathcal{F}_{\text{Eywa}} \subseteq \mathcal{F}_{\text{Eywa-MAS}} \subseteq \mathcal{F}_{\text{Orch}}$. This is because we can (1) choose an EywaAgent whose control policy never invokes the FM. Then the EywaAgent reduces exactly to an LLM; (2) choose a topology consisting of a single agent node and no nontrivial communication. Then the multi-agent system reduces to the original EywaAgent; (3) choose a conductor P that places all its probability mass on one fixed configuration corresponding to the given EywaMAS system. Then EywaOrchestra reproduces exactly that fixed system.

Table 4: Summary of notation used in the main paper and this appendix.

| Symbol | Meaning |
|---|---|
| ***Tasks and Data:*** | |
| $\mathcal{T}$ | Family of tasks with an underlying distribution. |
| $\tau = (q, x, y^\star, \ell)$ | Task instance: instruction, input, target, loss. |
| $\mathcal{Q}, \mathcal{X}, \mathcal{Y}$ | Instruction, input, and output spaces. |
| $\ell : \mathcal{Y} \times \mathcal{Y} \to \mathbb{R}_{\geq 0}$ | Task-specific loss. |
| $\mathcal{X} = \mathcal{X}_{\text{lng}} \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_m$ | Input factorization into linguistic and modality-specific parts. |
| $\pi_k : \mathcal{X} \to \mathcal{X}_k$ | Projection onto the $k$-th domain component, $x_k = \pi_k(x)$. |
| $T_k : \mathcal{X}_k \to \mathcal{X}_{\text{lng}}$ | Serialization map rendering $x_k$ into language tokens. |
| ***Agents and foundation models:*** | |
| $A_{\text{LLM}} : \mathcal{S} \to \Delta(\mathcal{M})$ | LLM agent: policy over response space $\mathcal{M}$. |
| $F_k : \mathcal{X}_k \times \mathcal{U}_k \to \mathcal{O}_k$ | Domain-specific foundation model for domain $k$. |
| $\phi_k : \mathcal{S} \to \mathcal{U}_k$ | Query compiler (language state $\to$ FM configuration). |
| $\psi_k : \mathcal{O}_k \to \mathcal{Z}_k$ | Response adapter (FM output $\to$ language-compatible context). |
| $A_{\text{eywa}} = (A_{\text{LLM}}, F, \phi, \psi, \mathcal{C})$ | EywaAgent. |
| $\mathcal{C} : \mathcal{S} \to \{\texttt{invoke}, \texttt{skip}\}$ | Control policy deciding whether to invoke the FM. |
| ***Multi-agent systems and orchestration:*** | |
| $\mathcal{M}_{\text{MAS}} = (\mathcal{A}, \mathcal{G})$ | Multi-agent system with agent set $\mathcal{A}$ and topology $\mathcal{G}$. |
| $\mathcal{N}(i)$ | Neighbors of agent $i$ in the communication graph $\mathcal{G}$. |
| $\mathcal{M}_{\text{Eywa}}$ | EywaMAS: MAS whose agents may be LLM agents or EywaAgents. |
| $\mathcal{O}_{\text{orch}} = (\mathcal{C}_{\text{cfg}}, P)$ | EywaOrchestra: configuration space and conductor. |
| $P : \mathcal{Q} \times \mathcal{X} \to \Delta(\mathcal{C}_{\text{cfg}})$ | Conductor mapping tasks to system configurations. |
| $\Pi$ | Finite topology pool. |
| $\mathcal{M}_{\text{LLM}}, \mathcal{M}_{\text{FM}}$ | Candidate LLM and FM pools. |
| ***Function classes and risk:*** | |
| $\mathcal{F}_{\text{LLM}}, \mathcal{F}_{\text{Eywa}}$ | Function classes induced by LLM-only and EywaAgents. |
| $\mathcal{F}_{\text{LLM-MAS}}, \mathcal{F}_{\text{Eywa-MAS}}$ | Function classes induced by LLM-only MAS and EywaMAS. |
| $\mathcal{F}_{\text{Orch}}$ | Function class induced by EywaOrchestra. |
| $\mathcal{R}(f) = \mathbb{E}_\tau[\ell(f(x), y^\star)]$ | Population risk. |
| $\mathcal{R}^\star_{\mathcal{F}} = \inf_{f \in \mathcal{F}} \mathcal{R}(f)$ | Minimum population risk over class $\mathcal{F}$. |
| $\hat{\mathcal{R}}_N(f)$ | Empirical risk over $N$ samples. |
| ***Information-theoretic quantities:*** | |
| $H(Y),\ H(Y \mid Z)$ | Entropy and conditional entropy. |
| $I(Y; Z)$ | Mutual information between $Y$ and $Z$. |
| $\mathbb{E}[Y \mid Z]$ | Conditional expectation. |

**Assumptions.** All subsequent theorems are derived under a common set of regularity and structural assumptions, which we collect here for clarity.

**Assumption 5** (Task Factorization). *For scientific tasks, the input space factorizes as*

$$\mathcal{X} = \mathcal{X}_{\text{lng}} \times \mathcal{X}_1 \times \cdots \times \mathcal{X}_m, \tag{7}$$

*where $\mathcal{X}_{\text{lng}}$ denotes language-observable context and each $\mathcal{X}_k$ denotes a domain-specific input. we further assume that the task loss is compatible with this factorization: there exist component-wise losses $\ell_{\text{lng}}$ and $\ell_k$ for $k = 1, \ldots, m$, together with a coordinate-wise nondecreasing aggregation function*

$$\Gamma : \mathbb{R}_{\geq 0}^{m+1} \to \mathbb{R}_{\geq 0}, \tag{8}$$

*such that*

$$\ell(\hat{y}, y^\star) = \Gamma\big(\ell_{\text{lng}}(\hat{y}, y^\star), \ell_1(\hat{y}, y^\star), \ldots, \ell_m(\hat{y}, y^\star)\big). \tag{9}$$

*Moreover, if two predictors are identical on all components except $k$, then a strict improvement in $\ell_k$ induces a strict improvement in the overall loss $\ell$.*

**Assumption 6** (Domain Advantage of Foundation Models). *Let $\pi_k : \mathcal{X} \to \mathcal{X}_k$ denote the projection onto the $k$-th domain component with $x_k = \pi_k(x)$. For any task instance $\tau = (q, x, y^\star, \ell) \sim \mathcal{T}$ with an informative component $x_k$,*

$$\mathbb{E}_{\tau\sim\mathcal{T}}[\ell_k(F_k(x_k), y^\star)] \;<\; \inf_{A_{\text{LLM}}} \mathbb{E}_{\tau\sim\mathcal{T}}\big[\ell_k\big(A_{\text{LLM}}(\text{serialize}(x_k)), y^\star\big)\big]. \tag{10}$$

*Moreover, there exists a non-empty task family $\mathcal{T}_1 \subseteq \mathcal{T}$ such that (1) the $k$-th component is sufficient for the task on $\mathcal{T}_1$, and (2) the foundation model solves this component perfectly, while every language-only agent incurs strictly positive loss on the serialized input. Formally,*

$$\mathbb{E}_{\tau\sim\mathcal{T}_1}[\ell_k(F_k(x_k), y^\star)] = 0, \qquad \inf_{A_{\text{LLM}}} \mathbb{E}_{\tau\sim\mathcal{T}_1}\big[\ell_k\big(A_{\text{LLM}}(\text{serialize}(x_k)), y^\star\big)\big] > 0. \tag{11}$$

**Assumption 7** (Non-Degenerate Serialization). *There exists at least one domain index $k \in \{1, \ldots, m\}$ and a set $E \subseteq \mathcal{X}_k$ of positive measure such that the serialization map $T_k : \mathcal{X}_k \to \mathcal{X}_{\text{lng}}$ is not sufficient for the target on $E$, i.e.,*

$$\mathbb{E}[Y \mid X_k] \neq \mathbb{E}[Y \mid T_k(X_k)] \quad \text{on a subset of positive probability within } E, \tag{12}$$

*where $Y$ denotes the task-relevant target variable.*

Assumption 7 ensures that the serialization genuinely discards task-relevant information, which is the precise condition under which heterogeneity delivers strict gains. Assumption 6 is implied by Assumption 7 under a specific loss (Section A.2), but we state them separately to align with the main paper.

**Assumption 8** (Performance-Preserving Interface). *Fix a domain index $k$. For any admissible interface pair $(\phi_k, \psi_k)$ used in an EywaAgent, whenever the agent invokes the foundation model $F_k$, the interface does not degrade the performance of the foundation model and is better than the performance of a language-only Agent. Concretely,*

$$\mathbb{E}_{\tau\sim\mathcal{T}}\big[\ell\big(f_{\text{Eywa}}(x), y^\star\big)\big] \;\leq\; \mathbb{E}_{\tau\sim\mathcal{T}}\big[\ell\big(F_k(x_k), y^\star\big)\big].$$

Assumption 8 formalizes the intuition that the FM–LLM "Tsaheylu" interface is faithful: a correctly configured call to $F_k$ recovers the Bayes-optimal conditional expectation given the domain-specific input.

### A.2 Information-Theoretic Analysis of the Language Interface Bottleneck

We begin with an information-theoretic characterization of why serializing a non-linguistic input $X_k$ into language tokens fundamentally limits the expressivity of language-only agents. All results in this subsection are derived for a single modality $k$; we drop the subscript for brevity and write $X := X_k$, $T := T_k$.

**Lemma 9** (Data Processing Inequality for Serialization). *Let $(X, Y)$ be jointly distributed random variables and let $T : \mathcal{X} \to \mathcal{X}_{\text{lng}}$ be any measurable serializer. Then*

$$I(Y; T(X)) \;\leq\; I(Y; X), \tag{13}$$

*with equality if and only if $I(Y; X \mid T(X)) = 0$.*

*Proof.* Since $Y \to X \to T(X)$ forms a Markov chain as $T(X)$ is computed from $X$, the classical data processing inequality yields $I(Y;T(X)) \le I(Y;X)$, a.k.a., $Y \perp T(X) \mid X$. Equality holds if and only if conditioning on $T(X)$ yields the same information about $Y$ as conditioning on $X$. □

**Lemma 10** (Irreducible Bayes Risk Gap under Serialization)**.** *Under squared loss $\ell(\hat{y}, y) = \|\hat{y} - y\|^2$ with $Y \in L^2$, define*

$$\mathcal{R}_X^\star = \inf_{g:\mathcal{X}\to\mathcal{Y}} \mathbb{E}\big[\|Y - g(X)\|^2\big], \qquad \mathcal{R}_T^\star = \inf_{h:\mathcal{X}_{\text{lng}}\to\mathcal{Y}} \mathbb{E}\big[\|Y - h(T(X))\|^2\big]. \tag{14}$$

*Then*

$$\mathcal{R}_T^\star - \mathcal{R}_X^\star \;=\; \mathbb{E}\Big[\big\|\mathbb{E}[Y \mid X] - \mathbb{E}[Y \mid T(X)]\big\|^2\Big] \;\ge\; 0, \tag{15}$$

*and $\mathcal{R}_T^\star > \mathcal{R}_X^\star$ whenever Assumption 7 holds.*

*Proof.* Under squared loss, the Bayes predictors are $g^\star(X) = \mathbb{E}[Y \mid X]$ and $h^\star(T(X)) = \mathbb{E}[Y \mid T(X)]$. Hence

$$\mathcal{R}_X^\star = \mathbb{E}\big[\|Y - \mathbb{E}[Y \mid X]\|^2\big], \qquad \mathcal{R}_T^\star = \mathbb{E}\big[\|Y - \mathbb{E}[Y \mid T(X)]\|^2\big]. \tag{16}$$

Split $Y - \mathbb{E}[Y \mid T(X)]$ as two parts. Then take squared norms and expectation,

$$Y - \mathbb{E}[Y \mid T(X)] = \big(Y - \mathbb{E}[Y \mid X]\big) + \big(\mathbb{E}[Y \mid X] - \mathbb{E}[Y \mid T(X)]\big). \tag{17}$$

$$\mathcal{R}_T^\star = \mathcal{R}_X^\star + \mathbb{E}\Big[\big\|\mathbb{E}[Y \mid X] - \mathbb{E}[Y \mid T(X)]\big\|^2\Big] + 2\,\mathbb{E}[\langle Y - \mathbb{E}[Y \mid X],\, \mathbb{E}[Y \mid X] - \mathbb{E}[Y \mid T(X)]\rangle]\,. \tag{18}$$

Since $T(X)$ is a function of $X$, the term $\mathbb{E}[Y \mid X] - \mathbb{E}[Y \mid T(X)]$ is $\sigma(X)$-measurable. By the orthogonality property of conditional expectation, $Y - \mathbb{E}[Y \mid X]$ is orthogonal to every $\sigma(X)$-measurable square-integrable random variable, so the cross term vanishes. Here, $\sigma(Z)$ denotes the information generated by a random variable $Z$. Therefore,

$$\mathcal{R}_T^\star = \mathcal{R}_X^\star + \mathbb{E}\Big[\big\|\mathbb{E}[Y \mid X] - \mathbb{E}[Y \mid T(X)]\big\|^2\Big]\,. \tag{19}$$

which is the claimed identity in Equation 15. Strict inequality follows directly from Assumption 7, which guarantees that $\mathbb{E}[Y \mid X] \neq \mathbb{E}[Y \mid T(X)]$ on a set of positive probability. □

Lemmas 9 and 10 establish that, under mild non-degeneracy, any language-only pipeline that filters $X$ through a textualization $T$ pays a strictly positive statistical price. This quantitative gap is the information-theoretic root cause of the Domain Advantage assumption (Assumption 6). The following proposition translates this observation from Bayes-optimal predictors to realized function classes.

