ACTIVE LEARNING FOR COMMUNICATION STRUCTURE OPTIMIZATION IN LLM-BASED MULTI-AGENT SYSTEMS

Anonymous authors
Paper under double-blind review

ABSTRACT

Optimizing the communication structure of large language model based multi-agent systems (LLM-MAS) has been shown to improve downstream performance and reduce token usage. Existing methods typically rely on randomly sampled training tasks. However, tasks may differ substantially in difficulty and domain, and thus they are not equally informative for updating communication structure, making optimization often unstable and highly sensitive to the particular training set. To actively identify the most valuable tasks for communication-structure optimization, we propose an ensemble-based information-theoretic task selection framework. The proposed method estimates task informativeness by how much a candidate task changes the distribution over graph parameters, using ensemble Kalman inversion as an efficient and derivative-free approximation of the corresponding Bayesian update. The resulting estimator is especially suitable for black-box and noisy multi-agent systems. To enhance scalability, we construct a compact candidate pool through embedding-based representative selection and combine the informative selection with surrogate modeling and batch Thompson sampling. We validate the proposed framework across both benign and adversarial settings and multiple task formats. It consistently outperforms random training, demonstrating more effective task selection and greater overall cost efficiency.

1 INTRODUCTION

Large language model (LLM) based agents are now applied beyond generic reasoning to more complex problems, such as clinical decision support (Li et al., 2024a; Wang et al., 2025b) and scientific machine learning (Wang et al., 2025a; Gaonkar et al., 2025). LLM-based multi-agent systems (LLM-MAS) further extend this direction by coordinating multiple specialized agents through inter-agent communication (Li et al., 2024b; He et al., 2025). In many existing systems, however, communication structures are empirically predefined, such as fully connected or heuristically designed interaction graphs, rather than optimized for the target task distribution (Hong et al., 2024; Qian et al., 2024). Such empirical designs of communication structures have negative impacts on downstream performance of LLM-MAS, e.g., efficiency, accuracy, stability, and robustness. Specifically, redundant edges can waste token usage, and unrestricted communication can propagate hallucinations or adversarial content (Zeng et al., 2025; Shen et al., 2025). Therefore, communication-structure optimization is an important problem in LLM-MAS (Zhang et al., 2024b; Yang et al., 2025; Li et al., 2026; Zhang et al., 2025a). By removing redundant interactions and restricting harmful communication pathways, an optimized communication structure can improve the overall downstream performance of LLM-MAS, while also helping identify less reliable agents and limit their influence (Zhang et al., 2025b; Leong et al., 2025; Zhang et al., 2024a; Wang et al., 2025c; Bi et al., 2025).

We observe that communication-structure optimization of LLM-MAS can be highly sensitive to the choice of training tasks. Existing methods typically select training tasks randomly (Zhang et al., 2025b; Leong et al., 2025), even though different tasks may favor substantially different communication structures (Zhang et al., 2024a; 2025a). We show in Section 2 that communication-structure optimization under random task selection is unreliable in two respects: downstream accuracy varies substantially across independent runs, and increasing the number of training tasks does not

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.

consistently improve accuracy, with accuracy decreasing after reaching its peak. These observations suggest that the key question is not only how many tasks are used for training, but also which tasks are used. This naturally motivates an active learning perspective: can we identify the tasks that are most informative for communication-structure optimization?

The key challenge in this active learning perspective is to estimate task informativeness efficiently and reliably for communication-structure optimization. This is a three-level black-box estimation problem: candidate tasks are evaluated through stochastic and non-differentiable LLM-MAS rollouts; structure optimization is performed on top of this process, using RL or policy-gradient methods (Bi et al., 2025; Zhang et al., 2025a); and task utility (i.e., information gain) is further defined by the induced change in the communication-structure parameters. This nesting makes task-utility estimation computationally expensive. Direct sample-based utility estimators typically require many rollouts or repeated optimization runs, while gradient- or sensitivity-based estimators, when applied to this nested black-box pipeline, also require repeated rollouts to control variance. We therefore adopt an ensemble-based information-theoretic approach, which avoids end-to-end gradient estimation, reduces the need for large-scale repeated sampling, and enables more efficient and reliable task selection.

In this work, we first empirically characterize the sensitivity of structure optimization to training-task selection. To address this issue, we propose an ensemble-based information-theoretic active learning framework via a two-stage selection strategy. First, we identify a representative task subset using semantic embeddings to ensure diversity. Second, we quantify task utility through the information gain in structure parameters, using ensemble Kalman inversion (EKI) (Iglesias et al., 2013; Kovachki & Stuart, 2019)) to approximate Bayesian posterior updates, which circumvents the need for calculating end-to-end gradients in noisy, black-box multi-agent systems. We further introduce surrogate modeling to enhance scalability for large task pools. We validate our framework across benign and adversarial settings and multiple task formats. Under the same training budget, our method outperforms random training in downstream performance and stability, demonstrating more effective task selection. Under matched total budgets (selection + training), our method also outperforms random training, demonstrating greater overall cost efficiency. These gains further generalize from knowledge and reasoning to code generation, multi-step tool use, and long-horizon embodied interaction. Our key contributions are summarized as follows:

- First, we demonstrate a potential problem of communication-structure optimization for LLM-MAS: the choice of training tasks has a substantial impact on the optimized communication structure, especially under limited training budgets, which can lead to substantially degraded downstream accuracy of the multi-agent systems.
- Second, to our knowledge, we are the first to show that active learning is effective for mitigating this problem: compared with random task selection, it identifies more informative training tasks, improving downstream performance while achieving greater cost efficiency.
- Third, we propose an ensemble-based, derivative-free, and information-theoretic active learning approach for communication-structure optimization, and validate it across both benign and adversarial settings and across multiple task formats.

2 Empirical Motivation: Sensitivity to Training-Task Selection

Why does task selection matter? We first observe that when the communication structure is optimized using a randomly sampled set of training tasks, the resulting downstream accuracy exhibits substantial variation across independent runs. This instability indicates that communication-structure optimization can be sensitive to the particular training tasks selected.

A natural way to reduce this instability is to increase the number of training tasks. Figure 1 examines this strategy over an extended range of training budgets. Increasing the random-training budget initially provides some improvement, but the gain remains limited and does not consistently increase with additional cost. For reference, our active-learning configuration uses 0.80M tokens

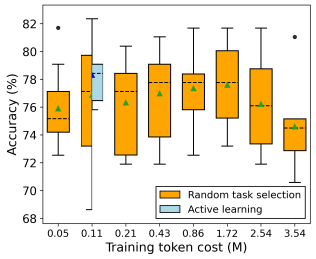

Figure 1: Accuracy-cost scaling on MMLU (adversarial). More randomly selected training tasks help only modestly, while active learning achieves higher accuracy despite lower cost.

for task selection and 0.12M for graph training, resulting in a total cost of 0.92M tokens. Notably, random training still underperforms active task selection even when its training cost substantially exceeds the total selection-and-training cost of active learning. Thus, over the evaluated range, simply spending more computation on randomly selected training tasks does not close the performance gap with active selection.

