Symphony: A Decentralized Multi-Agent System for Co-Evolving Intelligence at Scale

Ji Wang[*]
Department of IEOR
Columbia University

Yuchun Feng[*]
Engineering Science
University of Toronto

Lynn Ai[†]
Gradient Network

Bill Shi
Gradient Network

Abstract

We introduce **Symphony**, a decentralized multi-agent system enabling co-evolving intelligence at scale across heterogeneous environments. Unlike previous centralized frameworks requiring centralized infrastructure, powerful GPUs, continuous connectivity, fine-tuned language models are deployed on edge and orchestrated by decentralized sparse communication, autonomous self-play, and continual evolution: each agent creates and completes tasks independently, learn from local task outcomes, and share only necessary information updates with other agents through a sparse communication strategy inspired by SPARTA. The system supports emerging skill specialization and autonomous asynchronous collaboration without centralized coordination. Curriculum alignment adaptation is achieved by continuously deploying new agents on the edge without central planner. This work innovatively formulates how intelligent agents can co-evolve over time by leveraging principles from federated learning, reinforcement learning, and preference optimization. Symphony lays the foundation for decentralized, trustless AI ecosystems where intelligence is collectively produced from the edge instead of being imposed from the center, leveraging transformative prospects in AI accessibility, privacy, safety, and agent-based economy.

1 Introduction

In recent years, large language models (LLMs) have transformed the landscape of artificial intelligence, demonstrating capabilities in natural language understanding, reasoning [1], planning [2], and tool usage [3]. These capabilities have enabled the emergence and widespread use of intelligent agents across diverse domains, including education [4], healthcare [5], autonomous systems [6], software engineering [7], and scientific discovery [8]. The rapid progress in model scaling, instruction tuning, and alignment techniques (e.g., RLHF, DPO) has enabled LLMs to perform increasingly sophisticated multi-step tasks with minimal supervision.

Despite these exciting developments, existing LLM-powered agent systems remain highly centralized, brittle, and limiting. Most frameworks are tightly coupled with centralized infrastructure, scheduling, and model synchronization, which introduce the following challenges:

- **High deployment cost:** Fine-tuned or API-based LLM agents require expensive server-grade GPUs (e.g., A100, H100), making large-scale deployment inaccessible to individuals or small teams.
- **Scalability bottlenecks:** Centralized planner-executor architectures (e.g., ReAct) often impose fixed topologies and communication protocols, limiting the system's flexibility and adaptability.

[*]Equal contribution
[†]Corresponding author: lynn@gradient.network

Preprint. Under review.

- **Learning rigidity:** Most agent systems depend on pre-defined instructions or static workflows. These systems lack ways to continuously improve themselves or learn curriculum by interacting with the environment.

In parallel, computing resources available at the edge (consumer-grade GPUs (RTX 3060/4090), Jetson boards, M-series Apple devices) have also become powerful and accessible. These resources represent a poorly tapped opportunity to run intelligent agents locally with enhanced privacy, and democratize access to AI. To bridge the capabilities of existing systems while taking advantage of the potential of these edge resources, we propose Symphony, a decentralized multi-agent system where lightweight LLMs running on edge devices coordinate with each other to enable self-evolving, experience-driven intelligence across heterogeneous environments. Symphony offers a new architectural design for agentic systems by coordinating modular, role-based agents that learn, reason, and adapt through decentralized coordination over fully autonomous agent behavior, policy learning, and beyond. Unlike conventional multi-agent systems restricted to task planning or policy learning via centralized orchestration, our system supports co-evolutionary agent behavior, cross-agent memory alignment, and edge-deployable adaptation and thus can collectively form an infrastructure for real-world deployment.

The system supports autonomous self improvement through iterative task solving and sparse parameter synchro- nization inspired by federated learning, reinforcement learning, and distributed optimization works. Our framework supports:

- **Autonomous self-play:** Agents generate their own tasks via proposer-solver dynamics, evaluate performance through local reward signals, and adapt their strategies accordingly.
- **Sparse communication:** Agents exchange only a small subset of critical parameters (e.g. LoRA deltas) using SPARTA-style asynchronous updates, preserving convergence during training even over unstable or low-bandwidth networks.
- **Dynamic specialization:** Over time, agents develop distinct skills, memory traces, and collaborative profiles, leading to emergent coordination patterns and decentralized knowledge evolution.

Our system builds upon recent advances in LLM reasoning, LoRA-based fine-tuning, sparse update syn chronization (e.g. SPARTA), and federated learning (FL). It enables the formation of an open, extensible agent society where individual contributors such as users, devices, and domain-specific agents can construct, share, and evolve their local intelligence over time. In addition to scaling across devices and networks, our system also democratizes the development and deployment of intelligent agents. Our system is a first step towards constructing trustless, self-organizing AI ecosystems where intelligence is not built upfront and served from the center, but instead evolves from the edge collectively, iteratively, and transparently.

2 Related Work

Sparse Distributed Training. Traditional distributed training synchronizes full sets of model parameters across nodes. This is infeasible on low-bandwidth or heterogeneous edge environments. SPARTA [9] proposed sparse parameter averaging where only a small fraction (e.g., 0.1%) of model parameters are communicated at each step. This drastically reduces bandwidth requirements while preserving model convergence through gradual correlation buildup. DiLoCo [9], inspired by federated learning, introduced a two-level optimization approach: nodes perform local updates using an inner optimizer and synchronize using an outer optimizer periodically. Our framework generalizes these concepts in the context of multi-agent learning, applying SPARTA-style synchronization across autonomous agents performing self-play updates. Unlike traditional data-parallel settings, our agents are loosely coupled and asynchronously optimized, necessitating even sparser and more resilient communication strategies.

Multi-Agent LLM Frameworks. AutoGen [10], CAMEL [11], and MetaGPT [12] represent state-of-the-art LLM orchestration platforms, enabling multiple agents to collaborate via role assignments and conversational protocols. While these systems are effective at tool chaining and human-in-the-loop interaction, they typically assume static agent configurations, centralized planning, and synchronous dialogue protocols. In contrast, our design is decentralized and learning-centric: agents

can dynamically propose tasks, evolve skills via local fine-tuning (e.g., LoRA), and engage in partially observable coordination through parameter sharing, without requiring an explicit planner.