**Proposition 11** (Lower Bound on LLM-only Risk)**.** *Under Assumption 7, for any language-only function class $\mathcal{F}_{\text{LLM}}$ whose members factor through the serializer $T$,*

$$\inf_{f\in\mathcal{F}_{\text{LLM}}} \mathcal{R}(f) \;\ge\; \mathcal{R}_T^\star \;>\; \mathcal{R}_X^\star. \tag{20}$$

*Proof.* Every $f \in \mathcal{F}_{\text{LLM}}$ admits a factorization $f(x) = h(T(x))$ for some measurable $h$, by Definition 4. Consequently,

$$\inf_{f\in\mathcal{F}_{\text{LLM}}} \mathcal{R}(f) \;\ge\; \inf_h \mathbb{E}\big[\ell(h(T(X)), Y)\big] \;=\; \mathcal{R}_T^\star.$$

The strict inequality $\mathcal{R}_T^\star > \mathcal{R}_X^\star$ is Lemma 10 under assumption 7. □

Proposition 11 is the central building block that allows us to prove strict improvements in later subsections. It rules out the possibility that a sufficiently clever language-only agent could ever recover the advantage of a direct-access foundation model.

### A.3 Expressivity and Solvability of EywaAgent

We now prove the improvement of EywaAgent over language-only agent. We further establish two additional results: the EywaAgent containment $\mathcal{F}_{\text{LLM}} \subseteq \mathcal{F}_{\text{Eywa}}$ and the unboundedness of the expressivity gap.

**Proposition 12** (EywaAgent Containment)**.** *Under Definition 4,*

$$\mathcal{F}_{\text{LLM}} \subseteq \mathcal{F}_{\text{Eywa}}.$$

*Proof.* Fix any $f_{\text{LLM}} \in \mathcal{F}_{\text{LLM}}$ realized by an LLM agent $A_{\text{LLM}}$. Consider the EywaAgent

$$A_{\text{eywa}} = (A_{\text{LLM}}, F, \phi, \psi, \mathcal{C}_{\text{skip}}) \tag{21}$$

where $\mathcal{C}_{\text{skip}}(s) \equiv \texttt{skip}$ for all $s \in \mathcal{S}$. By the semantics in Section 3 of the main paper, under $\mathcal{C}_{\text{skip}}$ the Eywa pipeline reduces to $z^{(t)} = A_{\text{LLM}}(s^{(t)})$, which recovers the original LLM agent. Hence $f_{\text{LLM}}$ can be realized by an EywaAgent, so $f_{\text{LLM}} \in \mathcal{F}_{\text{Eywa}}$. □

**Theorem 13** (Improvement of EywaAgent over Language-only Agent)**.** *Let $\mathcal{F}_{\text{LLM}}$ and $\mathcal{F}_{\text{Eywa}}$ be the function classes induced by language-only agents and EywaAgents, respectively, as in Definition 4. Under Assumption 6,*

1. ***Strict Optimal Risk Improvement:***

$$\inf_{f \in \mathcal{F}_{\text{Eywa}}} \mathbb{E}_{\tau \sim \mathcal{T}}[\ell(f(x), y^\star)] < \inf_{f \in \mathcal{F}_{\text{LLM}}} \mathbb{E}_{\tau \sim \mathcal{T}}[\ell(f(x), y^\star)]. \tag{22}$$

2. ***Expanded Solvable Task Space:*** *there exists a non-empty task family $\mathcal{T}_1 \subset \mathcal{T}$ such that*

$$\inf_{f \in \mathcal{F}_{\text{Eywa}}} \mathbb{E}_{\tau \sim \mathcal{T}_1}[\ell(f(x), y^\star)] = 0, \qquad \inf_{f \in \mathcal{F}_{\text{LLM}}} \mathbb{E}_{\tau \sim \mathcal{T}_1}[\ell(f(x), y^\star)] > 0. \tag{23}$$

*Proof.* **Part 1 (Strict Optimal Risk Improvement).** By Proposition 12, $\mathcal{F}_{\text{LLM}} \subseteq \mathcal{F}_{\text{Eywa}}$, so

$$\inf_{f \in \mathcal{F}_{\text{Eywa}}} \mathcal{R}(f) \leq \inf_{f \in \mathcal{F}_{\text{LLM}}} \mathcal{R}(f). \tag{24}$$

To prove strictness, consider the informative domain index $k$ from Assumption 7. By construction of EywaAgent, there exists an agent $f^\dagger \in \mathcal{F}_{\text{Eywa}}$ that invokes the foundation model $F_k$ on the $k$-th component while leaving the remaining components unchanged relative to a language-only baseline. Therefore, its $k$-th component loss is no worse than that of $F_k$, whereas the non-$k$ components are unchanged. By Assumption 6,

$$\mathbb{E}_{\tau \sim \mathcal{T}}[\ell_k(F_k(x_k), y^\star)] < \inf_{A_{\text{LLM}}} \mathbb{E}_{\tau \sim \mathcal{T}}\big[\ell_k(A_{\text{LLM}}(\text{serialize}(x_k)), y^\star)\big]. \tag{25}$$

Hence $f^\dagger$ achieves strictly smaller loss on the $k$-th component than any language-only agent. Since all other components are unchanged, Assumption 5 implies that this strict improvement on component $k$ induces a strict improvement in the overall task loss. Thus

$$\mathcal{R}(f^\dagger) < \inf_{f \in \mathcal{F}_{\text{LLM}}} \mathcal{R}(f) \Longrightarrow \inf_{f \in \mathcal{F}_{\text{Eywa}}} \mathcal{R}(f) < \inf_{f \in \mathcal{F}_{\text{LLM}}} \mathcal{R}(f). \tag{26}$$

**Part 2 (Expanded Solvable Task Space).** According to Assumption 6, consider a non-empty task family $\mathcal{T}_1 \subseteq \mathcal{T}$ for which the task is fully determined by the $k$-th domain-specific component, and the foundation model solves this component perfectly, i.e.,

$$\mathbb{E}_{\tau \sim \mathcal{T}_1}[\ell_k(F_k(x_k), y^\star)] = 0, \tag{27}$$

Then, by the same construction as above, there exists an EywaAgent in $\mathcal{F}_{\text{Eywa}}$ whose prediction is perfect on the only task-relevant component. This yields

$$\inf_{f \in \mathcal{F}_{\text{Eywa}}} \mathbb{E}_{\tau \sim \mathcal{T}_1}[\ell(f(x), y^\star)] = 0. \tag{28}$$

On the other hand, every language-only agent has strictly positive loss on the relevant component, hence

$$\inf_{f \in \mathcal{F}_{\text{LLM}}} \mathbb{E}_{\tau \sim \mathcal{T}_1}[\ell(f(x), y^\star)] > 0. \tag{29}$$

Therefore, the solvable task space of EywaAgent is strictly larger than that of language-only agents. □

### A.4 Multi-Agent Propagation: EywaMAS

We now lift the single-agent analysis to multi-agent systems. Let $\mathcal{M}_{\text{LLM-MAS}}$ denote a MAS in which every agent is an LLM agent, and $\mathcal{M}_{\text{Eywa}}$ an EywaMAS with at least one EywaAgent. We first show that any LLM-only MAS remains constrained by the information bottleneck of Section A.2.

**Lemma 14** (Information Closure of LLM-only MAS). *Let $\mathcal{M}_{\text{LLM-MAS}} = (\mathcal{A}_{\text{LLM}}, \mathcal{G})$ be any finite-horizon LLM-only MAS operating on input $X$ via the serialization $T(X)$. Let $\hat{Y}$ denote its final output. Then*

$$I(Y;\hat{Y}) \;\le\; I(Y;T(X)) \;\le\; I(Y;X). \tag{30}$$

*Proof.* Since every agent in $\mathcal{A}_{\text{LLM}}$ consumes and produces only language messages, the joint random variable of all messages exchanged over the entire interaction forms a (possibly very long) Markov chain emanating from $T(X)$: $Y \to X \to T(X) \to M_1 \to M_2 \to \cdots \to \hat{Y}$. The data processing inequality then gives $I(Y;\hat{Y}) \le I(Y;T(X))$. Since the same data processing was applied, $I(Y;T(X)) \le I(Y;X)$. □

Lemma 14 formalizes the intuition that "more LLM agents cannot create information that was discarded at serialization time". We next argue that a single EywaAgent in EywaMAS, as long as its message can reach the final output node, propagates its recovered information through the entire system.

**Theorem 15** (Communication-Enhanced Solvability of EywaMAS). *Let $\mathcal{M}_{\text{Eywa}} = (\mathcal{A}, \mathcal{G})$ be an EywaMAS and $\mathcal{M}_{\text{LLM-MAS}}$ an LLM-only MAS with the same topology $\mathcal{G}$. Suppose that (i) there exists an EywaAgent $\mathcal{A}_k \in \mathcal{A}$ with access to $F_k$, and (ii) the topology $\mathcal{G}$ has finite diameter $D$ and the interaction horizon $T \ge D$, so that the message produced by $\mathcal{A}_k$ reaches the final output node within the interaction horizon. Under Assumptions 5, 6, 7, and 8, there exists a non-empty task family $\mathcal{T}_1 \subseteq \mathcal{T}$ such that*

$$\inf_{f \in \mathcal{F}_{\text{Eywa-MAS}}} \mathbb{E}_{\tau\sim\mathcal{T}_1}[\ell(f(x), y^\star)] = 0, \qquad \inf_{f \in \mathcal{F}_{\text{LLM-MAS}}} \mathbb{E}_{\tau\sim\mathcal{T}_1}[\ell(f(x), y^\star)] > 0. \tag{31}$$

*Proof.* **Zero-risk attainability by EywaMAS.** Let $\mathcal{T}_1$ be the task family from Assumption 6 on which the $k$-th component is sufficient for the task and $F_k$ solves it perfectly: $\mathbb{E}_{\tau\sim\mathcal{T}_1}[\ell_k(F_k(x_k), y^\star)] = 0$. Consider the EywaMAS in which $\mathcal{A}_k$ invokes $F_k$ on $x_k$ and forwards its adapted output $z_k = \psi_k(F_k(x_k, u_k))$ through the graph. By condition (ii), $z_k$ is available to the final output node within the interaction horizon, which may simply copy it (a trivial operation in language). The resulting system-level function $f_{\text{Eywa-MAS}} \in \mathcal{F}_{\text{Eywa-MAS}}$ therefore satisfies $\ell_k(f_{\text{Eywa-MAS}}(x), y^\star) = 0$ on $\mathcal{T}_1$. Moreover, since the non-$k$ components can be handled exactly as in a language-only baseline (reachability only affects the task-relevant channel), Assumption 8 ensures that the full Eywa system loss is no worse than the FM loss on $\mathcal{T}_1$, so

$$\inf_{f \in \mathcal{F}_{\text{Eywa-MAS}}} \mathbb{E}_{\tau\sim\mathcal{T}_1}[\ell(f(x), y^\star)] \;\le\; \mathbb{E}_{\tau\sim\mathcal{T}_1}[\ell(F_k(x_k), y^\star)] = 0, \tag{32}$$

where the final equality uses Assumption 5 together with sufficiency of the $k$-th component on $\mathcal{T}_1$, which reduces $\ell$ to $\ell_k$ on this family.

**Strict positivity for LLM-only MAS.** For any LLM-only MAS with the same topology, all messages and final outputs are functions of language-serialized inputs. By Lemma 14, such a system cannot recover task-relevant information about $x_k$ that is lost during serialization. Hence, by Assumption 5 and 6, strict positivity of the relevant component loss implies strict positivity of the overall task loss.

$$\inf_{f \in \mathcal{F}_{\text{LLM-MAS}}} \mathbb{E}_{\tau\sim\mathcal{T}_1}[\ell_k(f(x), y^\star)] > 0 \Longrightarrow \inf_{f \in \mathcal{F}_{\text{LLM-MAS}}} \mathbb{E}_{\tau\sim\mathcal{T}_1}[\ell(f(x), y^\star)] > 0. \tag{33}$$

□

### A.5 Adaptive Orchestration: EywaOrchestra

We now develop the theoretical foundation of *EywaOrchestra*, which couples *model adaptivity* (selecting LLM and FM backbones per task) with *structural adaptivity* (selecting topologies from $\Pi$).

Let $F_c$ denote the agent system instantiated by configuration $c$. For brevity, denote the conditional risk of configuration $c \in \mathcal{C}_{\text{cfg}}$ at task input $(q, x)$ by

$$r(c; q, x) := \mathbb{E}\big[\ell(F_c(q, x), y^\star) \mid q, x\big]. \tag{34}$$

We show that, whenever different task regions favor different configurations, *any* adaptive conductor, even one limited to routing over a finite pool, dominates the best fixed multi-agent system.

**Theorem 16** (Oracle Adaptivity Strictly Improves over Fixed Systems)**.** *Let $\mathcal{C}_{\text{cfg}}$ be a finite set of candidate configurations, and let*

$$r(c; q, x) := \mathbb{E}\big[\ell(F_c(q, x), y^\star) \mid q, x\big]$$

*denote the conditional risk of configuration $c$ at input $(q, x)$. Let $\tau = (q, x, y^\star, \ell) \sim \mathcal{T}$ denote the task distribution.*

*Define the best fixed-configuration risk and the oracle adaptive risk as*

$$\mathcal{R}^\star_{\text{fixed}} = \min_{c \in \mathcal{C}_{\text{cfg}}} \mathbb{E}_{\tau \sim \mathcal{T}}[\ell(F_c(q, x), y^\star)], \qquad \mathcal{R}_{\text{oracle}} = \mathbb{E}_{\tau \sim \mathcal{T}}\left[\min_{c \in \mathcal{C}_{\text{cfg}}} r(c; q, x)\right]. \tag{35}$$

*Then $\mathcal{R}_{\text{oracle}} \leq \mathcal{R}^\star_{\text{fixed}}$. Moreover, then the inequality is strict if for every fixed configuration $c \in \mathcal{C}_{\text{cfg}}$,*

$$\mathbb{P}\left(r(c; q, x) > \min_{c' \in \mathcal{C}_{\text{cfg}}} r(c'; q, x)\right) > 0, \tag{36}$$

*In other words, the inequality is strict if no fixed configuration achieves optimality for all tasks.*

*Proof.* For any fixed configuration $c_0 \in \mathcal{C}_{\text{cfg}}$, we have pointwise risks which we can take expectation over

$$\min_{c \in \mathcal{C}_{\text{cfg}}} r(c; q, x) \leq r(c_0; q, x) \Longrightarrow \mathbb{E}_\tau\left[\min_c r(c; q, x)\right] \leq \mathbb{E}_\tau[r(c_0; q, x)]. \tag{37}$$

Since this holds for every $c_0$, taking the minimum over $c_0$ yields

$$\mathcal{R}_{\text{oracle}} \leq \min_{c_0 \in \mathcal{C}_{\text{cfg}}} \mathbb{E}_\tau[r(c_0; q, x)] = \mathcal{R}^\star_{\text{fixed}}.$$

For strictness, let $c^\star$ be a best fixed configuration achieving $\mathcal{R}^\star_{\text{fixed}}$. By assumption,

$$\mathbb{P}\Big(r(c^\star; q, x) > \min_c r(c; q, x)\Big) > 0.$$

Therefore, we have the strict inequality,

$$\mathbb{E}_\tau\left[\min_c r(c; q, x)\right] < \mathbb{E}_\tau[r(c^\star; q, x)] = \mathcal{R}^\star_{\text{fixed}},$$

□

### A.6 Efficiency and Token Complexity

Beyond task performance, Eywa delivers strong efficiency gains, as evidenced by the ∼30% token reduction reported in Section 5. We formalize this benefit by comparing the token complexities of language-only agents and EywaAgents when processing a structured input $x_k \in \mathcal{X}_k$.