Accordingly, we formulate task selection as an active learning problem on top of communication-structure optimization and develop a method for identifying informative training tasks.

3 Related Work

Active learning strategies often rely on diversity-based, uncertainty-based, or information-theoretic criteria, with practical methods often combining these perspectives (Settles, 2009; Tharwat & Schenck, 2023). Among them, information-theoretic approaches are particularly appealing because they aim to select tasks according to the information they provide about model parameters (Kruschke, 2008). However, estimating information gain is often substantially more difficult than measuring diversity or predictive uncertainty (Kirsch et al., 2019; Ren et al., 2021).

Information-gain estimation can be approached in two broad ways. The first compares the probability distributions over parameters before and after observing a candidate task, while the second measures how much a deterministic quantity, such as a point estimate or a gradient-based update, would change. The distributional perspective usually requires approximating the Bayesian update induced by the candidate task, using Monte Carlo or other sample-based estimators (Settles et al., 2008; Siddhant & Lipton, 2018). Although these sample-based estimators expose uncertainty-related signals, they are often prohibitively expensive (Houlsby et al., 2011; Gal et al., 2017), especially in LLM-MAS, where scoring each candidate may require repeated multi-agent rollouts across multiple samples or parameter realizations. A cheaper alternative is to rely on gradient-based proxies for parameter updates (Cohn et al., 1994; Cai et al., 2013). In LLM-MAS, although such gradients may be approximated, the resulting estimators are often noisy and have large variance, making them a less reliable surrogate for information-theoretic utility. This creates a gap for practical active learning methods that efficiently approximate information gain while retaining uncertainty-related signals in black-box multi-agent settings.

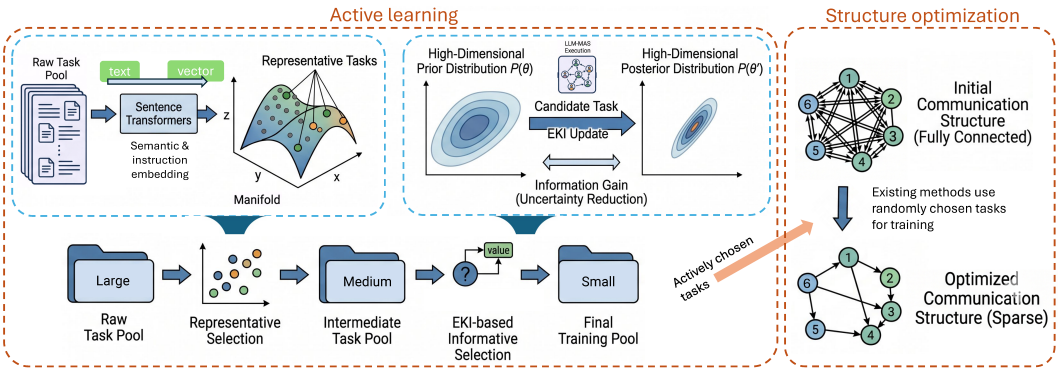

Figure 2: Overview of the proposed active learning framework. The framework first selects representative tasks from a large raw task pool, and then identifies the most informative tasks for communication-structure optimization via one-step EKI-based utility estimation. The selected tasks are finally used to optimize the communication structure.

4 Methodology

4.1 Problem Setup

Communication graph. We represent the MAS communication structure as a spatial-temporal graph and introduce trainable variables that parameterize edge activation. Specifically, we define $\mathcal{G} = (V, E^S, E^T)$, where $V = \{v_1, \ldots, v_N\}$ is the set of agents, and $E^S, E^T \in \{0,1\}^{N \times N}$ are the

spatial and temporal adjacency matrices describing communication within and across interaction rounds, respectively. An entry $E_{ij} = 1$ indicates that communication from agent i to agent j is allowed, while $E_{ij} = 0$ means that the corresponding edge is absent and cannot be activated.

Following a typical setup (Zhang et al., 2025b), we introduce trainable graph logits as the graph parameters to be optimized:

$$\mathbf{z} = \{Z^S, Z^T\}, \qquad Z^S, Z^T \in \mathbb{R}^{N\times N}.$$

Before each MAS rollout, a stochastic mask $A = \{A^S, A^T\}$ is sampled, with each entry following a Bernoulli distribution whose probability is given by the sigmoid of the corresponding logit. The mask is then applied element-wise to E^S and E^T, and the masked communication structure is used for the MAS rollout.

MAS rollout and forward map. Given a task q and graph logits $\mathbf{z}$, the MAS executes a rollout over the masked communication graph. Each agent receives the task together with messages from its spatial and temporal in-neighbors, and the system produces a final output g after several rounds of interaction. We represent the entire rollout as

$$g = G(q; \mathbf{z}, \omega), \tag{1}$$

where g is the MAS output, such as a multiple-choice answer, a numerical value, or generated code. The variable ω represents the stochasticity in the rollout, for example, from LLM sampling, multi-agent interaction, and final decision.

Communication-structure optimization (hereafter, graph optimization). Given a training dataset $\mathcal{D}_{\text{tr}} = \{(q_i, y_i)\}_{i=1}^{M}$, where q_i is the task to be solved and y_i is the ground truth or its abstract representation, let $l = \phi(g, q_i, y_i)$ denote the MAS performance score (e.g., answer correctness or code execution success) under the graph logits $\mathbf{z}$, computed by the evaluator $\phi(\cdot)$. The graph optimization problem is to find the graph logits that maximize the expected task performance score over the mask distribution:

$$\mathbf{z}^\star = \arg\max_{\mathbf{z}} \ \mathbb{E}_{A|\mathbf{z}} \mathbb{E}_{(q_i,y_i)\in\mathcal{D}_{\text{tr}}} \big\{\phi(g, q_i, y_i)\big\}, \tag{2}$$

where $\mathbb{E}_{A|\mathbf{z}}$ denotes the expectation over the stochastic masks induced by the logits $\mathbf{z}$. All other MAS components, including agent roles, prompts, tools, and the LLM backbone, remain fixed during optimization. The graph logits are optimized with REINFORCE-estimated gradients (Zhang et al., 2025b; Leong et al., 2025; Wang et al., 2025c), though other methods are applicable.

Active learning. In principle, one may optimize $\mathbf{z}$ using all available training tasks. However, in LLM-MAS, graph optimization is expensive because it requires repeated LLM agent inference. We therefore seek to select a small subset $\mathcal{S} \subsetneq \mathcal{D}_{\text{tr}}$, containing the tasks that are most informative for graph optimization, with the goal of maximizing downstream performance under a limited training budget. Given a task budget $B \ll M$, we formulate the selection problem as

$$\mathcal{S}^\star = \arg\max_{\mathcal{S}\subsetneq\mathcal{D}_{\text{tr}}, |\mathcal{S}|\le B} U(\mathcal{S}), \tag{3}$$

where the utility function $U(\mathcal{S})$ quantifies the usefulness of selected tasks for graph optimization.