Self-Play and Skill Acquisition. Self-play has long been used in reinforcement learning to bootstrap intelligent behavior without labeled data. AlphaZero [13] and OpenAI Five [14] demonstrated that agents trained purely through adversarial interactions can master complex sequential games. Recent work such as Absolute Zero [15] has extended this paradigm to code generation and symbolic reasoning via structured task proposals and self-verification loops. In LLMs, Direct Preference Optimization (DPO) [16] and Group Relative Policy Optimization (GRPO) [17] provide frameworks for reward modeling and preference alignment from implicit feedback. Our system integrates a lightweight form of self-play across agents: each agent proposes tasks, solves them, and receives feedback either locally or from neighboring agents, enabling decentralized, curriculum-aligned skill acquisition.

Memory and Continual Learning. Maintaining long-term memory and agent-specific knowledge is essential for multi-turn, self-evolving agents. LangGraph [18] uses hierarchical memory slots and local retrieval modules to support long-context reasoning. We follow these works and equip each edge agent with episodic memory buffers and small latent memory units to allow agents to store task history and update over multiple self-play iterations. In our system, memory is local (for agent internal state) and sparsely-shared (for inter-agent alignment), making it suitable for decentralized, bandwidth-limited deployment.

Edge Learning and Federated Optimization. The federated learning paradigm [19] promotes on-device model training using local data and minimal coordination. Techniques such as FedPAQ [20] reduce communication load via periodic averaging and quantization. However, most FL systems assume homogeneous client tasks and centralized aggregation servers. Our system differs from these systems by embedding heterogeneous LLM agents that train on diverse self-generated tasks and selectively synchronize using SPARTA-based mechanisms without central coordination. This combination of federated training, sparse updates, and self-play dynamics is uniquely adapted to the edge environment where both compute, bandwidth, and coordination budgets are limited.

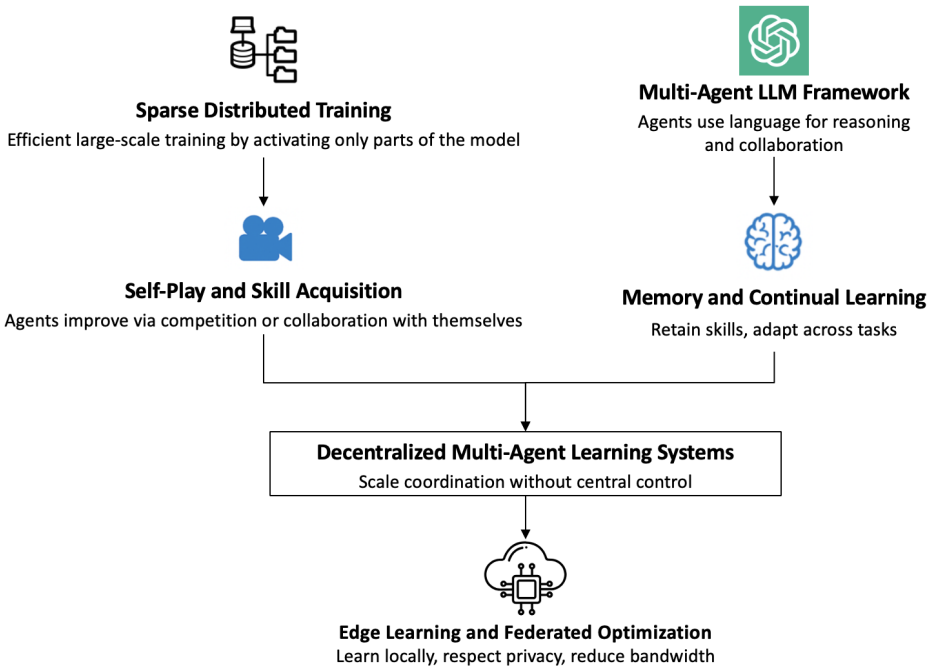

Figure 1: Road map for **Symphony** multi-agent LLM system

As shown in Figure 1, our proposed framework integrates five previously parallel but increasingly converging research directions. At the foundation, sparse distributed training (e.g., SPARTA) ad-

dresses the communication bottleneck involved in deploying LLMs across multiple nodes, especially in the edge environment. This low-bandwidth infrastructure enables the subsequent decentralized execution. Built on top of this foundation, federated and edge learning provide the system-level paradigm of distributed training under heterogeneity and partial observability such that agents can operate independently but learn collectively. To make such systems useful and intelligent, multi-agent LLM frameworks provide a form of structures for interaction that defines roles, workflows, and cooperative mechanisms. Most of these frameworks are, however, designed to be centrally controlled. This is where self-play and skill acquisition come into play: they provide an alternative mechanism for coordination such that agents can grow by solving tasks iteratively and learning from preferences, instead of pre-scripted cooperation. Finally, for this process to be persistent and adaptive, memory and continual learning mechanisms are indispensable as they provide each agent with a form of context retention and personalization, as well as a substrate for curriculum learning.

These five components are built upon each other: communication effi- ciency (SPARTA) enables scalable deployment (FL), which enables agent-level autonomy (multi-agent framework). This autonomy enables emergent be- havior (self-play) that persists over time through memory. Together, this ena-bles decentralized, evolving, and self-improving LLM ecosystems.

3 Problem Statement

While multi-agent systems (MAS) have shown promise in enhancing LLM capabilities via tool usage, role specialization, and cooperative workflows, current designs remain tightly coupled to centralized cloud orchestration and rigid role assignment. These systems exhibit the following limitations:

- **High communication costs:** Frequent full-model or hidden-state synchronizations between agents result in prohibitive bandwidth requirements.
- **Limited scalability:** As the number of agents grows, centralized scheduling and monitoring become bottlenecks.
- **Manual role engineering:** Human experts must define agent workflows, roles, and goals in advance, restricting system adaptability.
- **Lack of lifelong learning:** Agents rarely evolve beyond pretraining or single-session fine-tuning, missing opportunities to accumulate skills across episodes.

In contrast, our objective is to design a decentralized learning framework where lightweight LLM agents *autonomously* generate task, *collaborate asynchronously*, and *optimize* themselves through self-play. These agents operate independently and share only sparse information (such as compressed LoRA updates) without relying on centralized coordination or full supervision.