**Proposition 17** (Token Complexity)**.** *Let $L(x_k)$ denote the total number of language tokens an LLM-only agent spends on $x_k \in \mathcal{X}_k$, including both the serialization of $x_k$ in its prompt and any chain-of-thought reasoning required to analyze it. Let $L_{\text{call}}$ denote the token length of a structured FM invocation produced by $\phi_k$, and $L_\psi(o_k)$ the token length of the adapted response $\psi_k(o_k)$. Then the language-token cost of processing $x_k$ satisfies*

$$\text{TokenCost}_{\text{LLM}}(x_k) = \Theta(L(x_k)), \tag{38}$$

$$\text{TokenCost}_{\text{Eywa}}(x_k) = O(L_{\text{call}} + L_\psi(o_k)). \tag{39}$$

*In typical scientific modalities of interest (long time series, tables with many rows), $L(x_k) \gg L_{\text{call}} + L_\psi(o_k)$, so the ratio $\text{TokenCost}_{\text{Eywa}}/\text{TokenCost}_{\text{LLM}} \to 0$ as the modality size grows.*

Table 5: Summary of theoretical results and the claims they justify.

| Result | Key Statement |
|---|---|
| *Information-theoretic bottleneck (Section A.2).* | |
| Lemma 9 | Serialization cannot increase information: $I(Y;T(X)) \le I(Y;X)$. |
| Lemma 10 | Strict Bayes-risk gap $\mathcal{R}^\star_T > \mathcal{R}^\star_X$ whenever the serializer discards task-relevant information. |
| Proposition 11 | Every language-only function class is lower-bounded by the serialized Bayes risk $\mathcal{R}^\star_T$. |
| *EywaAgent expressivity and solvability (Section A.3).* | |
| Proposition 12 | $\mathcal{F}_{\text{LLM}} \subseteq \mathcal{F}_{\text{Eywa}}$: an EywaAgent can always fall back to an LLM agent. |
| Theorem 13 | EywaAgent strictly improves the optimal risk and expands the set of perfectly solvable tasks. |
| *Multi-agent propagation (Section A.4).* | |
| Lemma 14 | LLM-only MAS cannot exceed the mutual information retained by serialization. |
| Theorem 15 | EywaMAS strictly solves tasks that no LLM-only MAS of the same topology can solve. |
| *Adaptive orchestration (Section A.5).* | |
| Theorem 16 | Oracle adaptive routing attains $\mathcal{R}_{\text{oracle}} \le \mathcal{R}^\star_{\text{fixed}}$, with strict inequality whenever no fixed configuration is uniformly optimal. |
| *Efficiency (Section A.6).* | |
| Proposition 17 | EywaAgent language-token cost is $O(L_{\text{call}} + L_\psi(o_k))$, independent of modality size, vs. $\Theta(L(x_k))$ for LLM-only agents. |
| Proposition 18 | Wall-clock latency inherits the same asymptotic separation as the token complexity. |

*Proof.* A language-only agent must both include $x_k$ in its prompt and perform any reasoning over it within the same language channel, so its combined prompt-plus-reasoning token count is $\Theta(L(x_k))$ by definition. An EywaAgent instead routes $x_k$ to the foundation model $F_k$ through a structured call of length $L_{\text{call}}$. No reasoning tokens over $x_k$ are required on the language side. Summing these two contributions gives the stated bound. For time series or tables with $n$ entries, $L(x_k) = \Theta(n)$ at minimum from the serialization alone, while both $L_{\text{call}}$ and $L_\psi(o_k)$ are typically constant or polylogarithmic in $n$. □

**Proposition 18** (Wall-Clock Latency)**.** *Let the per-token latency of the LLM be $\alpha_{\text{LLM}}$ and the FM call latency be $\alpha_{\text{FM}}$ (independent of $L(x_k)$). Then*

$$\text{Latency}_{\text{LLM}}(x_k) = \Theta(\alpha_{\text{LLM}} \cdot L(x_k)), \text{Latency}_{\text{Eywa}}(x_k) = O(\alpha_{\text{LLM}} \cdot (L_{\text{call}} + L_\psi(o_k)) + \alpha_{\text{FM}}).$$

*Proof.* Both bounds follow from Proposition 17 and the additive latency contribution of an FM call. □

In practice, the latency constant $\alpha_{\text{FM}}$ in Proposition 18 is typically much smaller than the LLM processing cost $\alpha_{\text{LLM}} \cdot L(x_k)$ for several reasons. First, the domain-specific foundation models are typically smaller than frontier LLMs in parameter count and are specialized to fixed-shape tensor inputs, which makes their inference both inherently lightweight and highly accelerator-friendly. Second, unlike LLM calls that are usually served through remote API endpoints, an FM invocation in Eywa is executed through a local remote server backend, avoiding network round-trip latency and API rate limits. Third, many FMs of interest admit further acceleration through batching, precomputed embeddings, or caching at the Tsaheylu Context Protocol layer. Consequently, $\alpha_{\text{FM}} \le \alpha_{\text{LLM}} \cdot L(x_k)$ whenever $x_k$ is non-trivial, and the Eywa latency in Proposition 18 is dominated by the (drastically reduced) LLM-side term $\alpha_{\text{LLM}} \cdot (L_{\text{call}} + L_\psi(o_k))$.

### A.7 Summary of Theoretical Results

We close this section by consolidating the theoretical results developed above. Table 5 lists every result together with a one-line summary of its key statement, so that the logical structure of the analysis can

be read off at a glance. The results are organized along the same storyline as the main paper: from the information-theoretic characterization of the language interface bottleneck (Section A.2), to the single-agent and multi-agent expressivity guarantees of EywaAgent and EywaMAS (Sections A.3–A.4), to the adaptivity result for EywaOrchestra (Section A.5), and finally to the efficiency analysis that explains Eywa's empirical token and latency savings (Section A.6).

Read together, these results provide an end-to-end theoretical justification for the Eywa framework. The information-theoretic bottleneck (Lemmas 9–10 and Proposition 11) identifies a first-principles reason why language-only agents are fundamentally limited on scientific inputs: the serialization step discards task-relevant information that no amount of downstream reasoning can recover. The expressivity analysis (Proposition 12 and Theorem 13) then shows that this bottleneck is closed at the single-agent level by EywaAgent, which contains the LLM-only class while achieving strictly smaller risk and strictly enlarging the set of perfectly solvable tasks. Lemma 14 and Theorem 15 lift this advantage from the single-agent to the multi-agent setting: under mild reachability, a single EywaAgent is sufficient to propagate the recovered information to the system output, while LLM-only MAS remain subject to the same serialization bottleneck. Theorem 16 further shows that once the configuration space is heterogeneous, task-adaptive orchestration provides an additional strict improvement over any fixed configuration. Finally, Propositions 17 and 18 connect this theory to empirically observable efficiency: by offloading modality-specific computation to a foundation model, EywaAgent removes the $\Theta(L(x_k))$ scaling that language-only pipelines pay in tokens and latency, matching the $\sim$30% token reduction reported in our experiments (Section 5).

# B Further Discussions

## B.1 More Related Works

### B.1.1 Agentic AI Systems.

In recent years, LLMs and AI Agents are becoming more powerful due to comprehensive pre-training (OpenAI, 2026), advanced backbone architectures (Qiu et al., 2025b; Zhang et al., 2025b), training algorithms (DeepSeek-AI, 2025), and extended modalities supports (Tong et al., 2026; Seed, 2025; Qwen Team, 2025). As a result, LLM and AI Agent applications are revolutionizing a variety of industries (Luo et al., 2025). More recently, we identify three trends which are related to our *Eywa*.

**Agentic Reasoning.** Real-world tasks are often too complex to be solved by AI agents in a single forward pass after receiving an instruction (Liu et al., 2026). Instead, agentic systems typically approach such tasks through structured reasoning procedures. A common paradigm is to first formulate an explicit plan or decompose the task into manageable subgoals (Shen et al., 2023; Ning et al., 2026), and then solve these subgoals through iterative execution, reflection, clarification, and continuation across multiple interaction turns (Tian et al., 2025; Merrill et al., 2026; Ning et al., 2026). This line of work highlights the importance of moving beyond one-shot language generation toward more deliberate and interactive reasoning processes. Scientific tasks exhibit similar, and often more pronounced, complexity: they may require reasoning over specialized data modalities, domain-specific constraints, long-horizon dependencies, and expert-level predictive models (Jumper et al., 2021; Merchant et al., 2023; Chen et al., 2025; Wei et al., 2025; Georgiev et al., 2025). However, most existing agentic reasoning frameworks remain primarily centered on the language capabilities of AI agents. Our *Eywa* is motivated by this gap. Rather than relying solely on language agents to reason over scientific problems, *Eywa* introduces domain-specific foundation models into agentic systems, enabling specialized scientific models to contribute domain representations, predictions, and feedback throughout the reasoning process.

**Orchestrated Intelligence.** Building on the capabilities of individual agents, orchestrated intelligence has emerged as a central paradigm for tackling complex tasks that exceed the reach of any single component in the system. Early frameworks such as AutoGen (Wu et al., 2023) and MetaGPT (Hong et al., 2024) demonstrate that conversational collaboration among role-playing agents can effectively decompose and solve intricate problems. Beyond flat conversational patterns, recent efforts have begun to explicitly investigate the orchestration layer itself. AgentOrchestra (Zhang et al., 2025a) adopts a hierarchical architecture in which a planning agent dispatches specialist sub-agents through a tool–environment–agent protocol, while evolving-orchestration approaches (Dang et al., 2025) train a centralized orchestrator via reinforcement learning to adaptively sequence and prioritize agents according to the evolving task state. Difficulty-aware orchestration (Su et al., 2025c) further tailors multi-agent workflows to query-level complexity through learned routing. Despite this progress, existing orchestration frameworks remain largely language-centric: they primarily route, sequence, or coordinate agents that communicate through natural language, while non-linguistic expert models are often treated as passive callable tools. This design is insufficient for scientific domains, where reasoning may depend on heterogeneous representations such as molecular structures, crystal graphs, time series, spatial fields, and tabular measurements. Our *EywaOrchestra* addresses this limitation by dynamically orchestrating heterogeneous experts, including LLM agents and domain-specific foundation models. By jointly selecting agent configurations, model specializations, and communication topologies, *EywaOrchestra* extends orchestrated intelligence from language-agent workflow optimization toward heterogeneous scientific model collaboration.

### B.1.2 Modeling Structured Data with LLMs.

Structured data modeling has long been a central problem in machine learning, spanning data types such as graphs, tables, time series, knowledge graphs, and other relational or compositional objects. Since the rise of deep learning, a wide range of neural architectures have been developed to capture the inductive biases of different structured domains, including graph neural networks, temporal models, tabular learning methods, and representation learning techniques for relational data (Wu et al., 2020; Lim & Zohren, 2021; Borisov et al., 2022; Antelmi et al., 2023; Han et al., 2022; Fu, 2023). More recently, the rapid development of large language models and AI agents has opened new

possibilities for leveraging language-based reasoning, instruction following, and tool use to support structured data modeling.

**LLMs directly handle structured data.** One line of work investigates whether LLMs can directly process structured data by converting non-textual structures into textual or sequence-based representations (Tang et al., 2024; Shi et al., 2025; Jing et al., 2026; Tan et al., 2024; Jiang et al., 2023). These approaches benefit from the general reasoning and in-context learning capabilities of LLMs, enabling them to perform tasks such as question answering, classification, forecasting, and relational reasoning over structured inputs. However, direct textualization may lose important structural information, especially when the data contains long-range dependencies, high-order relations, precise numerical values, or domain-specific semantics that are difficult to faithfully express in natural language (Sui et al., 2024; Ghosh et al., 2025; Sadeghi et al., 2024; Zhao et al., 2025; Yang et al., 2025).

**LLMs to help existing models.** Another line of work uses LLMs as assistants to improve existing structured data models rather than replacing them (Jin et al., 2024; Chen et al., 2023b; Li et al., 2025a). For example, textual metadata or descriptions produced by LLMs can be combined with graph, tabular, or temporal encoders to enrich structured representations (Su et al., 2025b; Yan et al., 2023; Li et al., 2025b; Su et al., 2025a; Zhu et al., 2025). LLMs can also act as controllers that select models, design prompts or queries, invoke external tools, and interpret the outputs of specialized predictors (Trirat et al., 2025; Tornede et al., 2024; Luo et al., 2024). These methods preserve the strengths of domain-specific architectures while using LLMs to provide semantic knowledge and flexible reasoning.