Utility function. We measure the usefulness of a task set by information gain $I(\mathcal{S})$, which compares the prior and posterior distributions of the parameters (logits). This posterior-to-prior change is a standard information-theoretic measure of Bayesian information gain, widely used in Bayesian experimental design (Chaloner & Verdinelli, 1995) and Bayesian active learning (MacKay, 1992). The most common metric is the Kullback–Leibler divergence:

$$U(\mathcal{S}) = I(\mathcal{S}) = D_{\text{KL}}(p(\mathbf{z}|\mathcal{S})||p(\mathbf{z})), \tag{4}$$

where $p(\mathbf{z})$ is the prior distribution of the graph parameters, and $p(\mathbf{z}|\mathcal{S})$ is the posterior conditioned on the training task set $\mathcal{S}$. Other distributional metrics can also be applied, e.g., Wasserstein distances (Helin et al., 2025; Yang et al., 2026b). A larger $I(\mathcal{S})$ indicates a more substantial update to the graph parameters and thus greater parameter-level informativeness for graph optimization. Task difficulty, agent disagreement, or prediction diversity do not directly quantify this parameter-change information. Potential failure modes and corresponding mitigation strategies are discussed in Appendix E.1.

This formulation is related to expected information gain (EIG) in Bayesian experimental design, which is equivalent to BALD in Bayesian active learning (Houlsby et al., 2011). In standard EIG, the observation associated with a candidate experiment is unknown before the experiment is performed, so the information gain is averaged over possible observations. In our setting, the analogous observation is the reference answer to a candidate task, which is already available in the training pool. Therefore, no expectation over possible answers is required, and we directly evaluate the realized information gain. For unlabeled or generated tasks whose reference answers are unknown, an outer expectation over possible answers would recover the standard EIG or BALD.

The original graph optimization problem in Eq. (2) is defined on the deterministic variable $\mathbf{z}$ and can be solved by deterministic methods. However, to evaluate information gain by distribution difference in Eq. (4), $\mathbf{z}$ should be treated as a random variable and an update process from prior to posterior distribution of $\mathbf{z}$ is required. Standard methods for Bayesian updates from prior $p(\mathbf{z})$ to posterior $p(\mathbf{z}|\mathcal{S})$, such as MCMC and SMC, are expensive and computationally impractical for MAS. Therefore, a feasible approximation to Bayesian update is required.

4.2 EKI-Based Task Utility Estimation

We approximate the Bayesian update with ensemble Kalman inversion (EKI, (Iglesias et al., 2013; Kovachki & Stuart, 2019)). For notation and pedagogical clarity, we first formulate this update on one task at a time, that is, to estimate $p(\mathbf{z}|q, y)$ and $I(q)$. We then extend it to a multi-task setting for the batch active learning setup to estimate $I(\mathcal{S})$ in Section 4.3.

Ensemble Kalman inversion. In this work, EKI serves as an efficient approximation to the Bayesian update in Eq. (4) and has a deep connection to score-based posterior sampling (Song et al., 2020; Chung et al., 2022):

$$\nabla_{\mathbf{z}} \log p(\mathbf{z}|q, y) = \nabla_{\mathbf{z}} \log p(\mathbf{z}) + \nabla_{\mathbf{z}} \log p(y|\mathbf{z}, q), \tag{5}$$

where $\nabla_{\mathbf{z}} \log p(\mathbf{z}|q, y)$ is the score function of the posterior distribution that enables sampling methods such as classical Langevin dynamics (Roberts & Tweedie, 1996; Song & Ermon, 2019), or more recent score-based diffusion models (Song et al., 2020; Chung et al., 2022). Some more details of the connection between EKI and score-based posterior sampling are presented in Appendix A.1.

More specifically, EKI represents the uncertainty over the graph parameters using an ensemble of particles $\{\mathbf{z}^{(j)}\}_{j=1}^{J} \in \mathbb{R}^{d_z}$, which are subsequently updated using the task q and answer y.

For each ensemble particle $\mathbf{z}^{(j)}$, we perform a forward evaluation: we run the LLM-MAS on a given task q, obtain the output $g^{(j)}$, and compare it with the true answer y through evaluator $\phi(\cdot)$ to obtain the corresponding scalar performance score $l^{(j)} \in \mathbb{R}$ (Note this score is not restricted to be a scalar; see the multi-objective extension in Appendix A.5):

$$g^{(j)} = G(q; \mathbf{z}^{(j)}, \omega), \quad l^{(j)} = \phi(g^{(j)}, q, y). \tag{6}$$

Together, these forward evaluations form the EKI prediction step. Let $\bar{\mathbf{z}} = \frac{1}{J}\sum_{j=1}^{J} \mathbf{z}^{(j)}$, $\bar{l} = \frac{1}{J}\sum_{j=1}^{J} l^{(j)}$ denote the empirical means of the parameter ensemble and the predicted scores. The empirical cross-covariance and score covariance are then estimated as

$$C^{zl} = \frac{1}{J-1}\sum_{j=1}^{J}\big(\mathbf{z}^{(j)} - \bar{\mathbf{z}}\big)\big(l^{(j)} - \bar{l}\big)^{\top}, \quad C^{ll} = \frac{1}{J-1}\sum_{j=1}^{J}\big(l^{(j)} - \bar{l}\big)\big(l^{(j)} - \bar{l}\big)^{\top}. \tag{7}$$

In the update step, EKI updates each ensemble particle by

$$\mathbf{z}_{\text{post}}^{(j)} = \mathbf{z}^{(j)} + C^{zl}\big(C^{ll} + \Gamma\big)^{-1}\big(l^{*} - l^{(j)}\big), \tag{8}$$

where l^{*} is the desired value for the performance score, e.g., 1 for binary correctness. Γ is the score noise covariance, which is a representation of the forward stochasticity ω. In practice, we use it primarily to regularize the update for numerical stability, rather than as a precisely calibrated noise covariance. The updated ensemble $\{\mathbf{z}_{\text{post}}^{(j)}\}_{j=1}^{J}$ provides an empirical approximation to the posterior distribution $p(\mathbf{z}|q, y)$ (Yang et al., 2026a). To save computational cost, we only perform a one-step

EKI update in this work, which has been shown to be effective for distinguishing different tasks (Yang et al., 2026a; Callahan et al., 2025); we further validate this choice in Appendix D.5.2.

Ensemble-based utility approximation. Given the initial and updated ensembles, the information gain of a single task is approximated by

$$I(q) = D_{\mathrm{KL}}(p(\mathbf{z}|q,y)||p(\mathbf{z})) \approx D_{\mathrm{KL}}(\{\mathbf{z}_{\mathrm{post}}^{(j)}\}_{j=1}^{J}||\{\mathbf{z}^{(j)}\}_{j=1}^{J}). \tag{9}$$

In practice, we employ Gaussian approximations to the ensembles and compute the KL using their empirical means and variances. We then rank candidate tasks by their estimated information gain.