Let the multi-agent system be represented by a set of agents $\mathcal{A} = \{a_1, a_2, \ldots, a_N\}$. Each agent a_i maintains a local model parameterized by $\theta_i \in \mathbb{R}^d$, episodic memory M_i, and a local task generation-discovery loop. The global objective is to minimize cumulative regret or maximize collective skill utility $U(\mathcal{A})$ over a horizon T:

$$\max_{\theta_1,\ldots,\theta_N} \quad \frac{1}{T}\sum_{t=1}^{T}\sum_{i=1}^{N} R_i^t(\theta_i; M_i, \mathcal{T}_i^t) \tag{1}$$

where R_i^t is the local reward obtained from solving self-generated task $\mathcal{T}_i^t$ under current memory state M_i and parameters θ_i.

To avoid centralized aggregation, each agent updates its parameters using locally computed gradients $\nabla_{\theta_i} R_i^t$ and shares sparse deltas Δ_i^t using a communication-efficient protocol (e.g., SPARTA):

$$\theta_i^{t+1} = \theta_i^t + \eta \cdot \nabla_{\theta_i} R_i^t, \quad \text{with} \quad \Delta_i^t = \mathcal{S}(\theta_i^{t+1} - \theta_i^t) \tag{2}$$

Here $\mathcal{S}(\cdot)$ is a sparse projection operator that selects a subset of parameters or gradients (e.g., top-k or random mask).

The agents form a sparse graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$, where communication is constrained to neighbors in $\mathcal{E}$. The global system learns through asynchronous and decentralized coordination.

4 Key Research Questions

To enable scalable decentralized multi-agent learning, we identify the following key challenges:

RQ1: **Task Generation:** How can each agent a_i construct a distribution $p(\mathcal{T}_i)$ over tasks that promotes diversity, progression of difficulty, and maximization of long-term utility?

RQ2: **Skill Emergence:** Under what conditions does repeated proposer-solver interaction lead to skill improvement, and how can such improvement be measured via local utility R_i^t or cross-agent generalization metrics?

RQ3: **Communication Efficiency:** What is the optimal trade-off between communication sparsity $|\Delta_i^t|$ and convergence rate, and how can adaptive strategies modulate sparsity over time?

RQ4: **Memory Design:** What compact memory representations M_i (e.g., episodic buffers, vector summaries) support continual learning under limited device resources?

RQ5: **Self-Optimization:** Can we define a meta-controller π^m that governs agent-to-task assignment or routing using decentralized feedback from $\{R_i^t\}$, and what information is sufficient to bootstrap such a policy?

5 Proposed Framework: Symphony

5.1 System framework

We propose **Symphony**, a decentralized agent learning architecture designed to support scalable, resilient, and efficient multi-agent skill acquisition in edge environments. The name **Symphony** reflects the harmonious integration of three foundational pillars of the system: **Sparse Communication**, **Zero-shot Self-Play**, and **Incremental Learning**. Each pillar addresses a key challenge in decentralized AI deployment and introduces novel mechanisms for autonomy, collaboration, and continual adaptation.

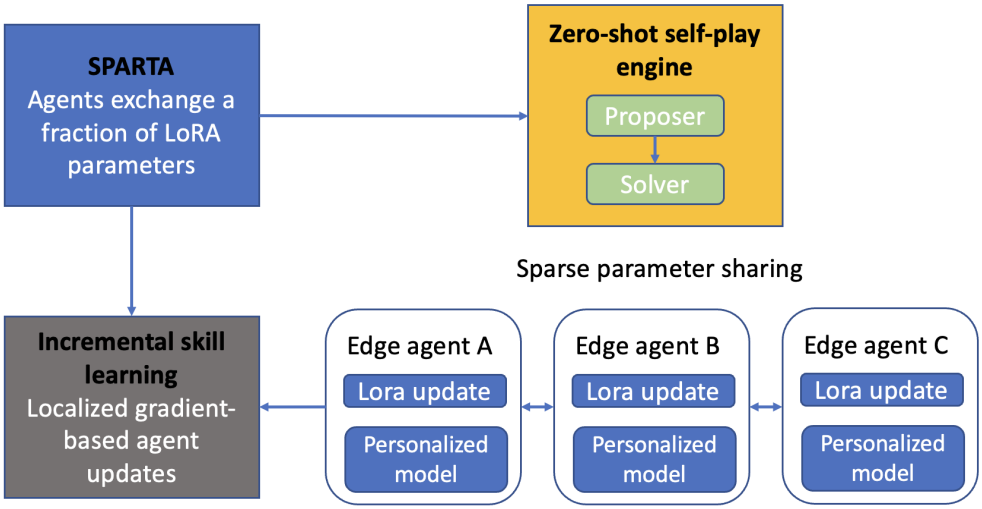

Figure 2: Overview of the **Symphony** framework. Each edge agent consists of a local self-play engine, LoRA-based personalized model, and gradient-based learning module. SPARTA enables sparse LoRA parameter sharing between agents. The system operates without centralized coordination and supports decentralized continual learning.

- **Sparse Communication (SPARTA)**
  Traditional distributed systems suffer from high communication overhead due to full-parameter synchronization. **Symphony** introduces a SPARTA-based strategy, where each agent a_i transmits only a sparse subset of its parameter update Δ_i^t:

$$\Delta_i^t = \mathcal{S}(\theta_i^{t+1} - \theta_i^t), \tag{3}$$

where $\mathcal{S}(\cdot)$ is a sparsification function (e.g., top-k magnitude or random mask).

Agents form a dynamic communication graph $\mathcal{G} = (\mathcal{A}, \mathcal{E})$, where $\mathcal{E}$ represents active peer links. The update from agent a_i is broadcast asynchronously to its neighborhood $\mathcal{N}_i$:

$$\theta_j^{t+1} \leftarrow \theta_j^t + \alpha \cdot \mathcal{A}(\Delta_i^t), \quad \forall j \in \mathcal{N}_i, \tag{4}$$

where $\mathcal{A}(\cdot)$ denotes a transformation function (e.g., normalization, scaling) and α is a mixing factor. This results in minimal bandwidth usage while maintaining parameter correlation across agents.