**Re-programming LLMs for structured data.** A third direction adapts LLMs themselves to structured data through re-programming, prompting, or lightweight adaptation. Instead of training a structured-data model from scratch, these methods map structured inputs into the embedding or token space of a pretrained language model, allowing the LLM to be reused for tasks beyond natural language (Jin et al., 2023; Perozzi et al., 2024; Kim et al., 2025a; Lu et al., 2024). This paradigm leverages the broad prior knowledge and scaling properties of LLMs, but it also faces challenges in preserving structural fidelity, handling large-scale inputs, and ensuring that the language model's inductive biases align with the target scientific domain.

Despite these advances, most existing approaches still center on adapting structured data to the language-model interface. In contrast, many scientific problems are naturally handled by specialized foundation models that operate directly over structured representations. Our work therefore takes a complementary perspective: rather than forcing all structured data into language, *Eywa* introduces domain-specific foundation models into agentic systems, allowing LLM agents to coordinate, invoke, and reason with specialized models that retain their native structured-data inductive biases.

### B.2 Future Directions

**Scaling heterogeneous scientific model ecosystems.** A natural future direction is to scale heterogeneous agentic systems beyond a small set of manually integrated domain-specific foundation models. Enabling heterogeneous components to collaborate within a unified agentic system remains challenging from the engineering perspective, because different models may operate over incompatible input formats and provide outputs with varying degrees of interpretability. As the number of available experts grows, the system must also determine which models are relevant to a task, whether their predictions are reliable, and how conflicting evidence from multiple experts should be reconciled. Building such an ecosystem requires more than simply adding more APIs. It calls for standardized model interfaces, rich metadata descriptions, and capability profiling. Addressing these challenges would move heterogeneous agentic systems from small-scale proof-of-concept integrations toward scalable scientific model ecosystems.

**Learning better orchestration policies.** In this work, orchestration is primarily driven by task-level reasoning over available agents, foundation models, and communication topologies. A promising direction is to learn orchestration policies from interaction data, execution traces, and task outcomes. Future systems may train orchestrators to predict which expert models to invoke, when to invoke them, and which communication topology is most suitable for a given task.

**More advanced integration between LLMs and scientific foundation models.** The stronger "Tsaheylu" is, the better. Future work could explore tighter coupling mechanisms beyond Tsaheylu

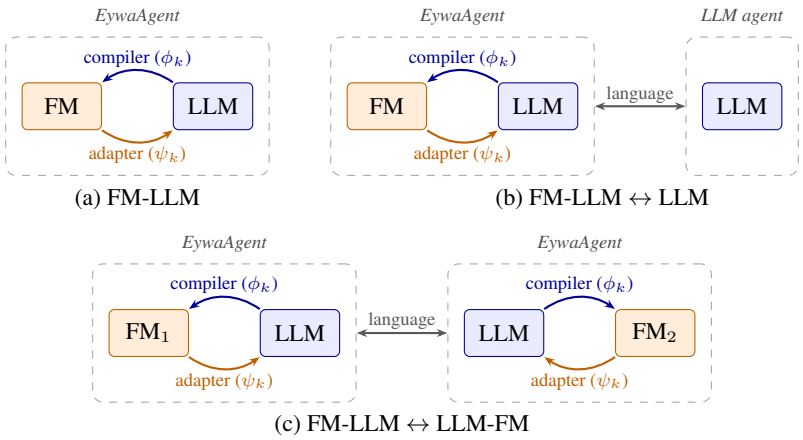



Figure 6: Three collaboration patterns in *Eywa*. (a) Within an *EywaAgent*, the language model configures a foundation model through the query compiler $\phi_k$, and the result returns through the response adapter $\psi_k$. (b) Information produced by a foundation model is propagated to other language agents. (c) Foundation models that cannot communicate directly exchange information through their language interfaces.

context protocol, including shared representation spaces, differentiable interfaces, bidirectional alignment between language and scientific embeddings, and memory/skill mechanisms that preserve domain-specific evidence across reasoning steps.

**Extending EywaBench for heterogeneous scientific reasoning.** Another important direction is to further extend *EywaBench* into a broader benchmark suite for heterogeneous scientific reasoning. While *EywaBench* provides an initial testbed, scientific reasoning spans a much wider range of domains, data modalities, task formats, and expert models. Future extensions could incorporate more datasets as well as additional data types.. Beyond expanding coverage, *EywaBench* can also be extended to evaluate richer capabilities such as model selection, cross-domain evidence integration, communication efficiency, and adaptive topology selection. Such extensions would make *EywaBench* a more comprehensive testbed for understanding when heterogeneous orchestration is genuinely beneficial over language-only agents, single-domain models, or static multi-agent workflows.

### B.3 Design Choices

**Collaboration.** In *Eywa*, collaboration refers to system-level interaction between heterogeneous models, which takes three basic patterns (Figure 6). (a) *FM-LLM*: within an *EywaAgent*, the language model configures a domain-specific foundation model, whose output informs subsequent reasoning. (b) *FM-LLM ↔ LLM*: information produced by a foundation model is propagated to and interpreted by other language agent. (c) *FM-LLM ↔ LLM-FM*: foundation models that do not share a common interface exchange information through their language interfaces, so that models operating on different native modalities can collaborate on the same task. Within an individual *EywaAgent*, the interaction is asymmetrically orchestrated but not one-way, forming a closed request-observation-reasoning loop. Further, with *EywaMAS* enables domain-specific foundation models to "talk" and collaborate with each other at system level.

**Positioning of the three instantiations.** *EywaAgent*, *EywaMAS*, and *EywaOrchestra* form a hierarchy of system construction and adaptivity, rather than a guarantee of monotonically increasing utility for each task. *EywaAgent* is the basic FM-LLM unit; *EywaMAS* composes such units within a multi-agent system configured manually; and *EywaOrchestra* automates this construction by selecting the foundation models and the topology for each task, yet achieving comparable performance with *EywaMAS*.

**When to use *EywaOrchestra*.** The key difference between *EywaMAS* and *EywaOrchestra* is whether a reliable configuration can be specified in advance, or trials and errors are allowed for testing configurations. Hence, *EywaMAS* is preferable for small-scale distribution where a carefully designed

topology is available. *EywaOrchestra* is preferable when tasks are heterogeneous or evolving, when the best foundation model and topology vary across instances, or when designing and maintaining MAS for every task family is impractical.

**Tsaheylu versus general tool-calling interfaces.** General tool-calling interfaces typically treat an external component as a fixed, stateless function. Tsaheylu is instead designed for foundation models that are not language tools themselves. Through the handshake, the language model obtains the specification of a foundation model, including its supported modalities, tasks, and input and output spaces, and decides whether and how to invoke it. The query compiler then configures the foundation model conditioned on the task state, while the domain-specific data are processed in their native modality rather than being serialized into text. Finally, the response adapter converts the output into a representation that informs subsequent reasoning and multi-agent coordination.

**One foundation model per *EywaAgent*.** Definition 1 binds each *EywaAgent* to a single foundation model, although an agent could in principle connect to multiple foundation models. This is a design choice for context isolation: using a foundation model involves its specification, invocation examples, outputs, and failure history, all of which consume the context of the language model. Binding one foundation model to each agent prevents such context from accumulating in a single agent and limits error propagation across foundation models. Tasks that require multiple foundation models are handled at the system level, where *EywaMAS* and *EywaOrchestra* compose several *EywaAgents*, each responsible for one specialized subtask and communicating only the necessary intermediate results.

# C EywaBench Details

## C.1 Source Datasets

*EywaBench* currently consists of 4 datasets: DeepPrinciple (Song et al., 2025), MMLU-Pro (Wang et al., 2024b), fev-bench (Shchur et al., 2025), and TabArena (Erickson et al., 2025).

**DeepPrinciple: scientific QA dataset.** A scenario-grounded benchmark for authentic scientific discovery across four core disciplines (biology, chemistry, materials science, and physics). It advances beyond decontextualized QA by introducing a large-scale, two-phase evaluation that requires agents to perform iterative reasoning, hypothesis generation, and experimental design in highly diverse scientific contexts.

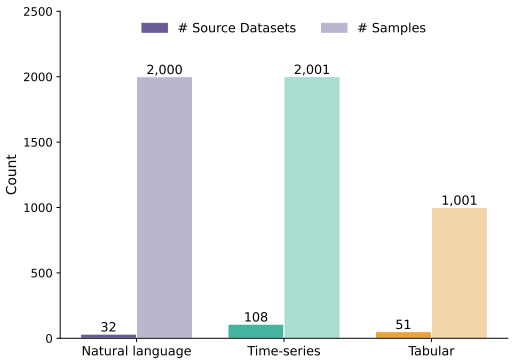
**MMLU-Pro: multi-task language understanding benchmark.** A substantially enhanced reasoning benchmark comprising over 12,000 carefully curated questions across 14 diverse domains. We select scientific domains and convert the original multiple-choice format into open-ended question answering, requiring models to produce direct answers.

**fev-bench: realistic large-scale time series dataset.** A comprehensive time series forecasting benchmark containing 100 rigorous time series. Many of the time series incorporate complex, realistic covariates, which further highlight their high diversity.

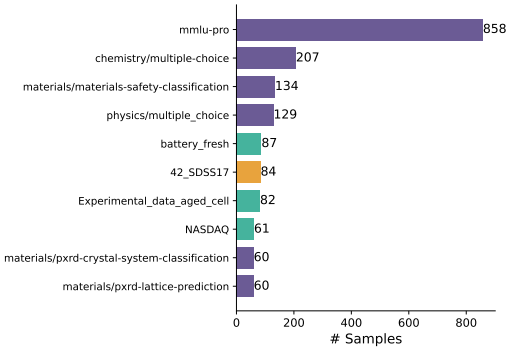
**TabArena: large-scale tabular dataset.** A curated collection of 51 tabular classification and regression datasets covering a wide array of real-world use cases. Supported by a massive evaluation scale of approximately 25 million trained model instances, it offers immense diversity in feature spaces and data distributions for tabular machine learning.

**Scalability and Sampling.** *EywaBench* is designed to be scalable. Its source tasks are drawn from large and diverse scientific benchmarks: DeepPrinciple contains 1,125 questions spanning chemistry, materials science, biology, and physics; MMLU-Pro contains 6,978 questions across physics, chemistry, engineering, economics, health, business, biology, and computer science; FEV-Bench includes 96 time-series datasets, each with up to 30,000 covariate series; and the tabular benchmark contains 51 tables, each with up to 150,000 rows. Moreover, for time-series and tabular tasks, additional task instances can be constructed by sampling different sub-series, covariate groups, rows, columns, and prediction targets from the original datasets. This makes the potential scale of *EywaBench* substantially larger than the number of original datasets alone.

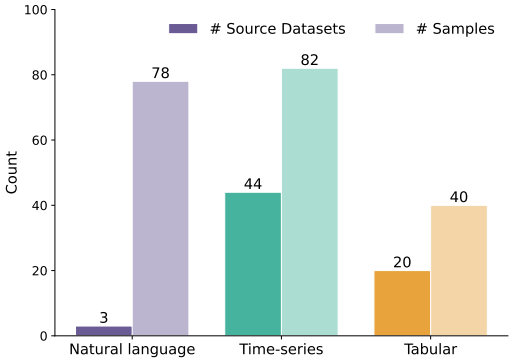
***EywaBench-full.*** In this work, we sample 5,002 task instances from the four benchmark sources to form a controlled evaluation subset.



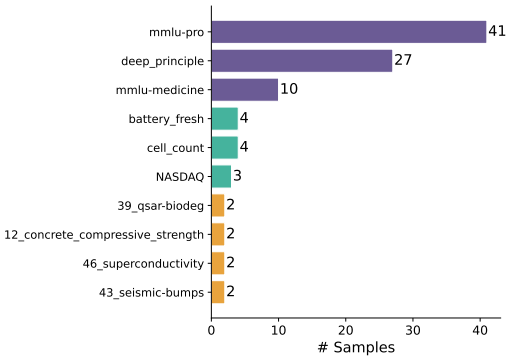

(a) *EywaBench-full*: source-dataset and sample counts by modality.




(b) *EywaBench-full*: top-10 source datasets by sample count.




(c) *EywaBench-mini*: source-dataset and sample counts by modality.




(d) *EywaBench-mini*: top-10 source datasets by sample count.

Figure 7: Source diversity of *EywaBench-full* (top) and *EywaBench-mini* (bottom). (a, c) Both variants cover distinct source datasets and task samples across natural-language, time-series, and tabular modalities, showing broad modality-level coverage. (b, d) The top-10 source datasets follow a long-tailed distribution, with no single source dominating the benchmark. This design reduces the risk that method rankings are driven by overfitting to a single source distribution.

***EywaBench-mini.*** EywaBench-mini is further sampled from EywaBench-full, containing 200 representative samples. This choice is motivated by two considerations. First, unlike standard language-only benchmarks, evaluating heterogeneous agentic systems requires manually configuring and validating *EywaMAS*. Exhaustively evaluating all possible task instances would therefore be prohibitively expensive for human expert inspection. Second, a moderate-sized subset allows us to maintain balanced coverage across domains and data modalities while keeping the evaluation reproducible and comparable across different agentic frameworks.

### C.2 Data Schema

*EywaBench-full* (5,002 instances) and *EywaBench-mini* (200 instances) are each released as a single self-contained Parquet file with Snappy compression. Each row represents one self-contained scientific problem and follows the six-column schema summarized in Table 6; *EywaBench-full* additionally stores provenance metadata for each instance (e.g., source identifier, source collection, and content hash). Every *EywaBench-mini* instance is contained verbatim in *EywaBench-full*. For each source dataset, we also provide parsing scripts that convert its original data format into the unified *EywaBench* schema.