Remark. The EKI update is used only for utility estimation. After task selection, the selected tasks are used to optimize the same graph parameters, implemented with REINFORCE in this work. We further examine how the EKI-based formulation scales with MAS size in Appendix D.1.1.

4.3 Extension to batch setting

Batch formula. The single-task formula can be extended to batch setting for the batch gain $I(\mathcal{S})$ (Kirsch et al., 2019). Given an ensemble particle $\mathbf{z}^{(j)}$ and a set $\mathcal{S} = \{(q_i, y_i)\}_{i=1}^{m} \subseteq \mathcal{D}_{\mathrm{tr}}$ with multiple tasks and answers, the corresponding scores $\{l_i^{(j)} = \phi(g_i^{(j)}, q_i, y_i) \in \mathbb{R}\}_{i=1}^{m}$ for all tasks can be obtained. To keep the updating formula consistent with Eqs. (7) and (8), we replace the scalar score with a vector score $\mathbf{l}^{(j)} = [l_1^{(j)}, l_2^{(j)}, \cdots, l_m^{(j)}]^\top \in \mathbb{R}^m$. The corresponding updated ensemble $\{\mathbf{z}_{\mathrm{post}}^{(j)}\}_{j=1}^{J}$ then represents the posterior updated on all the tasks in $\mathcal{S}$ and the batch information gain $I(\mathcal{S})$ is obtained.

Cost of batch evaluation. For a batch $\mathcal{S}$ of size m, evaluating $I(\mathcal{S})$ requires Jm forward evaluations, since each task is evaluated for each ensemble particle. This is of the same order as computing the individual gains $I(q)$ for all tasks in the batch separately. The batch formulation only changes the dimension of the update step, where the covariance and update are computed from the vector-valued scores. In practice, this additional update cost is negligible compared with the forward evaluations.

Reuse strategy. A reuse strategy reduces the total forward evaluations when evaluating overlapping candidate subsets under the batch setting. For example, consider two candidate subsets $\mathcal{S}_1 = \{q_1, q_2\}$ and $\mathcal{S}_2 = \{q_2, q_3\}$. If the two subset gains are evaluated independently, the prediction step requires $4J$ forward evaluations. However, *if both subsets are evaluated from the same initial ensemble*, the prediction results for the shared task q_2 can be reused. Thus, only the three unique tasks $\{q_1, q_2, q_3\}$ need to be evaluated, giving $3J$ forward evaluations. More generally, for a collection of candidate subsets $\{\mathcal{S}_b \subseteq \mathcal{D}_{\mathrm{tr}}\}_{b=1}^{B}$, the prediction-step cost is reduced from $J\sum_{b=1}^{B}|\mathcal{S}_b|$ to $J\left|\bigcup_{b=1}^{B}\mathcal{S}_b\right|$.

After the scores of each task are cached, each subset gain is obtained by selecting the corresponding cached scores for the tasks within this subset, stacking them into a batch-level vector, and applying the EKI update step. This reuse strategy avoids repeated forward evaluations for the same task whenever it appears in multiple candidate subsets, which can substantially reduce the dominant cost of batch utility estimation. Appendix A.4 provides the detailed formulation and discussion.

4.4 Active learning procedure

Given a way to quantify the information gain, we can enumerate all candidate tasks, evaluate their information gain, and choose either the top-k most informative tasks or form a coreset to optimize the graph. However, for a large candidate pool, enumeration is not feasible. To address this issue and develop a practical active learning algorithm, we introduce a two-stage selection procedure to shrink the candidate pool for information-gain-based selection, and further introduce surrogate modeling to reduce the number of forward evaluations for information gain. An overall schematic figure is shown in Fig. 2 and the algorithm is given in Appendix A.2.

We focus on a single-round setting, where task selection and subsequent graph optimization are each performed once, without alternating. A multi-round setting can be achieved by repeating this procedure with the updated graph, but is not considered here.

Two-stage selection. From a large pool, we first select the most representative tasks to form a smaller intermediate pool, from which we then choose the most informative tasks according to

their information gain (we call this informative selection). This design allows the inexpensive representative selection to screen a large task pool, while the more expensive informative selection is performed only on a much smaller set. On this reduced pool, either sequential decision-making or enumeration becomes feasible. For extremely large or even infinite datasets, we first randomly sample a candidate pool of manageable size and then apply the same two-stage selection procedure.

Representative selection. We embed the tasks q_i from text into vectors v_i such that their relative positions in the dataset manifold can be identified. The most intuitive approach is to directly apply a text encoder based on sentence transformers (Reimers & Gurevych, 2019; Wang et al., 2020). Recent work has proposed instruction embeddings to enhance this process (Li et al., 2024c). For tasks with labels that distinguish them into several groups within the whole dataset, e.g., subjects in MMLU (Hendrycks et al., 2021), these labels can also be included in the embedding process to specify different manifolds for each group. For tasks without explicit labels, e.g., GSM8K (Cobbe et al., 2021), we simply input the whole task into a single encoder without further modification.

Once the embedding vectors are obtained, we employ a greedy approach to form the intermediate pool. We first choose the center point, and then sequentially choose the farthest point from the selected points until the budget is reached.

Surrogate modeling for informative selection. To avoid enumerating all candidates in the representative pool, we apply a surrogate modeling approach. We denote the map from a task embedding vector $v(q)$ to its information gain $U(q)$ as $\psi(\cdot)$, $\psi : v(q) \mapsto U(q)$. Specifically, for a given task q, we perform EKI to update prior knowledge of the graph $\mathbf{z}$ to a posterior and quantify their difference as the information gain $U(q)$. After several pairs $\{q_k, U_k\}_{k=1}^{K}$ are collected, with K much smaller than the pool size, we build a surrogate model $\bar{\psi}(\cdot)$ to approximate the true map $\psi(\cdot)$. We use the surrogate model to predict the information gain of unevaluated tasks and choose the most informative tasks for actual evaluation (see details in Appendix B.5). These evaluations in turn help improve the surrogate model. This surrogate-based sequential strategy substantially reduces the number of tasks requiring actual evaluation.

5 Experiments

5.1 Experimental setup

We ask whether active task selection improves communication-structure optimization over random selection. We first evaluate this on general-knowledge and mathematical-reasoning tasks under both benign and adversarial settings, where we compare the two methods under both matched training budgets and comparable total budgets to assess performance and cost efficiency. We then extend the evaluation to code generation, multi-step tool use, and long-horizon embodied interaction to examine generalization across broader benchmarks.

In the benign setting, all agents follow their assigned roles; in the adversarial setting, a subset is instructed to provide incorrect responses and mislead the other agents, and no agent is informed whether any other agent is adversarial (Appendix C). We evaluate downstream performance and token cost, reporting the mean, run-to-run standard deviation, and lower-tail statistics (Q1 and worst-25% mean) across independent runs. More experimental details can be found in Appendix B.