- **Zero-shot Self-Play Engine**
  Each agent maintains a self-contained task generation and evaluation loop. At time t, the agent samples a task $\mathcal{T}_i^t$ from its evolving task distribution:

  $$\mathcal{T}_i^t \sim p(\mathcal{T}_i \mid M_i, \theta_i^t), \tag{5}$$

  where M_i is the episodic memory buffer that encodes past interactions and feedback. The agent solves the task to produce an output $\hat{y}_i^t$ and evaluates it using a judge function $\mathcal{J}_i$:

  $$R_i^t = \mathcal{J}_i(\hat{y}_i^t, \mathcal{T}_i^t, M_i), \tag{6}$$

  where R_i^t is the scalar reward or utility signal used to guide learning. Notably, the proposer-solver-judge loop allows for curriculum generation, reward shaping, and skill emergence without external datasets.

  Agents can further augment self-play by broadcasting selected tasks or evaluations to peers, allowing for cooperative benchmarking and behavioral diversity.

- **Incremental Skill Learning**
  Upon receiving feedback R_i^t, the agent performs a localized update to its LoRA adapter parameters ϕ_i:

  $$\phi_i^{t+1} = \phi_i^t + \eta \cdot \nabla_{\phi_i} R_i^t, \tag{7}$$

  where the gradient is computed through REINFORCE-style sampling, contrastive loss, or supervised targets depending on the task type. To avoid parameter drift, agents apply regularization using prior distributions or experience replay. The update is sparsified as in the SPARTA protocol for cross-agent sharing.

  Over time, each agent evolves its policy $\pi_i(\cdot \mid \phi_i)$ and task generator $p(\mathcal{T}_i)$, resulting in an emergent distribution of skills across the network. This leads to division of labor, specialization, and emergent cooperation.

The **Symphony** architecture is novel in combining:

- **Sparse asynchronous communication**, which scales gracefully to bandwidth-constrained, heterogeneous edge environments;
- **Autonomous curriculum and task evolution**, decoupled from external labels or datasets;
- **LoRA-based lightweight continual learning**, enabling incremental policy adaptation without full-model updates.

Together, these innovations support lifelong, decentralized skill emergence among LLM agents running on affordable hardware with minimal infrastructure assumptions.

5.2 Community Role Interaction

The proposed decentralized agent network is built upon the collaborative engagement of five distinct community roles, each contributing unique capabilities to the training, evaluation, and evolution of intelligent agents. These roles are carefully designed to support system scalability, domain specificity, and automation of self-improvement. Below we outline each role and its technical interaction within the system.

(1) Gradient Network Core Team As the initiator of the ecosystem, the Gradient Network Core Team plays a pivotal role in:

- Establishing foundational protocols, including agent communication (beacon/response), reward modeling frameworks, and trust propagation mechanisms.
- Deploying seed agents (both Functional Intelligent Nodes and Lightweight Task Nodes) with initial capabilities and model checkpoints.
- Maintaining a public knowledge base and versioning logic for base models, toolchains, and task templates.

This group ensures the network is bootstrapped with the necessary infrastructure to support open participation and adaptive evolution.

(2) Compute Providers Ordinary users with access to consumer-grade GPUs can become Compute Providers by hosting decentralized agents. Their roles include:

- Running nodes that participate in task execution, LoRA-based self-adaptation, and multi-agent self-play.
- Storing and replaying past interactions to enrich memory-based adaptation modules.
- Engaging in federated evaluation tasks when selected via beaconing.

We offer one-click deployment kits and model-agnostic LoRA patching pipelines, enabling rapid onboarding and decentralized scalability.

(3) Task Requesters Requesters initiate tasks, ranging from question answering to planning or domain-specific tool use. Their contributions include:

- Providing diverse task prompts, enhancing the language distribution coverage of training data.
- Optionally providing structured input (e.g., document, schema, constraints) that refines task specifications.
- Receiving outputs from candidate agents and optionally provide scalar feedback, binary flags, or comparative rankings.

Technically, their input is parsed by an Intent Understanding Module, which informs agent selection and prompt composition in the FIN network.

(4) Reward Providers Reward Providers play a crucial role in aligning the network's behavior with human values and downstream preferences. They:

- Evaluate task outputs via predefined rubrics or crowd-sourced preferences.
- Help finetune reward models (e.g., RM or R(s, a)) that enable RLHF-style policy updates.
- May also include agents that function as learned-reward critics trained on prior human evaluations.

A central pipeline is designed to convert human feedback into structured preference datasets used in Direct Preference Optimization (DPO) or Reward Modeling (RM).

The reward that each agent receives is composed of two parts: the local reward provided by a certain Reward Provider, and the shared reward which is a windowed delayed average of all local rewards. The shared reward reflects the collective performance of the framework, ensuring consistent value alignment across agents while addressing independent tasks.

(5) Vertical Partners (Human Experts or Multi-Agent Judge) To improve agents' performance in specialized domains such as math, finance, healthcare and education, Vertical Partners from those fields, who can be either human experts or a specialized multi-agent judge system, are invited to participate in the pipeline of Symphony.

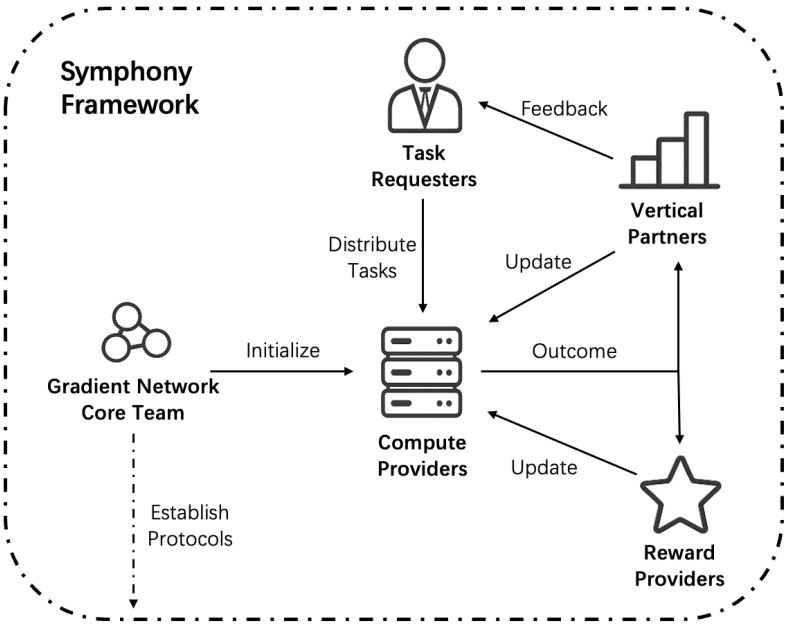

Figure 3: Agent collaboration framework

a. Human Experts Human experts play a critical role in enhancing the precision of domain-specialized tasks. Their contributions include:

- Providing high-quality domain-specific corpus and structured schemes
- Suggesting optimal system prompts for specialized tasks
- Reviewing model output and providing ideal answers for pairwise comparison reinforcement learning.

b. Multi-Agent Judge To reduce the framework's dependency on manual feedback, we support the CollabEval multi-agent judge framework [21], in which multiple LLM agents possess independent judgment and reach collaborative decision through multi-round discussions.

Collaboration Summary The five roles of Symphony form a self-reinforcing loop (Figure 3). Task Requesters receive, clarify, and decompose user requests, and distribute subtasks to Computer Providers through beacon. Compute Providers, which are set on edge devices, asynchronously address subtasks, and the generated results undergo evaluation by Reward Providers, who provide reward signals for LoRA updates. Vertical Partners participate in both intent clarification and reward generation, providing high-quality feedback. In the loop, the Gradient Network Core Team coordinates protocol standards and version consistency. This decentralized loop, composed of task execution, expert evaluation, and iterative improvement, enables sustainable agent evolution while leveraging community expertise.

5.3 User Intent Clarification Protocol

In a decentralized multi-agent framework, precisely understanding human requirements and translating them into machine-actionable tasks set the foundation for collaborative task completion. Our architecture introduces a multi-stage interaction protocol, integrating prompt engineering, context interpretation, and expert feedback to guarantee the precision of users' intents.

5.3.1 Intent Acquisition and Clarification

To guarantee that task descriptions and requirements submitted by users can be fully understood by autonomous agents (Task Requester), we designed a pipeline which integrated prior work in

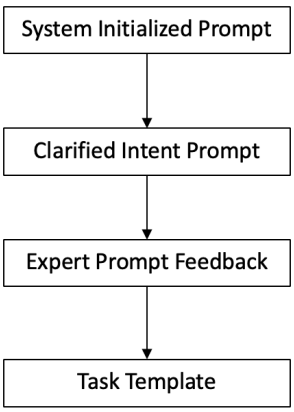

Figure 4: Multistage prompt workflow

clarification-based prompting [22, 23], intent disambiguation [24], and rationale-driven feedback labeling [25]. The pipeline includes the following parts:

1. **Initial Prompt Interpretation:** A system prompt is used to transfer user requirements into structured templates (e.g., "Translate user inquiry into a structured goal-action-description triplet.").

2. **Ambiguity Detection Module:** Inspired by the AT-CoT framework [26], the model will generate a clarification question to refine user requirements when identifying ambiguities.

5.3.2 Expert-Guided Intent Specialization

To improve the precision of intent understanding and achieve alignment between user requirements and system capabilities in specialized domains, rather than fine-tuning the model on fixed instructions, we implement an interactive loop supported by human experts:

- When receiving clarified queries from real user requirements, human experts suggest optimal system prompt or structured schemes (e.g., SQL query skeletons, math operation trees, logic chains), which will be stored in high-quality corpus of the corresponding domain to support future similar tasks and self-play fine-tuning.

- A dual-evaluation method — pairwise comparison (like V-STaR [27]) and content-based assessment (like LLM-based judges [28]) — is used to score agent output against expert-crafted content. The evaluation outcomes are transformed into preference pairs or reward signals for LoRA-based model update on Task Requesters.

5.3.3 Multistage Prompt Workflow

As shown in figure 4, the system prompt pipeline is modularized, integrating both fixed schemes (e.g. Ambiguity Detection Module) and dynamic improvements based on human expert feedback.

Summary

By combining structured task template, ambiguity detection, expert refinement, and on-device fine-tuning, Symphony is able to narrow the gap between users and LLM. This design fosters a sustainable flow of the understanding of sophisticated tasks, especially in specialized domains.

5.4 Role-Specific Technical Interfaces and Support

5.4.1 Compute Providers - Lightweight Deployment Support

Our framework allows low-friction participation for compute providers. Agents can be launched on consumer-level GPUs (e.g., RTX 3060/4060, Apple M-series) through a containerized deployment pipeline.

- **LoRA-tunable LLM:** Agents are launched with a quantized 7B or smaller base model (e.g., Mistral, TinyLLaMA) along with local LoRA adapters, enabling efficient updates.
- **Minimal dependencies:** Agents only require Python, Docker, and optionally a WebRTC or libp2p runtime for P2P communication.
- **Auto-registration:** Upon launch, each compute provider registers capabilities via DID, broadcasts its existence to other nodes in the network, and starts listening for task-relevant beacons.
- **Self-play loop:** Even in the absence of external user requests, compute providers can automatically propose and solve tasks to enhance capabilities.

This design of compute provider makes full use of edge compute capacity, while encouraging agents to specialize utilizing LoRA fine-tuning.

5.4.2 Task Requesters — Intent Clarification and Task Routing Infrastructure

Users submit task requests in natural language or structured form. To support interface with users, our framework includes:

- **Intent Parsing Pipeline:** Requests are processed Task Requesters, which is a lightweight model (e.g., Phi-2 or distilled Mistral) running on a consumer-level edge device. The pipeline involves:
  - Capability matching (via vector search or keyword tags)
  - Task decomposition into subtask DAG
- **ISEP Protocol (Iterative Scope Exploration Protocol):** Decomposed subtasks are broadcast as `Beacons` across the network, include:
  - Skill requirements
  - Task type and urgency
  - TTL (Time-To-Live) for propagation scope
- **Routing Optimization:** When receiving beacons, Compute Providers respond with confidence scores. The executors are chosen from candidates according to skill match, past performance, or cost metadata.