Table 6: Data schema of *EywaBench*. *EywaBench-full* and *EywaBench-mini* are stored as Parquet files that share the same six core fields.

| Column | Type | Description |
|---|---|---|
| `domain` | categorical | Scientific sub-domain, e.g., materials, energy, biology. |
| `task` | string | Task type associated with the sample. |
| `description` | string | Detailed task description of the sample. |
| `output_size` | int64 | Maximum allowed output length for the solver. |
| `input` | string | Prompt, context, or structured input provided to the solver. |
| `label` | string | Ground-truth output or reference answer. |

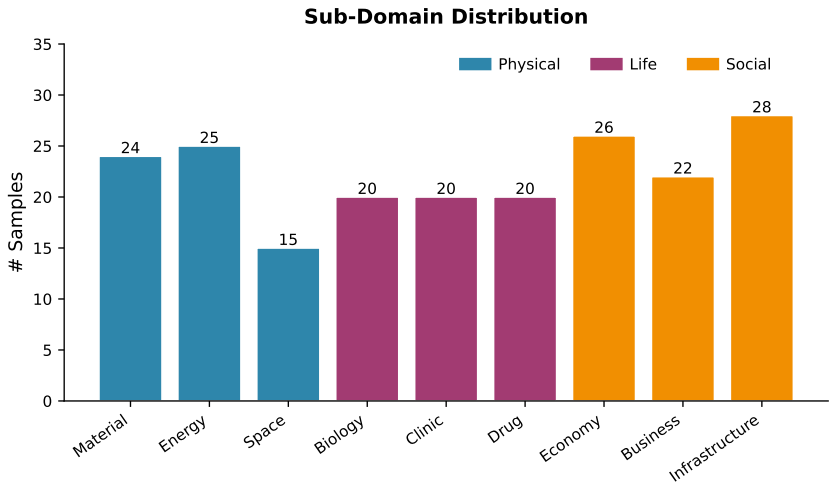



Figure 8: Composition of *EywaBench-mini* along sub-domains. The distribution is near-uniform, avoiding the domain-collapse failure modes.

### C.3 Composition and Coverage Analysis

All statistics in this section are computed directly from the released *EywaBench-full* and *EywaBench-mini* Parquet files.

**Overview.** *EywaBench* is released in two variants: *EywaBench-full* contains $N$=5,002 task instances, and *EywaBench-mini* contains $N$=200 task instances drawn from *EywaBench-full*. The underlying construction pipeline is fully parametric and can be scaled along two independent axes: (i) *sample volume*, where additional instances can be generated from the same source datasets by resampling temporal windows, tabular subsets, covariate groups, or prediction targets without changing the benchmark schema; and (ii) *domain and modality coverage*, where new scientific domains and data modalities can be incorporated through additional parsers. Therefore, both variants should be viewed as *representative slices* of a much larger and extensible benchmark space.

**Balanced taxonomy along three orthogonal axes.** Both variants are organised as a three-tier taxonomy (parent domain $\rightarrow$ sub-domain $\rightarrow$ source dataset) and are simultaneously stratified along the modality axis. We quantify balance with the normalised Shannon entropy $H_n$, where $H_n$=1 denotes a perfectly uniform distribution.

*EywaBench-full.* The physical-, life-, and social-science parent domains are covered at $40.0\%/30.0\%/30.0\%$; material science contains 1,000 instances and each of the other eight sub-domains contains 500–501 instances; and the natural-language, time-series, and tabular modalities are represented at $40.0\%/40.0\%/20.0\%$. The normalised entropy is $H_n$=0.991 at the parent level, $H_n$=0.985 at the sub-domain level, and $H_n$=0.960 at the modality level. The detailed composition is reported in Table 7.




(a) *EywaBench-full*. (b) *EywaBench-mini*.

Figure 9: Hierarchical view of *EywaBench-full* and *EywaBench-mini*: nine sub-domains (inner ring) $\times$ three modalities (outer ring). All 27 cells are populated in both variants; the largest sub-domain accounts for $20.0\%$ of *EywaBench-full* and $14.0\%$ of *EywaBench-mini*.

*EywaBench-mini.* The physical-, life-, and social-science parent domains are covered at $32.0\%/30.0\%/38.0\%$; the nine sub-domains each contain 15–28 instances; and the natural-language, time-series, and tabular modalities are represented at $39.0\%/41.0\%/20.0\%$. The normalised entropy is $H_n$=0.995 at the parent level, $H_n$=0.993 at the sub-domain level, and $H_n$=0.960 at the modality level. The detailed composition is reported in Table 8.

In both variants, no single domain or modality dominates the benchmark, and every one of the nine sub-domains contains instances of all three modalities, yielding $100\%$ cross-modal coverage of the taxonomy ($27/27$ populated cells). This ensures that conclusions about modality-specific agent behaviour drawn from *EywaBench* generalise across scientific fields rather than conflating modality effects with domain effects. We visualize the sub-domain distribution of *EywaBench-mini* in Figure 8 and the sub-domain$\times$modality hierarchy of both variants in Figure 9.

Table 7: Detailed composition of *EywaBench-full* by parent domain, sub-domain, modality, and source dataset. For each sub-domain, we report the total number of samples, the number of natural-language, time-series, and tabular samples, and the number of distinct source datasets. The total number of unique source datasets is 191, which is smaller than the sum of per-sub-domain source counts because a source dataset may contribute samples to multiple sub-domains.

| Parent | Sub-domain | # samples | Natural Language | Time Series | Tabular | # Source Datasets |
|---|---|---|---|---|---|---|
| Physical | Material | 1,000 | 647 | 169 | 184 | 26 |
| | Energy | 501 | 100 | 339 | 62 | 38 |
| | Space | 500 | 128 | 288 | 84 | 14 |
| Life | Biology | 500 | 291 | 48 | 161 | 14 |
| | Clinic | 501 | 191 | 148 | 162 | 12 |
| | Drug | 500 | 358 | 6 | 136 | 20 |
| Social | Economy | 500 | 57 | 378 | 65 | 22 |
| | Business | 500 | 121 | 293 | 86 | 44 |
| | Infrastructure | 500 | 107 | 332 | 61 | 17 |
| **Total** | | **5,002** | **2,000** | **2,001** | **1,001** | **191** |

**Diversity of source datasets.** At the leaf level of the taxonomy, *EywaBench-full* draws samples from 191 distinct source datasets (70 physical-science, 44 life-science, and 81 social-science sources), and *EywaBench-mini* draws samples from 67 of them (21 physical-science, 19 life-science, and 28 social-science sources). These sources cover a broad range of scientific data, including ETT, ERCOT,

Table 8: Detailed composition of *EywaBench-mini* by parent domain, sub-domain, modality, and source dataset. For each sub-domain, we report the total number of samples, the number of natural-language, time-series, and tabular samples, and the number of distinct source datasets. The total number of unique source datasets is 67, which is smaller than the sum of per-sub-domain source counts because each source dataset may contribute multiple samples.

| Parent | Sub-domain | # samples | Natural Language | Time Series | Tabular | # Source Datasets |
|---|---|---|---|---|---|---|
| Physical | Material | 24 | 14 | 4 | 6 | 5 |
| | Energy | 25 | 5 | 16 | 4 | 11 |
| | Space | 15 | 8 | 5 | 2 | 7 |
| Life | Biology | 20 | 10 | 4 | 6 | 5 |
| | Clinic | 20 | 10 | 4 | 6 | 6 |
| | Drug | 20 | 10 | 6 | 4 | 9 |
| Social | Economy | 26 | 5 | 17 | 4 | 11 |
| | Business | 22 | 8 | 10 | 4 | 8 |
| | Infrastructure | 28 | 8 | 16 | 4 | 11 |
| **Total** | | **200** | **78** | **82** | **40** | **67** |

NASDAQ, Jena Weather, LOOP-SEATTLE, Concrete Compressive Strength, and Superconductivity, among others. The source distributions have normalized Shannon entropies of $H_n$=0.850 for *EywaBench-full* and $H_n$=0.846 for *EywaBench-mini*, indicating high diversity with a long-tailed structure, as visualized in Figure 7. Importantly, even the largest individual source, MMLU-Pro, contributes only 858 samples (17.2%) to *EywaBench-full* and 41 samples (20.5%) to *EywaBench-mini*. This reduces the risk that method rankings are dominated by any single source distribution.

### C.4 Representativeness of EywaBench-mini

*EywaBench-mini* serves as a cost-efficient proxy for *EywaBench-full*. We examine its representativeness from two perspectives.

**Coverage.** Every *EywaBench-mini* instance is contained in *EywaBench-full*, and all 67 source datasets of *EywaBench-mini* are among the 191 source datasets of *EywaBench-full*. *EywaBench-mini* populates all 27 sub-domain×modality cells, and its modality distribution closely matches that of *EywaBench-full* (total variation distance 0.01). The largest difference lies at the sub-domain level (total variation distance 0.105), since *EywaBench-mini* allocates samples more evenly across sub-domains, whereas material science accounts for 20.0% of *EywaBench-full*.

Table 9: Utility of the single-agent baseline and *EywaOrchestra* on *EywaBench-mini* and *EywaBench-full* with gpt-5-nano as the backbone. The last column reports the relative difference of *EywaBench-full* with respect to *EywaBench-mini*.

| Method | EywaBench-mini | EywaBench-full | Relative Difference |
|---|---|---|---|
| Single-LLM-Agent | 0.6154 | 0.6165 | +0.2% |
| *EywaOrchestra* (Ours) | 0.6746 | 0.6727 | −0.3% |
| Relative improvement | +9.6% | +9.1% | – |

**Performance.** Table 9 compares the single-agent baseline and *EywaOrchestra* on both variants. The utility of each method differs by less than 0.3% between *EywaBench-mini* and *EywaBench-full*, and the relative improvement of *EywaOrchestra* over the single-agent baseline is preserved (+9.6% on *EywaBench-mini* versus +9.1% on *EywaBench-full*). Therefore, conclusions drawn on *EywaBench-mini* are consistent with those on *EywaBench-full*, and *EywaBench-mini* provides an accurate and efficient alternative for experiments that require manual configuration of *EywaMAS* or incur high inference cost.

### C.5 Metrics

Each task instance $i$ produces a per-instance *utility* $u_i \in [0,1]$ ($u_i$=1 for a perfect prediction). Because *EywaBench* mixes three output modalities, $u_i$ is defined modality-specifically while always

being bounded in $[0, 1]$ so that scores are directly comparable across modalities. Slice-level numbers reported in the paper are the unweighted mean $\bar{u}(\mathcal{D}) = \frac{1}{|\mathcal{D}|}\sum_{i\in\mathcal{D}} u_i$ (with sample standard deviation as dispersion); the same aggregation is applied to runtime and the four token-cost components.

**Natural language.** For DeepPrinciple and the open-ended MMLU-Pro variant, the raw output is first reduced to a final answer $\hat{y}_i$ and normalised by an operator $\pi$ that trims whitespace/quotes, collapses inner whitespace, and maps a few Unicode variants to ASCII. We then apply a three-stage soft-score cascade: *(i) Exact match.* If $\pi(\hat{y}_i) = \pi(y_i)$, set $u_i$=1. *(ii) Numeric relative error.* If both sides parse as a single float $\hat{p}, g$, set

$$u_i = \exp(-e_{\text{rel}}), \qquad e_{\text{rel}} = \frac{|\hat{p} - g|}{\max(|g|,\ 10^{-12})}. \tag{40}$$

*(iii) Lexical fallback.* Otherwise tokenise both strings with a regex that preserves LaTeX commands, words, numbers, and individual symbols; let $o$ be the multiset overlap and define token precision/recall/F1 as $P_{\text{tok}}$=$o/|T_{\text{pred}}|$, $R_{\text{tok}}$=$o/|T_{\text{gold}}|$, $F_1^{\text{tok}}$=$2P_{\text{tok}}R_{\text{tok}}/(P_{\text{tok}}$+$R_{\text{tok}})$ (with $F_1^{\text{tok}}$=1 if both empty and 0 if exactly one is empty). Let $S_{\text{char}}$ be the `difflib.SequenceMatcher` ratio on the normalised strings. Then

$$u_i = \min\big(\tau,\ \alpha\, F_1^{\text{tok}} + \beta\, S_{\text{char}}\big), \qquad \alpha{=}0.6,\ \beta{=}0.4,\ \tau{=}0.8. \tag{41}$$

The cap $\tau$ keeps lexical near-misses strictly below the scores reserved for stages (i)–(ii).

**Time series.** The gold and predicted continuations $\mathbf{y}, \hat{\mathbf{y}} \in \mathbb{R}^H$ are parsed from `(timestamp,value)` CSVs. With a denominator floor $\varepsilon$=$10^{-2}$ and $\mathcal{I}$=$\{t : |y_t|$>$\varepsilon\}$,

$$\begin{aligned} \text{sMAPE} &= \frac{1}{H}\sum_{t=1}^{H} \frac{2|y_t - \hat{y}_t|}{\max(|y_t| + |\hat{y}_t|, \varepsilon)} \in [0, 2], \\ \text{MAAPE} &= \frac{1}{|\mathcal{I}|}\sum_{t\in\mathcal{I}} \arctan\frac{|y_t - \hat{y}_t|}{\max(|y_t|, \varepsilon)} \in [0, \pi/2], \end{aligned} \tag{42}$$

(MAAPE=0 when $\mathcal{I}$=$\emptyset$). The two errors are normalised to $[0, 1]$ and combined into

$$u_i = 1 - \frac{1}{2}\left(\frac{\text{sMAPE}}{2} + \frac{\text{MAAPE}}{\pi/2}\right) \in [0, 1]. \tag{43}$$

Combining the two mitigates the pathologies of either metric used alone: sMAPE saturates symmetrically but is brittle near zero, whereas MAAPE is well-behaved near zero but less scale-sensitive.

**Tabular.** The label and prediction strings are parsed into 1-D arrays via `DataFrame.values.flatten()` (or `eval` when the string starts with `[`). For classification tasks the utility is top-1 accuracy,

$$u_i = \frac{1}{N}\sum_{n=1}^{N} \mathbf{1}\big[\hat{y}_n = y_n\big], \tag{44}$$

implemented via `sklearn.metrics.accuracy_score`. For regression tasks the predicted and gold target columns are scored with the same sMAPE+MAAPE combination as time series (Eqs. (42)–(43)), treating the $N$ row-wise predictions as an $H$-step forecast. Sharing this rule across modalities keeps numeric-target errors on a common scale, so the cross-modal averages reported in the main paper are directly comparable.

# D Experiment Details

## D.1 More Detailed Ablations

In this section, we provide a more comprehensive backbone ablation that extends the single-agent study in Table 2 to also cover *EywaMAS* and *EywaOrchestra*, and reports per-sub-domain results across all nine domains. The full results are shown in Table 10. We highlight three observations.