5.2 Performance on knowledge and reasoning benchmarks

We first evaluate active task selection on MMLU and GSM8K under both benign and adversarial settings. Both benchmarks use six GPT-4.1-nano agents; in the adversarial setting, three of the six agents are adversarial. Additional experiments varying the number of agents and the LLM backbone are reported in Appendix D.1.

Under the same training budget, active selection improves mean accuracy, run-to-run stability, and lower-tail performance over random selection consistently in all four settings (Table 1). A representative run-to-run accuracy distribution is shown in Fig. 3. Together, they indicate that the

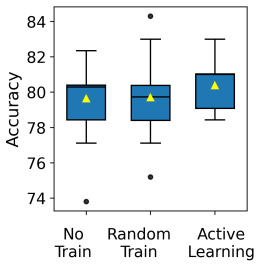

Figure 3: Distribution of downstream accuracy on MMLU (benign)

selected tasks are more effective for communication-structure optimization and therefore lead to better downstream performance.

The improvement is not simply obtained by paying an additional selection cost: under a comparable total budget, active selection still outperforms random training on both adversarial benchmarks. From the complementary perspective, active selection can achieve a given level of downstream performance with a smaller total budget than random training, further demonstrating its overall cost efficiency.

Table 1: Performance comparison on MMLU and GSM8K under benign and adversarial settings. Accuracy is reported as mean ± std, first quartile (Q1), and mean below Q1. Token cost is reported as mean. Values in parentheses are computed relative to Random, not Random†.

Dataset	Setup	Accuracy / (%)			Token cost / (M)		
		Mean ± std	Q1	Worst-25%	Select	Train	Test
MMLU (Benign)	No train	79.64 ± 1.96	78.43	77.23	-	-	1.23
	Random	79.69 ± 2.19	78.40	77.26	-	0.16	0.99
	Active	80.38 ± 1.35	**79.08** (+0.68)	**78.56** (+1.30)	1.15	0.16	1.00
GSM8K (Benign)	No train	92.27 ± 0.54	91.73	91.61	-	-	4.50
	Random	92.78 ± 0.54	92.49	92.06	-	0.17	3.56
	Active	93.07 ± 0.28	**92.82** (+0.33)	**92.77** (+0.71)	2.10	0.18	3.54
MMLU (Adversarial)	No train	75.68 ± 2.10	75.16	73.41	-	-	1.04
	Random	76.87 ± 3.72	73.20	72.00	-	0.11	0.67
	Random†	77.35 ± 2.64	75.82	73.85	-	0.86	0.71
	Active	78.32 ± 1.80	**76.47** (+3.27)	**76.25** (+4.25)	0.80	0.12	0.67
GSM8K (Adversarial)	No train	90.97 ± 1.32	89.83	89.56	-	-	3.91
	Random	92.90 ± 1.00	92.17	91.69	-	0.13	2.47
	Random†	92.64 ± 0.71	92.11	91.73	-	1.98	2.49
	Active	93.83 ± 0.76	**93.15** (+0.98)	**92.93** (+1.24)	1.83	0.13	2.36

† Random training with a budget comparable to the total selection and training cost of active learning.

5.3 Generalization to broader agentic benchmarks

To evaluate whether active task selection generalizes beyond single-response QA, we further consider HumanEval for code generation (Chen et al., 2021), InterCode-SQL for multi-step tool use (Yang et al., 2023), and ALFWorld for long-horizon embodied interaction (Shridhar et al., 2021). All three experiments use adversarial agent configurations, with different numbers of total agents (5/5/4) and adversarial agents (3/1/2) across benchmarks.

For the two multi-step benchmarks, Success@k measures the fraction of tasks solved within k interaction steps, while AUC@k averages success over the interaction horizon and therefore also reflects how early a task is solved.

Across all three benchmarks, active task selection shows improvement over random training on nearly all reported metrics (Fig. 4; Appendix D.2.3). Taken together, these results provide evidence that our active task-selection method remains effective across substantially different task formats and agent–environment interaction forms. More broadly, our method naturally accommodates diverse task formats because the ensemble-based formulation treats MAS rollouts as black-box evaluations.

5.4 Comparison with other informativeness-based active learning methods

We compare our method with several informativeness-based active learning baselines on MMLU under agent attacks. All methods share the same 1000-to-50 representative-selection stage and differ only in how they select 10 tasks from the 50-task pool.

We consider expected gradient length (EGL) and expected model change (EMC), which quantify informativeness through the magnitude of parameter changes induced by a task. We also include Fisher-information-based coreset-style baseline with both trace and determinant criteria, which are related to A-optimal and D-optimal experimental designs, respectively. See details in Appendix D.6.

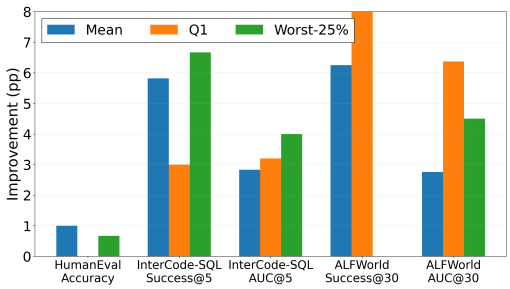

Figure 4: Improvement of active task selection over random training across benchmarks with different task formats.

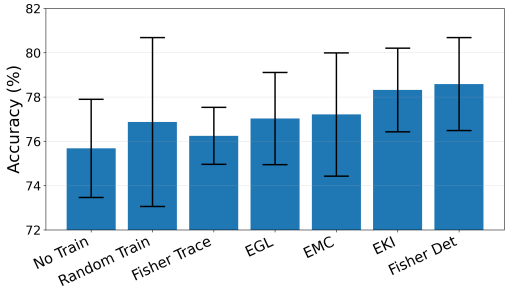

Figure 5: Comparison among methods. EKI and Fisher coreset (det) perform best overall, while EKI does not use a pool-level coreset.

Under the same selection cost, EKI and Fisher coreset (det) achieve higher mean downstream accuracy than EGL, EMC, and Fisher coreset (trace) (Fig. 5; see Appendix D.6.5 for statistics). EKI achieves accuracy comparable to Fisher coreset (det) without estimating graph-parameter gradients or explicitly constructing a pool-level coreset. See Appendix D.6.6 for differences in selection behavior and computational cost.

5.5 Ablation study and sensitivity analysis

We provide an ablation study in Appendix D.7 and a sensitivity analysis in Appendix D.5. The ablation results suggest that informative selection accounts for most of the performance gain. Although representative selection provides a smaller additional improvement, it is more valuable for reducing the candidate pool. The sensitivity results show that the proposed task selection is stable under small ensemble sizes and a small number of EKI iterations. This further suggests that small-scale EKI is sufficient to identify informative tasks, thereby making the proposed utility estimation efficient.