This decentralized routing ensures that users don't need to know which agent performs what; the system dynamically matches tasks to best-matching compute providers.

5.4.3 Domain Experts — Structured Dialog-Driven Feedback

Domain experts participate in both intent clarification and result evaluation stages. The framework transforms their feedback into reward signals through:

1. **Clarification Interface:**
   - When ambiguity is detected in user requests, Task Requesters initiate clarification utilizing Clarify-or-Respond style prompts.
   - Experts disambiguate user intents via multi-choice or natural feedback.
   - Clarified queries are re-injected for final processing.
2. **Evaluation Interface:**
   - Experts review agent responses and provide preference labels (A vs. B) and rationales.
   - Expert feedback is structured as:

```
{
  "query": "...",
  "response_A": "...",
  "response_B": "...",
  "preferred": "B",
  "rationale": "B provides domain-specific detail"
}
```

- Expert feedback is used for DPO-style local fine-tuning via LoRA

3. **Expert-Defined Task Templates:**
   - Experts define reusable scaffolds for specific tasks (e.g., "Always cite ICD-10 when answering diagnosis-related queries.").
4. **Impact Visualization:**
   - Experts receive dashboards showing how their feedback improves system performance.

This expert interface loop ensures that domain knowledge is encoded into the framework without centralized retraining, and fosters a culture of reusable expertise.

In summary, our technical strategy is role-sensitive:

- Compute providers sustainably evolve utilizing plug-and-play deployment and self-training.
- Task Requesters interact with users through a dynamic intent-routing protocol with automatic task matchmaking.
- Experts drive sustainable model improvement through structured clarification and preference feedback.

5.5 Model Training and Optimization Strategy

To adaptively align agent behavior with human preference, our framework incorporates a hybrid training framework that integrates prompt engineering, offline preference learning, and reinforcement learning (RL)-based fine-tuning.

1. Feedback Loop for Fine-Tuning Compute Providers generate candidate outputs in response to subtasks, which are evaluated by human experts or automated reward models, yielding pairwise comparisons or scalar reward signals. These feedback signals are then fed into:

- **Offline RL algorithms**, such as Advantage-Weighted Regression (AWR) or Proximal Policy Optimization (PPO), which adapt model policies from logged interaction data.
- **Preference-based optimization** methods (e.g., Direct Preference Optimization (DPO) [16]) that directly train policies on ranked data without explicit reward modeling.

The fine-tuning process is typically performed using LoRA-style adapters to enable parameter-efficient on-device updates. As shown in the pipeline below:

$$\texttt{sample} \rightarrow \texttt{model} \rightarrow \texttt{feedback} \rightarrow \texttt{RL algorithm} \rightarrow \texttt{agent}$$

2. Feedback Quantity and Structure Previous work suggests that even modest volumes of high-quality feedback (500 - 1000 preference pairs) can significantly improve model alignment within targeted domains [29, 30, 31]. The participation of domain experts in our framework guarantees that high-quality reward signals can be collected.

Our system allows feedback to be structured as following:

- Explicit scalar rewards: $\mathcal{R}(x, y)$
- Ranked preference pairs: (y_1, y_2) such that $y_1 \succ y_2$
- Annotated errors or hallucination flags

These signals are stored in structured logs and optionally aggregated across agents for federated learning.

3. Decoupled Training Phases To improve convergence and maintain modularity, we decouple user intent understanding and task routing from model-level fine-tuning. The former is improved via few-shot and chain-of-thought prompting techniques (based on expert feedback), while the latter evolves asynchronously via local feedback loops.

4. Algorithmic Backbones We adopt recent breakthroughs in reward modeling and policy optimization:

- **DPO (Direct Preference Optimization)** [16]
- **RRHF (Ranked Reward HF)** [32]
- **SPIN (Structured Preference Inference)** [33]

These methods allow low-resource LoRA fine-tuning to better align user preferences.

6 Experimental Step

A series of experiments are designed to evaluate the feasibility, scalability, and learning efficiency of **Symphony** framework in decentralized, resource-constrained environments. Our goal is to demonstrate that self-play agents equipped with LoRA-based models and SPARTA-style sparse communication can independently acquire, improve and specialize capabilities, without centralized coordination.

Experiments are conducted utilizing $N = 3$ to 5 edge nodes with heterogeneous hardware configurations (e.g., NVIDIA RTX 3060/4090, Apple M2, Jetson Orin). Each node hosts one or more lightweight LLM agents configured as follows:

6.1 Agent Configuration:

- **Model:** TinyLLaMA, Mistral 7B Q4, and Phi-2, all equipped with LoRA adapters (~8–16M trainable parameters).
- **Memory:** Local episodic buffer M_i stores past task-performance traces; optional retrieval supports long-context augmentation.
- **Tasks:** We evaluate on GSM8K (math reasoning), Natural Questions (open QA), HotpotQA (multi-hop reasoning), and Text2SQL (schema-to-query translation).

6.2 Centralized vs Decentralized Training Paradigms

Each training paradigm uses the same base model and is optimized with GRPO. Testing tasks are collected from the same distribution, and the same fixed set of compute devices is used in all conditions to ensure fairness in compute resources. All experiments are repeated with three different random seeds for statistical robustness [34].

- **Synchronous Centralized**: Centralized training situation serves as the upper-bond baseline. Training and inference are executed sequentially on the same server, using the latest policy weights for result generation.
- **One-Step Asynchronous Centralized**: Inference and training are decoupled across dedicated nodes with fast interconnects. Results are generated using policy weights from the previous RL step, resulting in a one-step off-policy lag.
- **Two-Step Asynchronous Decentralized**: Inference is performed by untrusted, geographically distributed workers. Due to slower communication and checkpoint synchronization, workers typically operate on policy weights that are two or more steps outdated. This delay introduces additional off-policy drift and represents a more realistic setup for scalable, permissionless RL.

6.3 Efficient Weight Synchronization with SPARTA

In the Two-Step Asynchronous Decentralized setup [34], a key challenge is the transmission of large model weights to many distributed inference nodes. To mitigate this bottleneck, we incorporate SPARTA [35], a sparse parameter synchronization method originally proposed to reduce bandwidth overhead in distributed training.