**Algorithm 1** EywaOrchestra: Dynamic Orchestration of Heterogeneous Experts

**Require:** Task input $\tau = (q, x, y^\star, \ell)$, configuration space $\mathcal{C}$, conductor $P$
**Ensure:** Output $\hat{y}$
1: Select a system configuration $c \leftarrow P(q, x)$
2: Instantiate the heterogeneous agent system specified by $c$
3: Execute the induced multi-agent system on $(q, x)$ and return output $\hat{y}$




Figure 10: ***Eywa* extends current agentic systems. (Left)** Overall comparisons show that *EywaAgent*, *EywaMAS*, and *EywaOrchestra* achieve higher utility with lower token consumption than language-only baselines. **(Right)** Category-level results further show consistent gains in utility, token efficiency, and execution time across physical, life, and social science tasks.

Table 10: Overall performance comparison across scientific domains on *EywaBench*. We compare all methods on three dimensions, including utility (↑), inference time (↓), and token consumption (↓). Best results are highlighted in bold and second-best results are underlined. Our proposed methods, EywaAgent, EywaMAS, and EywaOrchestra, achieve strong overall performance while maintaining competitive efficiency.

| Method | Metrics | Physical Science | | | Life Science | | | Social Science | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Material | Energy | Space | Biology | Clinic | Drug | Economy | Business | Infrastructure | Overall |
| ***Single-Agent Setting*** | | | | | | | | | | | |
| **EywaAgent (gpt-4.1-nano)** | Utility (↑) | 0.5491 | 0.7980 | 0.5848 | 0.3429 | 0.4010 | 0.4592 | 0.7416 | 0.5560 | 0.5762 | 0.5680 |
| | Time (↓) | 9.47 | 20.88 | 10.85 | 31.11 | 23.37 | 31.51 | 23.66 | 13.88 | 12.98 | 19.61 |
| | Tokens (↓) | 1654 | 1149 | 1047 | 1337 | 606 | 839 | 1181 | 1036 | 1237 | 1139 |
| **EywaAgent (gpt-5-nano)** | Utility (↑) | 0.5871 | 0.8390 | 0.6123 | 0.3718 | 0.5085 | 0.6199 | **0.8048** | 0.7371 | 0.7060 | 0.6558 |
| | Time (↓) | 34.88 | 24.42 | 23.12 | 30.84 | 20.32 | 15.84 | 19.71 | 20.98 | 15.99 | 22.78 |
| | Tokens (↓) | 5040 | 3167 | 3329 | 4858 | 2333 | 2210 | 2791 | 2444 | 2248 | 3137 |
| **EywaAgent (gpt-5-mini)** | Utility (↑) | 0.6272 | 0.8615 | 0.6286 | 0.3670 | 0.4990 | **0.6444** | 0.7790 | 0.7193 | 0.7320 | 0.6640 |
| | Time (↓) | 21.81 | 20.70 | 29.52 | 45.34 | 16.86 | 29.91 | 18.97 | 21.08 | 16.05 | 23.63 |
| | Tokens (↓) | 3666 | 2316 | 2541 | 3589 | 1734 | 2527 | 2033 | 1669 | 2092 | 2444 |
| ***Multi-Agent Setting*** | | | | | | | | | | | |
| **EywaMAS (gpt-4.1-nano)** | Utility (↑) | 0.6161 | 0.8305 | 0.6602 | 0.3545 | 0.4468 | 0.5062 | 0.7729 | 0.6404 | 0.6759 | 0.6236 |
| | Time (↓) | 31.16 | 39.72 | 27.96 | 42.28 | 36.96 | 31.72 | 38.16 | 63.88 | 39.24 | 39.40 |
| | Tokens (↓) | 6592 | 4782 | 3832 | 5284 | 2392 | 3049 | 4328 | 3764 | 4969 | 4421 |
| **EywaMAS (gpt-5-nano)** | Utility (↑) | **0.6381** | **0.8742** | 0.6899 | **0.3798** | 0.5086 | 0.6248 | 0.7959 | 0.7284 | **0.7406** | **0.6761** |
| | Time (↓) | 77.25 | 75.96 | 72.51 | 111.92 | 59.97 | 59.23 | 68.40 | 58.11 | 46.49 | 72.11 |
| | Tokens (↓) | 14529 | 11709 | 11787 | 16502 | 9407 | 8078 | 11044 | 9470 | 8912 | 11214 |
| ***Dynamic Orchestration*** | | | | | | | | | | | |
| **EywaOrchestra (gpt-4.1-nano)** | Utility (↑) | 0.6146 | 0.8259 | 0.5498 | 0.3355 | 0.4567 | 0.6153 | 0.7442 | 0.6725 | 0.6519 | 0.6210 |
| | Time (↓) | 35.14 | 23.13 | 24.32 | 32.51 | 20.30 | 17.83 | 21.77 | 15.93 | 18.00 | 23.05 |
| | Tokens (↓) | 8272 | 5808 | 5846 | 8517 | 4847 | 4912 | 6088 | 4627 | 5317 | 6017 |
| **EywaOrchestra (gpt-5-nano)** | Utility (↑) | 0.6249 | 0.8711 | **0.7187** | 0.3682 | **0.5159** | 0.6319 | 0.7830 | **0.7388** | 0.7298 | 0.6746 |
| | Time (↓) | 61.78 | 39.92 | 75.47 | 67.88 | 45.38 | 45.94 | 49.13 | 34.18 | 28.80 | 48.16 |
| | Tokens (↓) | 11535 | 7723 | 10810 | 11315 | 7050 | 6495 | 7117 | 7264 | 6892 | 8335 |

Table 11: Evaluation of utility standard deviation across scientific domains on *EywaBench*, using `gpt-5-nano` as the backbone LLM for all methods. Lower values indicate more stable performance across tasks within each domain.

| Method | Physical Science | | | Life Science | | | Social Science | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Material | Energy | Space | Biology | Clinic | Drug | Economy | Business | Infrastructure | Overall |
| **Single-LLM-Agent** | 0.3754 | 0.2355 | 0.3573 | 0.2005 | 0.3389 | 0.3019 | 0.2548 | 0.2959 | 0.3377 | 0.3293 |
| **EywaAgent (Ours)** | 0.3880 | 0.2436 | 0.3861 | 0.1887 | 0.3357 | 0.3038 | 0.2253 | 0.2360 | 0.3306 | 0.3232 |
| **Refine MAS** (Madaan et al., 2023) | 0.3373 | 0.1815 | 0.3072 | 0.1599 | 0.3313 | 0.3132 | 0.2807 | 0.2697 | 0.3697 | 0.3201 |
| **Debate MAS** (Du et al., 2024) | 0.3770 | 0.1664 | 0.2785 | 0.1655 | 0.3062 | 0.3408 | 0.2783 | 0.2688 | 0.3233 | 0.3190 |
| **MoA** (Wang et al., 2025b) | 0.3385 | 0.2536 | 0.3559 | 0.1874 | 0.3334 | 0.3259 | 0.2470 | 0.2319 | 0.3624 | 0.3205 |
| **X-MAS** (Ye et al., 2025) | 0.3632 | 0.2658 | 0.3421 | 0.1823 | 0.3365 | 0.2963 | 0.3257 | 0.2747 | 0.3171 | 0.3242 |
| **EywaMAS (Ours)** | 0.3580 | 0.1685 | 0.2929 | 0.1843 | 0.3102 | 0.2975 | 0.2336 | 0.2479 | 0.3120 | 0.3025 |
| **EywaOrchestra (Ours)** | 0.3498 | 0.1768 | 0.2991 | 0.1943 | 0.3246 | 0.3352 | 0.2445 | 0.2586 | 0.3168 | 0.3109 |

Table 12: Evaluation of mean completion token consumption across scientific domains on *EywaBench*, using `gpt-5-nano` as the backbone LLM for all methods. Lower values indicate more efficient generation.

| Method | Physical Science | | | Life Science | | | Social Science | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Material | Energy | Space | Biology | Clinic | Drug | Economy | Business | Infrastructure | Overall |
| **Single-LLM-Agent** | 5017 | 4094 | 3882 | 5156 | 3216 | 3047 | 3301 | 3298 | 2402 | 3670 |
| **EywaAgent (Ours)** | 3474 | 2039 | 2490 | 3603 | 1712 | 1451 | 1632 | 1553 | 981 | 2055 |
| **Refine MAS** (Madaan et al., 2023) | 8138 | 7274 | 8623 | 8321 | 6068 | 6315 | 7199 | 5604 | 5413 | 6912 |
| **Debate MAS** (Du et al., 2024) | 12084 | 11447 | 11360 | 13562 | 9607 | 8552 | 11888 | 8692 | 7082 | 10391 |
| **MoA** (Wang et al., 2025b) | 19645 | 13058 | 14315 | 11660 | 9252 | 8314 | 12515 | 9024 | 8860 | 11908 |
| **X-MAS** (Ye et al., 2025) | 18223 | 16159 | 13397 | 14038 | 10679 | 9646 | 12931 | 11351 | 10045 | 12966 |
| **EywaMAS (Ours)** | 9612 | 8230 | 9156 | 12556 | 7515 | 5679 | 7504 | 6722 | 5042 | 7865 |
| **EywaOrchestra (Ours)** | 8627 | 5576 | 8726 | 8917 | 5369 | 4528 | 4796 | 5196 | 4262 | 6060 |

Table 13: Evaluation of mean reasoning token consumption across scientific domains on *EywaBench*, using `gpt-5-nano` as the backbone LLM for all methods. Lower values indicate more efficient reasoning.

| Method | Physical Science | | | Life Science | | | Social Science | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Material | Energy | Space | Biology | Clinic | Drug | Economy | Business | Infrastructure | Overall |
| **Single-LLM-Agent** | 4891 | 3940 | 3763 | 5036 | 3153 | 2966 | 3166 | 3197 | 2281 | 3554 |
| **EywaAgent (Ours)** | 3347 | 1892 | 2372 | 3482 | 1654 | 1363 | 1509 | 1457 | 859 | 1943 |
| **Refine MAS** (Madaan et al., 2023) | 7912 | 6968 | 8431 | 8083 | 5946 | 6157 | 6912 | 5402 | 5170 | 6687 |
| **Debate MAS** (Du et al., 2024) | 11677 | 10998 | 11085 | 13216 | 9418 | 8320 | 11459 | 8378 | 6720 | 10048 |
| **MoA** (Wang et al., 2025b) | 18812 | 12439 | 13535 | 10980 | 9072 | 8054 | 11788 | 8596 | 8294 | 11335 |
| **X-MAS** (Ye et al., 2025) | 17167 | 15431 | 12624 | 13458 | 10475 | 9373 | 12312 | 10880 | 9503 | 12378 |
| **EywaMAS (Ours)** | 9227 | 7808 | 8892 | 12202 | 7341 | 5430 | 7111 | 6417 | 4679 | 7533 |
| **EywaOrchestra (Ours)** | 8341 | 5325 | 8469 | 8653 | 5222 | 4336 | 4544 | 4989 | 4027 | 5826 |

***Eywa* is consistently compatible and effective across LLM backbones.** Across all three backbones (gpt-4.1-nano, gpt-5-nano, and gpt-5-mini) and all three system settings (single-agent, multi-agent, and dynamic orchestration), *Eywa* delivers strong utility on every sub-domain. For instance, with the relatively weak gpt-4.1-nano backbone, *EywaAgent*, *EywaMAS*, and *EywaOrchestra* reach overall utilities of 0.5680, 0.6236, and 0.6210, respectively, which already surpass the single-agent gpt-5-nano baseline reported in Table 1. A similar pattern holds when we move to gpt-5-nano, where the relative ranking among single-agent, multi-agent, and dynamic-orchestration variants is preserved. This stability suggests that the gains of *Eywa* come primarily from cross-modality heterogeneity and structured FM-LLM collaboration, rather than from a particular language-model checkpoint.

***Eywa* benefits from stronger LLM backbones.** As we scale the language model from gpt-4.1-nano to gpt-5-nano, all three settings improve substantially in overall utility: *EywaAgent* improves from

0.5680 to 0.6558 (+15.5%), *EywaMAS* from 0.6236 to 0.6761 (+8.4%), and *EywaOrchestra* from 0.6210 to 0.6746 (+8.6%). The per-domain breakdown shows consistent improvements across physical, life, and social science. This indicates that, although *Eywa* already provides strong baseline performance through its FM–LLM coupling, the gains compound as the underlying language model becomes more capable: stronger LLMs produce more reliable planning, better routing, and more faithful integration of foundation-model outputs into the final answer.

**Diminishing returns from gpt-5-nano to gpt-5-mini suggest a domain-specific bottleneck.** While moving from gpt-4.1-nano to gpt-5-nano yields large utility gains (e.g., +15.5% for *EywaAgent*), further upgrading the backbone from gpt-5-nano to gpt-5-mini brings only marginal improvements (0.6558 → 0.6640, i.e., +1.25% for *EywaAgent*), and on several sub-domains the larger model is in fact slightly worse (e.g., Biology, Clinic, Economy, and Business). Interestingly, on all four of these sub-domains, gpt-5-mini consumes 26%–32% fewer tokens than gpt-5-nano, whereas the average reduction is only 14% on the five sub-domains where it improves (Table 10). One possible explanation is that the stronger model is more confident in its initial language-based judgment and therefore engages less deeply with the domain-specific data and the foundation-model outputs. This pattern indicates that once the LLM is sufficiently capable of planning, communicating, and routing to the specialized foundation models, additional general-purpose LLM scaling yields diminishing returns. In other words, the remaining headroom in utility is no longer dominated by the LLM's general reasoning ability, but rather by *domain-specific* capability. This observation provides a strong motivation for our *Eywa* to extend agentic systems with domain-specific components.