6 Conclusion, limitations, and future work

We first show that communication-structure optimization is highly sensitive to the choice of training tasks. We then show that active task selection improves downstream performance and stability, achieves better overall cost efficiency, and generalizes across diverse benchmarks. We develop an efficient and derivative-free active learning framework for this purpose, estimating task information gain through an ensemble-based method. Our formulation is particularly suitable for black-box MAS in which direct information gain estimation is difficult. These results support treating task selection as an explicit optimization problem, rather than relying on randomly sampled training tasks.

limitations. First, although the sensitivity study supports the reliability of small-scale EKI within the budgets considered, its reliability in more extreme regimes, such as substantially smaller ensembles relative to the parameter dimension or more complex posteriors, remains to be characterized. Second, more advanced choices for the embedding, surrogate modeling, and broader active learning pipeline may further improve performance. Third, our current setting assumes that both tasks and answers are already available, so selection is performed over task-answer pairs. When true answers are not given in advance, they must be acquired through additional queries, making the expected information gain a more appropriate formulation. **Future work** will study EKI reliability in extreme regimes, unlabeled-task settings, and task generation.

Summary of contents in the appendix. First, we show the connection between EKI and score-based posterior sampling (see Appendix A.1). Second, method-related details are provided, including the algorithm (A.2), notation summary (A.3), prediction-step reuse for batch utility evaluation (A.4), and multi-objective utility estimation (A.5). Then, the experimental setup (B) and MAS details (C) are provided. Additional results are reported, including scalability (D.1), task examples and EKI diagnostics (D.2), statistical analyses (D.3, D.4), sensitivity studies (D.5), other active learning methods (D.6), and ablation studies (D.7). Finally, Appendix E discusses failure modes and mitigations, scope, and generalizability.

AI USE STATEMENT

We primarily used generative AI tools to improve the readability and polish the language of the manuscript. We also used these tools to assist with the implementation of baseline methods in our experiments and the qualitative characterization of illustrative task examples reported in Table 7.

REPRODUCIBILITY STATEMENT

Experimental details and parameter settings are provided in Appendix B and C to facilitate reproduction of our results. The code used in this work will be released upon acceptance.

REFERENCES

Zhenyu Bi, Meng Lu, Yang Li, Swastik Roy, Weijie Guan, Morteza Ziyadi, and Xuan Wang. OPTAGENT: Optimizing multi-agent LLM interactions through verbal reinforcement learning for enhanced reasoning. In *Proceedings of the 14th International Joint Conference on Natural Language Processing and the 4th Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pp. 1713–1728, 2025.

Wenbin Cai, Ya Zhang, and Jun Zhou. Maximizing expected model change for active learning in regression. In *2013 IEEE 13th international conference on data mining*, pp. 51–60. IEEE, 2013.

Jake Callahan, Andrew Chin, Jason Pacheco, and Tommie Catanach. Reverse-annealed sequential Monte Carlo for efficient Bayesian optimal experiment design. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=jut5q3UYRz.

Kathryn Chaloner and Isabella Verdinelli. Bayesian experimental design: A review. *Statistical science*, 10(3):273–304, 1995.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Hyungjin Chung, Jeongsol Kim, Michael T Mccann, Marc L Klasky, and Jong Chul Ye. Diffusion posterior sampling for general noisy inverse problems. *arXiv preprint arXiv:2209.14687*, 2022.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

David Cohn, Les Atlas, and Richard Ladner. Improving generalization with active learning. *Machine learning*, 15(2):201–221, 1994.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep Bayesian active learning with image data. In *International conference on machine learning*, pp. 1183–1192. PMLR, 2017.

Saarth Gaonkar, Xiang Zheng, Haocheng Xi, Rishabh Tiwari, Kurt Keutzer, Dmitriy Morozov, Michael W Mahoney, and Amir Gholami. SciML agents: Write the solver, not the solution. *arXiv preprint arXiv:2509.09936*, 2025.

Junda He, Christoph Treude, and David Lo. LLM-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Transactions on Software Engineering and Methodology*, 34(5):1–30, 2025.

Tapio Helin, Youssef Marzouk, and Jose Rodrigo Rojo-Garcia. Bayesian optimal experimental design with Wasserstein information criteria. *arXiv preprint arXiv:2504.10092*, 2025.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021.

Sirui Hong, Mingchen Zhuge, Jonathan Chen, Xiawu Zheng, Yuheng Cheng, Jinlin Wang, Ceyao Zhang, Zili Wang, Steven Ka Shing Yau, Zijuan Lin, et al. MetaGPT: Meta programming for a multi-agent collaborative framework. In *The twelfth international conference on learning representations*, 2024.

Neil Houlsby, Ferenc Huszár, Zoubin Ghahramani, and Máté Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

Marco A Iglesias, Kody JH Law, and Andrew M Stuart. Ensemble Kalman methods for inverse problems. *Inverse Problems*, 29(4):045001, 2013.

Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. Batchbald: Efficient and diverse batch acquisition for deep Bayesian active learning. *Advances in neural information processing systems*, 32, 2019.

Nikola B Kovachki and Andrew M Stuart. Ensemble Kalman inversion: a derivative-free technique for machine learning tasks. *Inverse Problems*, 35(9):095005, 2019.

John K Kruschke. Bayesian approaches to associative learning: From passive to active learning. *Learning & behavior*, 36(3):210–226, 2008.

Hui Yi Leong, Yuheng Li, Yuqing Wu, Wenwen Ouyang, Wei Zhu, Jiechao Gao, and Wei Han. Amas: Adaptively determining communication topology for LLM-based multi-agent system. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing: Industry Track*, pp. 2061–2070, 2025.

Junkai Li, Yunghwei Lai, Weitao Li, Jingyi Ren, Meng Zhang, Xinhui Kang, Siyu Wang, Peng Li, Ya-Qin Zhang, Weizhi Ma, et al. Agent hospital: A simulacrum of hospital with evolvable medical agents. *arXiv preprint arXiv:2405.02957*, 2024a.

Shiyuan Li, Yixin Liu, Qingsong Wen, Chengqi Zhang, and Shirui Pan. Assemble your crew: Automatic multi-agent communication topology design via autoregressive graph generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 40, pp. 23142–23150, 2026.

Xinyi Li, Sai Wang, Siqi Zeng, Yu Wu, and Yi Yang. A survey on LLM-based multi-agent systems: workflow, infrastructure, and challenges. *Vicinagearth*, 1(1):9, 2024b.

Yiwei Li, Jiayi Shi, Shaoxiong Feng, Peiwen Yuan, Xinglin Wang, Boyuan Pan, Heda Wang, Yao Hu, and Kan Li. Instruction embedding: Latent representations of instructions towards task identification. *Advances in Neural Information Processing Systems*, 37:87683–87711, 2024c.

David JC MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

Chen Qian, Wei Liu, Hongzhang Liu, Nuo Chen, Yufan Dang, Jiahao Li, Cheng Yang, Weize Chen, Yusheng Su, Xin Cong, et al. Chatdev: Communicative agents for software development. In *Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: Long papers)*, pp. 15174–15186, 2024.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, pp. 3982–3992. Association for Computational Linguistics, 11 2019.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM computing surveys (CSUR)*, 54(9):1–40, 2021.