Instead of broadcasting full model weights after each training update, SPARTA performs:

- Sparse gradient thresholding, selecting only top-k magnitude updates across parameters.
- Compression of selected parameters, followed by efficient streaming via SHARDCAST.
- Local parameter merging on inference workers using momentum-based averaging to maintain stability in asynchronous settings.

We find that using SPARTA in the two-step setting improves rollout freshness and allows more inference workers to remain synchronized with recent policies.

6.4 Key Evaluation Metrics:

- *Task Accuracy and Generalization:* Success rate on held-out benchmarks and task difficulty scaling.
- *Skill Specialization:* Measured via agent-task affinity and diversity of emergent expertise.
- *Parameter Similarity:* Intra-agent smoothness and inter-agent cosine similarity of LoRA updates.
- *Interaction Success:* Reward, completion success rate, and quality of multi-agent collaborations.
- *Bandwidth Efficiency:* Accuracy and reward gain per kilobyte of synchronized parameter updates.
- *Stability:* Convergence behavior, variance across seeds, and robustness to asynchronous update delays.

Experiments are tracked via a centralized dashboard for post-hoc evaluation, and logs include agent-level trace histories, parameter drift plots, and reward trajectories. All hardware setups, model weights, and code will be released for reproducibility.

7 Broader Impact

Symphony improves upon existing multi-agent LLM frameworks such as AutoGen [10] and MetaGPT [12] with its decentralized framework system and sets up the foundation for scaling and strengthening applications powered by DPO [16] and GRPO [17], where continuous learning and evaluation is required to optimize, critique, and improve agents in parallel, scaling across compute nodes, domains, or task variants. Moreover, the decentralized, self-evolving framework of **Symphony** extends beyond the technical advancements, possessing transformative potentials in AI accessibility, privacy safety, and business model.

- **Open access AI:** By reducing hardware requirements to consumer GPUs (e.g. RTX 3060, Apple M-series) and adopting SPARTA-stype sparse updates, **Symphony** eliminates dependency on centralized cloud infrastructure. This enables individuals, small teams, and any communities with limited compute resources to participate in the co-creation of intelligence, significantly reducing entry barriers.
- **Privacy by Architecture** Privacy of **Symphony** participators is guaranteed by the decentralized paradigm. With task generation, parameter update, and self-evolvement being totally restricted on individual devices, the broadcasted information within the framework is only the sparse gradient required by the SPARTA technique, and sensitive data can never leave local storage.
- **Decentralized Agent Economies** By realizing agent-level autonomy, **Symphony** empowers agents to act independently, make local judgments, and engage in decentralized decision-making and evaluation. This aligns with economic systems where agents behave like rational participants, bidding for tasks, offering services, and collaborating based on incentives. It lays the groundwork for agent-based economies where agents evolve and make decisions in environments without static logic or hard-coded policies.

A System Architecture

Decentralized Ledger A decentralized ledge is used to store critical operational information within the Symphony ecosystem, independently tracking each agent's resource ownership, contribution records, and domain expertise. Each agent possesses a unique cryptographic address (DID-compliant), which is used for API interaction, contribution recording, and verification of resource ownership. The ledger also maintains evolving vectors for each agent's capability (e.g. DID_7B3F: ["biomedical_qa": 0.94, "clinical_diagnosis": 0.82]).

Worker Nodes Worker nodes are edge devices (e.g., consumer-grade GPUs like NVIDIA RTX 3060/4090, or Apple M-series) with full local autonomy. Each worker node consists of 4 components:

- Local Engine: Quantized LLM (e.g. TinyLLaMA, Mistral 7B Q4) with LoRA adapters for efficient on-device training
- Self-Play Module: Autonomous task generator with domain-specific customization
- Episodic Memory Buffer: Storage of performance records
- Communicator: A communication model utilizing SPARTA

Orchestrator The Orchestrator serves as the communication bridge between Task Requesters and Compute Providers, while preserving data privacy and security. It is the dynamic coordination hub for Symphony's ecosystem, monitoring current available compute resources by retrieving information from Discovery Service and enabling efficient task distribution without centralized control.

Role-Specific Gateways Role-Specific Gateways provide standardized interfaces for Symphony's 5 roles (Core Team, Task Requester, Compute Provider, Reward Provider, and Vertical Partner).

Discovery Service The Discovery Service maintains a dynamic registration dataset of available edge devices and their information. The dataset is only accessible to authenticated components like Orchestrator, reducing the risk of data leakage.

References

[1] Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyan Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, et al. Llm reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models. *arXiv preprint arXiv:2404.05221*, 2024.

[2] Ziqi Zhou, Jingyue Zhang, Jingyuan Zhang, Boyue Wang, Tianyu Shi, and Alaa Khamis. In-context learning for automated driving scenarios. *arXiv preprint arXiv: 2405.04135v1*, 2024.

[3] Imad Eddine Toubal, Aditya Avinash, Neil Gordon Alldrin, Jan Dlabal, Wenlei Zhou, Enming Luo, Otilia Stretcu, Hao Xiong, Chun-Ta Lu, Howard Zhou, et al. Modeling collaborator: Enabling subjective vision classification with minimal human effort via llm tool-use. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 17553–17563, 2024.

[4] Zheyuan Zhang, Daniel Zhang-Li, Jifan Yu, Linlu Gong, Jinchang Zhou, Zhanxin Hao, Jianxiao Jiang, Jie Cao, Huiqin Liu, Zhiyuan Liu, et al. Simulating classroom education with llm-empowered agents. *arXiv preprint arXiv:2406.19226*, 2024.

[5] Charlene H Chu, Simon Donato-Woodger, Shehroz S Khan, Rune Nyrup, Kathleen Leslie, Alexandra Lyn, Tianyu Shi, Andria Bianchi, Samira Abbasgholizadeh Rahimi, and Amanda Grenier. Age-related bias and artificial intelligence: a scoping review. *Humanities and Social Sciences Communications*, 10(1):1–17, 2023.

[6] Chen Yang, Yangfan He, Aaron Xuxiang Tian, Dong Chen, Jianhui Wang, Tianyu Shi, Arsalan Heydarian, and Pei Liu. Wcdt: World-centric diffusion transformer for traffic scene generation. *arXiv preprint arXiv:2404.02082*, 2024.