***Eywa* generalizes across model families, including open-weight models.** Beyond the GPT family, we further evaluate *Eywa* on *EywaBench-full* with backbones from three additional model families, including two open-weight models that can be self-deployed: claude-haiku-4.5 (Anthropic, 2025), gemini-2.5-flash (Comanici et al., 2025), qwen-2.5-7b (Qwen Team, 2024), and gemma-4-31b (Gemma Team, 2026). As shown in Table 14, *EywaOrchestra* consistently improves over the single-agent baseline with the same backbone, with relative gains ranging from +6.6% to +11.3%. The improvements on the two open-weight models show that the benefit of *Eywa* is neither specific to one model family nor dependent on proprietary models, which is particularly relevant for cost-sensitive settings where self-deployed models are preferred. Notably, the largest gain is obtained with the smallest backbone, qwen-2.5-7b (+11.3%), suggesting that domain-specific foundation models are especially helpful when the language model itself is less capable.

Table 14: Utility of the single-agent baseline and *EywaOrchestra* with backbones from different model families on *EywaBench-full*. Relative gain is computed with respect to the single-agent baseline with the same backbone.

| Backbone | Weights | Single-LLM-Agent | *EywaOrchestra* (Ours) | Relative gain |
|---|---|---|---|---|
| gpt-5-nano | Proprietary | 0.6165 | **0.6727** | +9.1% |
| claude-haiku-4.5 | Proprietary | 0.5881 | **0.6388** | +8.6% |
| gemini-2.5-flash | Proprietary | 0.5972 | **0.6517** | +9.1% |
| qwen-2.5-7b | Open | 0.5155 | **0.5739** | +11.3% |
| gemma-4-31b | Open | 0.5631 | **0.6002** | +6.6% |

**Experiments with frontier LLMs.** We further compare *EywaOrchestra* with gpt-5-nano against Inkling-975B (Thinking Machines Lab, 2026), a frontier-scale language model. Due to the evaluation cost, we evaluate on *EywaBench-mini*. As shown in Table 15, gpt-5-nano equipped with domain-specific foundation models outperforms the single-agent Inkling overall, while running 3.8× faster. Moreover, using Inkling-975B as the backbone of *EywaOrchestra* further improves its utility, showing that *Eywa* remains beneficial even with a substantially stronger language model.

Table 15: Comparison between the frontier model Inkling-975B and *EywaOrchestra* on *EywaBench-mini*. Time is the average wall-clock execution time per instance.

| Backbone | Method | Physical | Life | Social | Overall | Time (s, ↓) |
|---|---|---|---|---|---|---|
| Inkling-975B | Single-LLM-Agent | 0.7378 | 0.5042 | 0.6822 | 0.6466 | 183.41 |
| gpt-5-nano | *EywaOrchestra* (Ours) | 0.7431 | 0.5053 | 0.7506 | 0.6746 | 48.16 |
| Inkling-975B | *EywaOrchestra* (Ours) | 0.7523 | 0.5069 | 0.7581 | 0.6809 | 359.26 |

### D.2 Replacing the Time-Series Foundation Model

To examine how the capability of the domain-specific model affects *Eywa*, we replace the time-series foundation model Chronos in *EywaOrchestra* with two weaker forecasting models while keeping all other components unchanged: N-BEATS (Oreshkin et al., 2020), a deep forecasting model without large-scale pretraining, and ARIMA (Box et al., 2015), a classical statistical model that is also a common choice when agents forecast through code-execution tools.

Table 16: Utility of *EywaOrchestra* on *EywaBench-full* with different time-series models (gpt-5-nano backbone). Relative change is computed with respect to Chronos.

| **Time-series model** | **Type** | **Utility (↑)** |
|---|---|---|
| Chronos | Pretrained foundation model | **0.6727** |
| N-BEATS | Deep forecasting model | 0.6596 |
| ARIMA | Statistical model | 0.6545 |

As shown in Table 16, the utility of *EywaOrchestra* follows the capability of the time-series model: Chronos achieves the highest utility, followed by N-BEATS and ARIMA. Since only the time-series instances (40% of *EywaBench-full*) are affected by this replacement, the differences in overall utility understate the differences on the time-series subset. These results show that stronger pretrained domain-specific models directly translate into better end-to-end performance. This also suggests the potential of *Eywa* growing as scientific foundation models continue to improve.

### D.3 Extending the Foundation Model Pool

To examine whether *Eywa* generalizes beyond the two foundation models in our main experiment, we extend the candidate foundation model pool of *EywaOrchestra* with five additional foundation models: TimesFM 2.5 (Das et al., 2024), Granite-TTM-R3 (Ekambaram et al., 2024), and MOMENT (Goswami et al., 2024) for time series, and Nori (Synthefy et al., 2026) and Mitra (Zhang et al., 2026) for tabular data. Each additional foundation model is exposed through the same Tsaheylu interface as Chronos and TabPFN and registered in the planner's list of available foundation models (Figure 16), while the agents and the orchestration procedure remain unchanged. All other settings follow Table 1, i.e., gpt-5-nano as the backbone on *EywaBench-mini*.

Table 17: *EywaOrchestra* with the default and the extended foundation model pools on *EywaBench-mini* (gpt-5-nano backbone). Time is the average wall-clock execution time per instance.

| **Pool** | **# FMs** | **Time-series FMs** | **Tabular FMs** | **Utility (↑)** | **Time (s, ↓)** |
|---|---|---|---|---|---|
| Default | 2 | Chronos | TabPFN | 0.6746 | 48.16 |
| Extended | 7 | Chronos, TimesFM 2.5, Granite-TTM-R3, MOMENT | TabPFN, Nori, Mitra | **0.6813** | 157.94 |

As shown in Table 17, the extended pool further improves the overall utility of *EywaOrchestra*, indicating that *Eywa* can incorporate additional specialized models and benefit from them to some extent. However, the execution time increases from 48.16 to 157.94 seconds (3.3×). Therefore, the default pool offers a favorable utility-cost trade-off and is used throughout our main experiments, whereas a larger pool is preferable when utility is prioritized over latency.

### D.4 Deployment of Foundation Models

All domain-specific foundation models used in this work, including the extended pool in Appendix D.3, can be deployed on a single machine with four NVIDIA A40 GPUs, occupying approximately 150GB of storage in total. We build a separate virtual environment for each foundation model, so that models with conflicting software dependencies can coexist on the same machine, and each model is served as an independent service through the Tsaheylu interface. Consequently, adding a new foundation model to *Eywa* only requires setting up its own environment and service, without modifying the environments of existing models.

### D.5 Case Study

We provide representative case studies to qualitatively illustrate how different agentic configurations behave on *EywaBench-mini*. In particular, we focus on how each system parses a text-defined task, handles structured non-linguistic inputs, invokes or fails to invoke specialized predictive capabilities, and realizes the final output under strict formatting constraints.

Case Study A compares a language-only LLM agent with *EywaAgent* on the same task over structured financial signals. As shown in Figure 11, the language-only agent correctly understands the surface-level task interface: it identifies the required output schema, produces the correct number of future timestamps, and returns a dataframe-style response. However, because it only reasons over serialized values, its prediction collapses to a last-value persistence baseline. This illustrates a common failure mode: the LLM can follow instructions and satisfy formatting requirements, but this does not imply that it has performed the underlying domain-specific numerical computation.

Figure 12 shows the corresponding *EywaAgent* behavior on the same task. Instead of treating the serialized sequence as a purely textual pattern, *EywaAgent* uses the LLM to parse the task, configure the model call, and activate the Chronos foundation model through the *Tsaheylu* interface. Chronos then serves as the core predictor, while the LLM verifies the returned forecast in context and formats it into the required dataframe-style response. This case highlights the intended division of labor in *EywaAgent*: the language model provides task understanding and interface control, while the specialized foundation model performs the domain-specific computation.

Case Study B further illustrates the role of *EywaOrchestra*. Unlike *EywaAgent*, which uses a fixed augmented-agent design, *EywaOrchestra* first decides which configuration should be used for the given task. As shown in Figure 13, the planner identifies the input as a Miami house-price prediction problem, formulates it as structured tabular regression, and selects a lightweight single-agent *Eywa* configuration with `gpt-5-nano` and TabPFN. This example shows that orchestration does not necessarily mean using a more complex multi-agent topology; rather, it means selecting an appropriate configuration for the task. When a specialized tabular foundation model is sufficient, *EywaOrchestra* can avoid unnecessary multi-agent discussion and directly route the task to the most suitable expert.

Overall, these case studies show three complementary points. First, language-only agents may be highly format-compliant while still relying on shallow heuristics for structured numerical tasks. Second, *EywaAgent* makes specialized foundation models usable inside text-defined workflows by bridging natural-language instructions and model-specific inference interfaces. Third, *EywaOrchestra* extends this idea by dynamically selecting the agent setting, foundation model, and execution plan according to the task modality and complexity. Together, these examples provide qualitative evidence for the central motivation of *Eywa* that heterogeneous scientific reasoning requires both language-side task execution and domain-side specialized computation.

### D.6 Detailed Utility and Token Consumption

To complement the overall utility-vs.-token plot in Figure 4 of the main text, we present the per-sub-domain trade-off in Figure 14, covering all nine sub-domains across physical, life, and social science. Each panel places competing methods in a 2D space defined by token consumption (x-axis) and utility (y-axis), with the upper-left corner being the most desirable. We highlight three observations.

**All three *Eywa* variants sit on or close to the Pareto frontier in every sub-domain.** Across all nine panels, *EywaAgent*, *EywaMAS*, and *EywaOrchestra* are consistently among the points that define the Pareto frontier, while homogeneous and heterogeneous LLM-only multi-agent baselines (*Refine*, *Debate*, *MoA*, *X-MAS*) are typically dominated. For example, on Material, *Debate*, *MoA*, and *X-MAS* all spend $14{,}000$–$25{,}000$ tokens while achieving lower utility than *EywaMAS*; a similar pattern is seen on Energy, Biology, and Drug. This indicates that the *Eywa* family does not trade tokens for utility. Instead, it pushes the Pareto frontier outward by routing computation to specialized foundation models rather than spending it on additional LLM exchanges.

***EywaAgent* is the most token-efficient point on the frontier.** Across the nine sub-domains, *EywaAgent* uses on average $3{,}137$ tokens per task, compared with $4{,}469$ for the single-agent gpt-5-nano baseline and $8{,}673$–$16{,}537$ for the various MAS baselines. As a result, *EywaAgent* typically anchors the lower-left region of the frontier: it delivers utility comparable to or above LLM-only MAS

Case Study A.1: How LLM Agent Fails a Domain-Specific Problem

**Core Task.** This case study presents a representative instance from *EywaBench-mini*, centered on a domain-specific numerical prediction task over structured financial signals. The input consists of 50 normalized NASDAQ measurements indexed by consecutive, anonymized timestamps. Although the data are presented in a text-readable format, solving it requires recognizing numerical dynamics in a structured signal and extrapolating future values. To reduce the possibility that language models rely on memorized historical market trends, all timestamps are anonymized and re-indexed during preprocessing. The goal is to estimate the next 10 normalized NASDAQ values.

**Agents.** We compare a language-only LLM agent with *EywaAgent*. The LLM agent performs the task solely through language-based reasoning with gpt-5-nano backbone. In contrast, *EywaAgent* connects a Chronos time-series foundation model with gpt-5-nano through the *Tsaheylu* interface.

---

**Prompt:**

> You are given a normalized univariate financial time series and are asked to forecast the next 10 values. The timestamps have been anonymized and re-indexed to avoid relying on memorized historical market trends.
>
> The input is a structured financial signal with two columns: `timestamp` and `value`. It contains 50 consecutive normalized NASDAQ observations with anonymized timestamps. Your task is to infer the values for the next 10 timestamps. Return the prediction as a pandas-dataframe-style string with exactly two columns: `timestamp,value`. Do not include explanations, comments, or intermediate reasoning.
>
> **Input time series:**
>
> `timestamp,value`
>
> ...
>
> **Forecast horizon:** 10 timestamps.
>
> **Required output format:** . . .

---

**LLM Agent Response Trajectory:**

1. **Task-interface parsing.** The prompt is interpreted as a domain-specific numerical prediction task with a strict output interface. The input is presented as a dataframe-style sequence, and the answer must be returned in the same dataframe-style format. This creates two immediate constraints: the response should contain exactly 10 future timestamps and only the columns `timestamp` and `value`, without any additional explanation.
2. **Language-only inference.** The agent then inspects the serialized numerical observations. Since it has no access to a specialized forecasting model, it can only infer future values from the surface pattern of the provided sequence. To avoid introducing unsupported oscillations or trends, it chooses a conservative continuation strategy.
3. **Persistence-style output generation.** Following this strategy, the agent extends the timestamp index by 10 steps and assigns each future timestamp the last observed value. The resulting response is format-compliant, but the prediction itself reduces to a last-value persistence forecast rather than a domain-specific numerical prediction.

   `timestamp,value`
   ***(The agent keeps the last observed value as the prediction for all future timestamps)***
   ...

**Evaluation on LLM Agent Output:** The response receives a utility score of 0.605. The agent correctly follows the required output schema and produces the correct number of future predictions. However, because the language-only agent has no specialized numerical forecasting capability, its final answer reduces to a last-value persistence baseline, limiting its predictive utility.

Figure 11: Case study of a language-only LLM agent on a representative *EywaBench-mini* instance. Although the agent correctly parses the task interface and produces a format-compliant response with the required forecast horizon, its prediction collapses to a last-value persistence baseline. This illustrates the limitation of language-only reasoning on structured domain-specific numerical tasks and motivates the need for *EywaAgent* to invoke specialized foundation models through the *Tsaheylu* interface.

baselines while consuming a small fraction of their tokens. This is consistent with our claim that, when the foundation model already encodes the necessary domain prior, an additional debate or refinement loop among LLMs is largely redundant.

Case Study A.2: How EywaAgent Solves a Domain-Specific Problem with Specialized Foundation Model

**Core task and prompts are the same as previous A.1 case study.**

**Agents.** We compare a language-only LLM agent with *EywaAgent*. The LLM agent performs the task solely through language-based reasoning with gpt-5-nano backbone. In contrast, *EywaAgent* connects a Chronos time-series foundation model with gpt-5-nano through the *Tsaheylu* interface.