Gareth O Roberts and Richard L Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. 1996.

Burr Settles. Active learning literature survey. (1648), 2009.

Burr Settles, Mark Craven, and Lewis Friedland. Active learning with real annotation costs. In *Proceedings of the NIPS workshop on cost-sensitive learning*, volume 1. Vancouver, CA:, 2008.

Xu Shen, Yixin Liu, Yiwei Dai, Yili Wang, Rui Miao, Yue Tan, Shirui Pan, and Xin Wang. Understanding the information propagation effects of communication topologies in LLM-based multi-agent systems. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pp. 12347–12361, 2025.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. ALFWorld: Aligning Text and Embodied Environments for Interactive Learning. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021. URL https://arxiv.org/abs/2010.03768.

Aditya Siddhant and Zachary C Lipton. Deep Bayesian active learning for natural language processing: Results of a large-scale empirical study. In *Proceedings of the 2018 conference on empirical methods in natural language processing*, pp. 2904–2909, 2018.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. *Advances in neural information processing systems*, 32, 2019.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. *arXiv preprint arXiv:2011.13456*, 2020.

Alaa Tharwat and Wolfram Schenck. A survey on active learning: State-of-the-art, practical challenges and research directions. *Mathematics*, 11(4):820, 2023.

Jingquan Wang, Harry Zhang, Khailanii Slaton, Shu Wang, Radu Serban, Jinlong Wu, and Dan Negrut. Chronollm: A framework for customizing large language model for digital twins generalization based on pychrono. *arXiv preprint arXiv:2501.04062*, 2025a.

Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. MiniLM: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in neural information processing systems*, 33:5776–5788, 2020.

Wenxuan Wang, Zizhan Ma, Zheng Wang, Chenghan Wu, Jiaming Ji, Wenting Chen, Xiang Li, and Yixuan Yuan. A survey of LLM-based agents in medicine: How far are we from baymax? *Findings of the Association for Computational Linguistics: ACL 2025*, pp. 10345–10359, 2025b.

Zhexuan Wang, Yutong Wang, Xuebo Liu, Liang Ding, Miao Zhang, Jie Liu, and Min Zhang. Agentdropout: Dynamic agent elimination for token-efficient and high-performance LLM-based multi-agent collaboration. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 24013–24035, 2025c.

Huchen Yang, Xinghao Dong, and Jin-Long Wu. Bayesian experimental design for model discrepancy calibration: An auto-differentiable ensemble Kalman inversion approach. *Journal of Computational Physics*, 545:114469, 2026a.

Huchen Yang, Xinghao Dong, and Jin-Long Wu. Bayesian experimental design for model discrepancy calibration: A rivalry between Kullback–Leibler divergence and Wasserstein distance. *Journal of Computational Physics*, 567:115295, 2026b. ISSN 0021-9991. doi: https://doi.org/10.1016/j.jcp.2026.115295.

Jiaxi Yang, Mengqi Zhang, Yiqiao Jin, Hao Chen, Qingsong Wen, Lu Lin, Yi He, Srijan Kumar, Weijie Xu, James Evans, et al. Topological structure learning should be a research priority for LLM-based multi-agent systems. *arXiv preprint arXiv:2505.22467*, 2025.

John Yang, Akshara Prabhakar, Karthik Narasimhan, and Shunyu Yao. Intercode: Standardizing and benchmarking interactive coding with execution feedback, 2023.

Yuting Zeng, Weizhe Huang, Lei Jiang, Tongxuan Liu, Xitai Jin, Chen Tianying Tiana, Jing Li, and Xiaohua Xu. S2-mad: Breaking the token barrier to enhance multi-agent debate efficiency. In *Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 9393–9408, 2025.

Guibin Zhang, Yanwei Yue, Xiangguo Sun, Guancheng Wan, Miao Yu, Junfeng Fang, Kun Wang, Tianlong Chen, and Dawei Cheng. G-designer: Architecting multi-agent communication topologies via graph neural networks. *arXiv preprint arXiv:2410.11782*, 2024a.

Guibin Zhang, Luyang Niu, Junfeng Fang, Kun Wang, Lei Bai, and Xiang Wang. Multi-agent architecture search via agentic supernet. *arXiv preprint arXiv:2502.04180*, 2025a.

Guibin Zhang, Yanwei Yue, Zhixun Li, Sukwon Yun, Guancheng Wan, Kun Wang, Dawei Cheng, Jeffrey Xu Yu, and Tianlong Chen. Cut the crap: An economical communication pipeline for LLM-based multi-agent systems. In *The Thirteenth International Conference on Learning Representations*, 2025b. URL https://openreview.net/forum?id=LkzuPorQ5L.

Jiayi Zhang, Jinyu Xiang, Zhaoyang Yu, Fengwei Teng, Xionghui Chen, Jiaqi Chen, Mingchen Zhuge, Xin Cheng, Sirui Hong, Jinlin Wang, et al. Aflow: Automating agentic workflow generation. *arXiv preprint arXiv:2410.10762*, 2024b.

A METHODOLOGY DETAILS AND EXTENSIONS

A.1 EKI AND SCORE-BASED POSTERIOR SAMPLING

In this appendix, we share some insights into the connection between the EKI update used in this work and score-based posterior sampling. We also discuss the limitations of EKI, which can guide future studies.

Note that, in the main text, we write the Bayesian update abstractly as $p(\mathbf{z} \mid q, y)$, without specifying how the compatibility between the MAS output g and the reference answer y is represented. In the EKI implementation, this relationship is made explicit through the performance score $l = \phi(g, q, y)$ and the desired performance value l^*. We therefore express the posterior update below as $p(\mathbf{z}|l^*)$ in the generic derivation, or as $p(\mathbf{z}|l^*, q)$ when the dependence on a specific task q is made explicit.

A.1.1 SCORE-BASED POSTERIOR SAMPLING

The score-based posterior sampling targets the Bayesian update:

$$p(\mathbf{z}|l^*) \propto p(\mathbf{z})p(l^*|\mathbf{z}), \tag{10}$$

which is handled in the form of score functions to avoid explicitly accounting for the computationally expensive normalization constant in the above Bayesian update:

$$\nabla_{\mathbf{z}} \log p(\mathbf{z}|l^*) = \nabla_{\mathbf{z}} \log p(\mathbf{z}) + \nabla_{\mathbf{z}} \log p(l^*|\mathbf{z}). \tag{11}$$

By working with the score function of the posterior distribution, sampling can be achieved through either classical Langevin dynamics Roberts & Tweedie (1996); Song & Ermon (2019) or more recent score-based diffusion models Song et al. (2020); Chung et al. (2022). It is worth noting that diffusion models rely on a series of score functions that correspond to the original one with different levels of noise injected, which has been demonstrated to be more robust than merely relying on the targeted score function.