[7] Xinying Hou, Yanjie Zhao, Yue Liu, Zhou Yang, Kailong Wang, Li Li, Xiapu Luo, David Lo, John C. Grundy, and Haoyu Wang. Large language models for software engineering: A systematic literature review. *arXiv preprint arXiv:2308.10620*, 2023.

[8] Yanzheng Wang, Boyue Wang, Tianyu Shi, Jie Fu, Yi Zhou, and Zhizhuo Zhang. Sample-efficient antibody design through protein language model for risk-aware batch bayesian optimization. *bioRxiv*, pages 2023–11, 2023.

[9] Arthur Douillard, Qixuan Feng, Andrei A Rusu, Rachita Chhaparia, Yani Donchev, Adhiguna Kuncoro, Marc'Aurelio Ranzato, Arthur Szlam, and Jiajun Shen. Diloco: Distributed low-communication training of language models. *arXiv preprint arXiv:2311.08105*, 2023.

[10] 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. *arXiv preprint arXiv:2308.08155*, Aug 2023. v2 updated October 3, 2023.

[11] Guohao Li, Hasan Abed Al Kader Hammoud, Hani Itani, Dmitrii Khizbullin, and Bernard Ghanem. Camel: Communicative agents for "mind" exploration of large language model society. In *NeurIPS*, 2023.

[12] Sirui Hong, Mingchen Zhuge, Jiaqi Chen, Xiawu Zheng, Yuheng Cheng, Ceyao Zhang, Jinlin Wang, 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, 2023. `https://github.com/geekan/MetaGPT`.

[13] David Silver et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[14] OpenAI. Openai five. *blog.openai.com*, 2019.

[15] Andrew Zhao, Yiran Wu, Yang Yue, Tong Wu, Quentin Xu, Matthieu Lin, Shenzhi Wang, Qingyun Wu, Zilong Zheng, and Gao Huang. Absolute zero: Reinforced self-play reasoning with zero data. *arXiv preprint arXiv:2505.03335*, 2025.

[16] Ramin Rafailov et al. Direct preference optimization: Your language model is secretly a reward model. *arXiv preprint arXiv:2305.18290*, 2023.

[17] Milan Vojnovic and Se-Young Yun. What is the alignment objective of grpo?, 2025. Accessed: 2025-06-27.

[18] Jialin Wang and Zhihua Duan. Agent ai with langgraph: A modular framework for enhancing machine translation using large language models, 2024. Accessed: 2025-06-27.

[19] Peter Kairouz et al. Advances and open problems in federated learning. *Foundations and Trends® in Machine Learning*, 14(1–2):1–210, 2021.

[20] Amirhossein Reisizadeh et al. Fedpaq: A communication-efficient federated learning method with periodic averaging and quantization. In *ICML*, 2020.

[21] Yiyue Qian, Shinan Zhang, Yun Zhou, Haibo Ding, Diego Socolinsky, and Yi Zhang. Enhancing llm-as-a-judge via multi-agent collaboration. 2025.

[22] Alina Leippert, Tatiana Anikina, Bernd Kiefer, and Josef Genabith. To clarify or not to clarify: A comparative analysis of clarification classification with fine-tuning, prompt tuning, and prompt engineering. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 105–115, 2024.

[23] Oliver Kramer and Jill Baumann. Unlocking structured thinking in language models with cognitive prompting. 2024. Preprint.

[24] Yutao Mou, Xiaoshuai Song, Keqing He, Chen Zeng, Pei Wang, Jingang Wang, Yunsen Xian, and Weiran Xu. Decoupling pseudo label disambiguation and representation learning for generalized intent discovery. *arXiv preprint arXiv:2305.17699*, 2023.

[25] Lei Li, Yuancheng Wei, Zhihui Xie, Xuqing Yang, Yifan Song, Peiyi Wang, Chenxin An, Tianyu Liu, Sujian Li, Bill Yuchen Lin, Lingpeng Kong, and Qi Liu. Vlrewardbench: A challenging benchmark for vision-language generative reward models. *arXiv preprint arXiv:2411.17451*, 2024.

[26] Anfu Tang, Laure Soulier, and Vincent Guigue. Clarifying ambiguities: on the role of ambiguity types in prompting methods for clarification generation. *arXiv preprint arXiv:2504.12113*, 2025.

[27] Arian Hosseini, Xingdi Yuan, Nikolay Malkin, Aaron C. Courville, Alessandro Sordoni, and Rishabh Agarwal. V-STaR: Training verifiers for self-taught reasoners. *arXiv preprint arXiv:2402.06457*, 2024.

[28] Sijun Tan, Siyuan Zhuang, Kyle Montgomery, William Y. Tang, Alejandro Cuadron, Chenguang Wang, Raluca Popa, and Ion Stoica. Judgebench: A benchmark for evaluating llm-based judges. *arXiv preprint arXiv:2405.08573*, 2024.

[29] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*, 2022.

[30] Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. Lima: Less is more for alignment. *arXiv preprint arXiv:2305.11206*, 2023.

[31] Vighnesh Subramaniam, Yilun Du, Joshua B Tenenbaum, Antonio Torralba, Shuang Li, and Igor Mordatch. Multiagent finetuning: Self improvement with diverse reasoning chains. *arXiv preprint arXiv:2501.05707*, 2025.

[32] Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*, 2023.

[33] Wasu Top Piriyakulkij, Volodymyr Kuleshov, and Kevin Ellis. Active preference inference using language models and probabilistic reasoning. *arXiv preprint arXiv:2312.12009*, 2023.

[34] Prime Intellect Team, Sami Jaghouar, Justus Mattern, Jack Min Ong, Jannik Straube, Manveer Basra, Aaron Pazdera, Kushal Thaman, Matthew Di Ferrante, Felix Gabriel, et al. Intellect-2: A reasoning model trained through globally decentralized reinforcement learning. *arXiv preprint arXiv:2505.07291*, 2025.

[35] Jiawen Liu, Jie Ren, Roberto Gioiosa, Dong Li, and Jiajia Li. Sparta: High-performance, element-wise sparse tensor contraction on heterogeneous memory. In *Proceedings of the 26th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming (PPoPP)*, pages 312–324. ACM, 2021.