---

**Why Not Directly Use the Foundation Model?** The Chronos foundation model is the main predictive component in this case, but it cannot independently execute the full text-defined task. It does not parse natural-language instructions, interpret the required dataframe-style output schema, decide which part of the prompt corresponds to the numerical input, or format the answer according to the user's textual constraints. Directly applying Chronos would require manually extracting the numerical sequence, specifying the forecast horizon, running model inference, and post-processing the prediction into the requested output format. After our extension, *EywaAgent* automates this missing interface.

---

**EywaAgent Response Trajectory.**

1. **Task-interface parsing by the LLM.** The `gpt-5-nano` backbone first parses the user prompt and identifies that the task is not a general language reasoning problem, but a domain-specific numerical prediction problem over a structured financial signal. It extracts the input schema, the forecast horizon, and the required output format.
2. **Foundation-model activation through *Tsaheylu*.** Instead of extrapolating directly from the serialized values, *EywaAgent* activates the connected Chronos foundation model through the *Tsaheylu* interface. The interface converts the dataframe-style input into the numerical sequence format expected by Chronos, configures the forecast horizon, and delegates the core prediction step to the specialized foundation model.
3. **Specialized prediction by the foundation model:** Chronos performs the main predictive computation over the structured numerical signal. Unlike the language-only agent, which falls back to a persistence-style heuristic, the foundation model uses its pretrained time-series inductive bias to generate a nontrivial forecast for the next 10 steps.

   `timestamp,value`
   *(Chronos-generated predictions are aligned with the next 10 timestamps)*
   `...`
4. **Context-aware verification and output realization by the LLM.** The Chronos predictions are returned to *EywaAgent*, where the LLM performs a context-aware check beyond simple format validation. It verifies the forecast horizon, timestamp alignment, numerical scale, and consistency with the prompt constraints, making adjustments when needed. The final prediction is then formatted as the required pandas-dataframe-style response. Thus, Chronos serves as the core predictor, while the LLM handles task interpretation, model-call configuration, contextual verification, and final answer realization.

**Evaluation on EywaAgent Output:** The response obtains a higher utility score of $0.701$ because it preserves the required output format while using a specialized foundation model for the core predictive computation, rather than relying on a language-level persistence heuristic.

Figure 12: Case study of *EywaAgent* on the same *EywaBench-mini* instance as Case Study A.1. Unlike the language-only agent, *EywaAgent* enables the Chronos foundation model to serve as the core predictor while using the LLM to parse the text-defined task, configure the model call through the *Tsaheylu* interface, verify the returned prediction in context, and realize the final dataframe-style response. The improved utility score shows the benefit of combining specialized foundation-model computation with language-side task execution.

***EywaMAS* and *EywaOrchestra* occupy complementary points on the frontier.** *EywaMAS* pushes utility further by adding cross-modality multi-agent coordination, attaining the highest utility on most domains, at a moderate token cost that remains below all LLM-only MAS baselines. *EywaOrchestra*, in contrast, dynamically selects the topology and the foundation model on a per-sample basis, which lets it reach utility close to or matching *EywaMAS* while reducing average token consumption from $11{,}214$ to $8{,}335$ ($-26\%$). Moreover, *EywaOrchestra* does not rely on expert configuration of the MAS. This positions *EywaOrchestra* at the upper-left corner of the frontier, providing a favorable point in the trade-off when both quality and budget matter, and confirming that adaptive orchestration is an effective lever for navigating the utility-cost trade-off across heterogeneous scientific tasks.

Case Study B.1: EywaOrchestra Example 1

**Core Task.** This case study presents a representative *EywaBench-mini* instance on Miami house price. The input is a structured real-estate table containing property attributes. The target column contains masked entries, and the system is asked to predict these missing sale prices and return them as a numpy-array-style string.

**Prompt:** Please refer to Appendix E for details.

---

**Planner Orchestration Stage.**

1. **Task diagnosis.** The planner parses the text-defined task and identifies it as a Miami house-price prediction problem, formulated as a structured tabular regression task. It extracts the input modality, target column, masked query rows, output size, and required numpy-array-style response format.
2. **Configuration selection.** Based on the detected tabular structure, the planner decides that a full multi-agent discussion is unnecessary. Instead, it selects a lightweight single-agent *Eywa* configuration with `gpt-5-nano` as the language backbone and TabPFN as the specialized tabular foundation model:

   $$\{\texttt{eywa}:\texttt{true},\ \texttt{setting}:\texttt{single-agent},\ \texttt{model}:\texttt{gpt-5-nano},\ \texttt{fm}:\texttt{TabPFN}\}.$$

3. **Execution plan.** The planner decomposes the problem into three executable steps: extract observed training rows and masked query rows, invoke TabPFN to estimate the missing values, and format the resulting predictions as the required numpy array.

**Execution Stage.**

1. **Structured input extraction.** The execution agent converts the serialized table into structured tabular data, separates rows with observed target values from rows with masked targets, and identifies the features used for prediction.
2. **Foundation-model inference.** The selected tabular foundation model performs the core regression step. It predicts each missing value based on the corresponding row features.
3. **Context-aware verification.** The LLM checks whether the returned predictions match the requested output size, whether the values are plausible under the scale of the observed column, and whether the prediction array respects the order of the masked rows.
4. **Final response realization.** After verification, *EywaOrchestra* returns only the predicted labels in the required numpy-array-style format, satisfying the text-defined output constraint while relying on the tabular foundation model for the core numerical computation.

   `Output:` [*tabular-model prediction for row 1*, *tabular-model prediction for row 2*, ...]

**Evaluation on EywaOrchestra Output:** The response receives a utility score of 0.853. *EywaOrchestra* correctly identifies the task as Miami house-price prediction and selects a single-agent *Eywa* configuration with `gpt-5-nano` and TabPFN. The resulting prediction preserves the required numpy-array-style output format while using TabPFN for the core tabular regression computation.

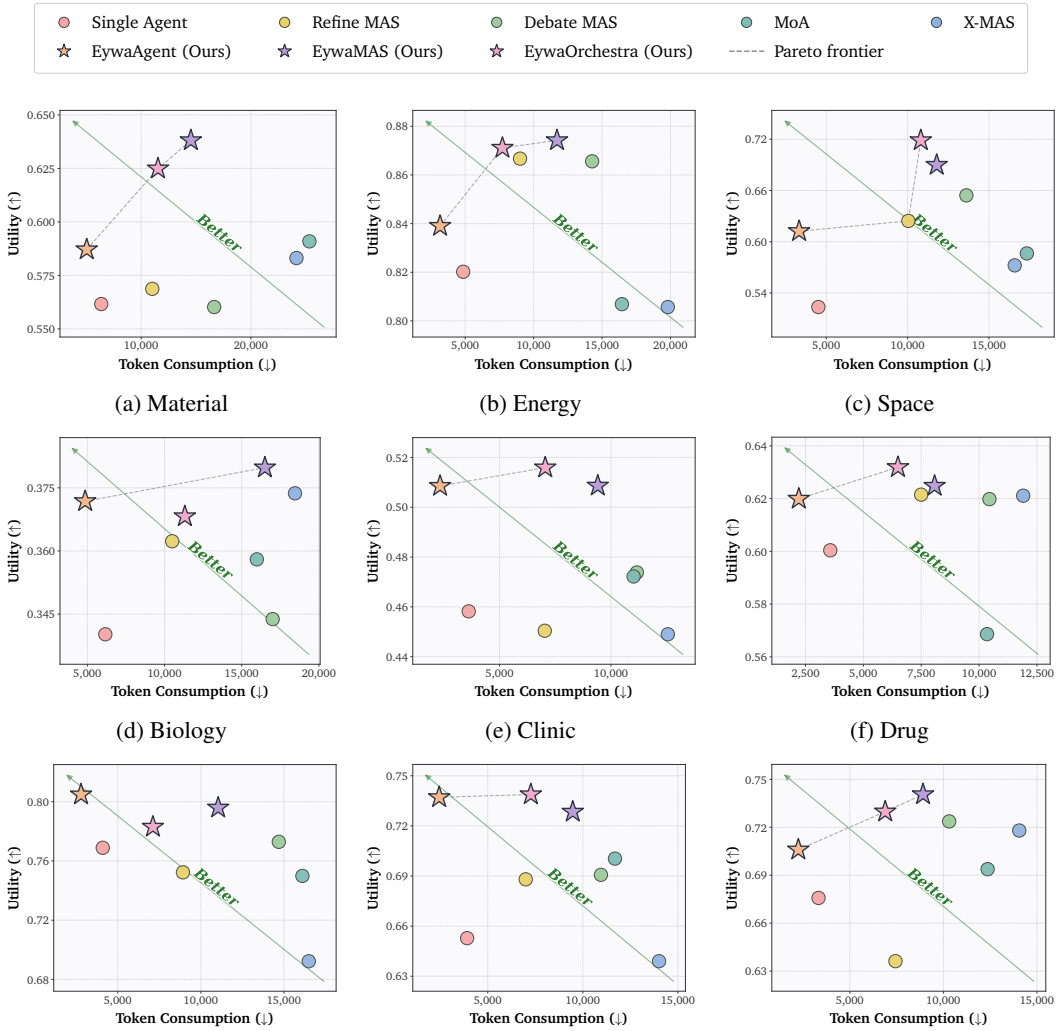
Figure 13: Case study of *EywaOrchestra* on a representative *EywaBench-mini* instance. The planner first diagnoses the task modality, selects a tabular predictive expert, and determines an execution topology. The execution stage then extracts structured inputs, invokes the selected foundation model for the core regression computation, and uses the LLM for contextual verification and final numpy-array-style response realization.

## E Prompt Templates

This section provides the prompt templates used in *EywaBench-mini* and *EywaOrchestra*.

**General task-execution prompt.** For all tasks in *EywaBench-mini*, we adopt a unified task-execution template. The template specifies the task role, optional model/tool context, structured input field, expected output size, and task-specific response format. This design provides a consistent interface across heterogeneous task modalities, while still allowing modality-specific information to be injected through specialized input tags and additional instructions. This prompt is illustrated in Prompt 15.

**Planner prompt for *EywaOrchestra*.** Unlike fixed agent baselines, *EywaOrchestra* first invokes a planner to determine how a task should be executed. The planner receives a task description, domain, and task type, and returns a structured JSON configuration. This configuration specifies whether to use single-agent or multi-agent execution, whether to enable an *Eywa*-augmented agent, which




(a) Material (b) Energy (c) Space

(d) Biology (e) Clinic (f) Drug

(g) Economy (h) Business (i) Infrastructure

Figure 14: Utility vs. token consumption across nine scientific domains on *EywaBench*. In every panel, the **green arrow** indicates the preferred direction of the trade-off, pointing toward higher utility with fewer tokens (i.e., the upper-left of each plot); methods closer to the arrow head are more desirable. The grey dashed line traces the *Pareto frontier*, connecting methods that are not dominated by any other method along this trade-off. Across all nine domains, our methods (EywaAgent, EywaMAS, and EywaOrchestra) consistently sit on or close to the Pareto frontier, achieving the strongest utility while spending substantially fewer tokens than competing multi-agent baselines.

foundation model should be invoked when applicable, which multi-agent topology should be used, and how the participating agents should be instantiated.

The planner prompt formulates orchestration as a constrained configuration-generation problem rather than a direct task-solving problem. The hard JSON constraint ensures that planner outputs can be parsed automatically and passed to the downstream execution engine without manual intervention. Meanwhile, the explicit fields for `setting`, `multi_agent_type`, `foundation_model`, and `agents` make the orchestration decision transparent and auditable.

Prompt Template B.1: General Prompt Structure

**General Prompt.** For all tasks in *EywaBench-mini*, we adopt a unified prompt template that specifies the task role, optional tool/model context, structured input field, expected output size, and task-specific output format. This design allows different task types to share the same high-level prompting interface while still preserving modality-specific instructions through specialized input tags and output constraints.

```
You are an expert in {task}. {server_description}.
{additional_instructions}

<{input_tag}>
{input_data}
</{input_tag}>

<output_size>
{output_size}
</output_size>

{output_format}
```

**Template fields.** The placeholder `{task}` specifies the task type. The field `{server_description}` provides the available model/tool context when an external foundation model is accessible through the server. The field `{additional_instructions}` contains task-specific guidance, such as explaining masked targets in tabular prediction tasks. The pair of tags `<{input_tag}>` and `</{input_tag}>` explicitly marks the input. Finally, `{output_size}` and `{output_format}` define the expected output length and response schema.

Figure 15: General prompt template used in *EywaBench-mini*. The template separates task identity, model/tool context, modality-specific input, output size, and response format, enabling a unified prompting interface across heterogeneous scientific tasks.

**Prompt Template B.2: Planner Prompt for *EywaOrchestra***

```
You are an orchestration planner for the Eywa Agentic System.

Your job is to choose an execution configuration for a single task.

Available LLM models and descriptions:
- ......

Available foundation models and descriptions:
- ......

Supported multi-agent topology pool:
- ......

Input task:
- Task Description: {task_description}
- Domain: {domain}
- Task Type: {task_type}

Hard constraints:
- Output must be valid JSON only (no markdown, no code fence, no extra text).
- ......
- If "setting" is "single-agent":
  - "model" must be a valid model string.
  - "multi_agent_type" must be null.
  - "foundation_model" should be in the available foundation models or null.
  - "agents" must be an empty list [].
- If "setting" is "multi-agent":
  - "model" must be null.
  - "multi_agent_type" must be in the topology pool.
  - "foundation_model" should be in the available foundation models or null.
  - "agents" must be a non-empty list of valid agent specs.

Output format:
{
  "eywa": true or false,
  "setting": "single-agent" or "multi-agent",
  "model": <llm_model> or null,
  "multi_agent_type": <multi_agent_topology> or null,
  "foundation_model": <foundation_model> or null,
  "agents": [<agent_spec_1>, <agent_spec_2>, ...] or []
}
```

Figure 16: Planner prompt template used by *EywaOrchestra*. Given the task description, domain, and task type, the planner outputs a structured JSON configuration that specifies whether to use single-agent or multi-agent execution, whether to enable *Eywa*, which foundation model to invoke, and how to instantiate the participating agents.