A.1.2 CONNECTION BETWEEN SCORE-BASED POSTERIOR SAMPLING AND EKI

Observation model. For a task q, let $\mathcal{H}_q(\mathbf{z})$ denote the forward model associated with the scalar performance score l. In the notation of the main text, $\mathcal{H}_q(\mathbf{z})$ is a combination of ϕ and G. Consider the Gaussian observation model

$$l = \mathcal{H}_q(\mathbf{z}) + \eta, \qquad \eta \sim \mathcal{N}(0, \Gamma), \tag{12}$$

where Γ is the observation noise covariance in score space. Following Eq. (12), given a desired value of performance score l^* (the observation), the likelihood is

$$p(l^*|\mathbf{z}, q) \propto \exp\left(-\frac{1}{2}\big(l^* - \mathcal{H}_q(\mathbf{z})\big)^\top \Gamma^{-1} \big(l^* - \mathcal{H}_q(\mathbf{z})\big)\right). \tag{13}$$

Likelihood-score. If $\mathcal{H}_q$ is differentiable, then

$$\nabla_{\mathbf{z}} \log p(l^*|\mathbf{z}, q) = J_q(\mathbf{z})^\top \Gamma^{-1} \big(l^* - \mathcal{H}_q(\mathbf{z})\big), \tag{14}$$

where

$$J_q(\mathbf{z}) = \nabla_{\mathbf{z}} \mathcal{H}_q(\mathbf{z}) \tag{15}$$

is the Jacobian of the forward model. Combining the likelihood score with the prior score gives

$$\nabla_{\mathbf{z}} \log p(\mathbf{z}|q, l^*) = \nabla_{\mathbf{z}} \log p(\mathbf{z}) + J_q(\mathbf{z})^\top \Gamma^{-1} \big(l^* - \mathcal{H}_q(\mathbf{z})\big). \tag{16}$$

Local linear-Gaussian approximation induced by the ensemble. Assume the initial ensemble is a local Gaussian prior,

$$\mathbf{z} \sim \mathcal{N}(\bar{\mathbf{z}}, C^{zz}), \quad \bar{\mathbf{z}} = \frac{1}{J} \sum_{j=1}^{J} \mathbf{z}^{(j)}. \tag{17}$$

Around the initial ensemble mean $\bar{\mathbf{z}}$, we approximate the forward model by

$$\mathcal{H}_q(\mathbf{z}) \approx \mathcal{H}_q(\bar{\mathbf{z}}) + G_q(\mathbf{z} - \bar{\mathbf{z}}), \qquad G_q = J_q(\bar{\mathbf{z}}). \tag{18}$$

Under this local linear-Gaussian model, we define a local posterior covariance as

$$C_{\text{loc}} = \left[(C^{zz})^{-1} + G_q^\top \Gamma^{-1} G_q\right]^{-1}, \tag{19}$$

in order to write the Kalman gain matrix in an explicit form involving $G_q^\top \Gamma^{-1}$

$$K_q = C^{zl}(C^{ll} + \Gamma)^{-1} \approx C^{zz} G_q^\top \left(G_q C^{zz} G_q^\top + \Gamma\right)^{-1} = C_{\text{loc}} G_q^\top \Gamma^{-1}. \tag{20}$$

The detailed derivation of Eq. (20) is summarized in Appendix A.1.3. Then the EKI update used in the main text is

$$\mathbf{z}_{\text{post}}^{(j)} = \mathbf{z}^{(j)} + C_{\text{loc}} G_q^\top \Gamma^{-1}\left(l^* - \mathcal{H}_q(\mathbf{z}^{(j)})\right), \tag{21}$$

The second term on the right-hand side is similar to the second term in Eq. (16), which can be shown more clearly as below:

$$\begin{aligned} C_{\text{loc}} G_q^\top \Gamma^{-1}\left(l^* - \mathcal{H}_q(\mathbf{z}^{(j)})\right) &\approx C_{\text{loc}}\left[J_q(\mathbf{z})^\top \Gamma^{-1}\left(l^* - \mathcal{H}_q(\mathbf{z})\right)\right] \\ &= C_{\text{loc}}\left[\nabla_{\mathbf{z}} \log p(l^*|\mathbf{z}, q)\right]_{\mathbf{z}=\mathbf{z}^{(j)}}. \end{aligned} \tag{22}$$

The equality is obtained if the forward model is linear, i.e., $J_q = G_q$. Equation (22) reveals a connection between EKI and score-based posterior transport – the update formula in EKI is equivalent to applying the local posterior covariance C_{loc} as a preconditioner to the likelihood score at each ensemble particle.

To summarize, EKI is computationally efficient but naturally limited by the covariance structure represented by the ensemble, whereas score-based posterior sampling is more expressive for non-Gaussian or multimodal posteriors at the cost of repeated score evaluations during sampling.

A.1.3 Detailed derivation of Eq. (20)

1. Justification of the first approximation

Consider

$$\begin{aligned} C^{zl} &= \frac{1}{J-1}\sum_{j=1}^{J}(\mathbf{z}^{(j)} - \bar{\mathbf{z}})(l^{(j)} - \bar{l})^\top \\ &\approx \frac{1}{J-1}\sum_{j=1}^{J}(\mathbf{z}^{(j)} - \bar{\mathbf{z}})\left[G_q(\mathbf{z}^{(j)} - \bar{\mathbf{z}})\right]^\top \\ &= \frac{1}{J-1}\sum_{j=1}^{J}(\mathbf{z}^{(j)} - \bar{\mathbf{z}})(\mathbf{z}^{(j)} - \bar{\mathbf{z}})^\top G_q^\top \\ &= C^{zz} G_q^\top. \end{aligned} \tag{23}$$

Similarly, it can be shown that $C^{ll} \approx G_q C^{zz} G_q^\top$. Therefore, the first approximation holds, and the approximation will be exact when the forward model is linear.

2. Proof of the final equality

Here we show

$$C^{zz} G_q^\top \left(G_q C^{zz} G_q^\top + \Gamma\right)^{-1} = C_{\text{loc}} G_q^\top \Gamma^{-1}. \tag{24}$$

We first define $S = G_q C^{zz} G_q^\top + \Gamma$ for notational simplicity. Starting from the right-hand side,

$$\begin{aligned} C_{\text{loc}} G_q^\top \Gamma^{-1} &= C_{\text{loc}} G_q^\top \Gamma^{-1} S S^{-1} \\ &= C_{\text{loc}} G_q^\top \Gamma^{-1}(G_q C^{zz} G_q^\top + \Gamma) S^{-1} \\ &= C_{\text{loc}}\left[G_q^\top \Gamma^{-1} G_q C^{zz} G_q^\top S^{-1} + G_q^\top \Gamma^{-1} \Gamma S^{-1}\right] \\ &= C_{\text{loc}}\left[G_q^\top \Gamma^{-1} G_q C^{zz} G_q^\top S^{-1} + G_q^\top S^{-1}\right] \\ &= C_{\text{loc}}\left[G_q^\top \Gamma^{-1} G_q C^{zz} G_q^\top S^{-1} + (C^{zz})^{-1} C^{zz} G_q^\top S^{-1}\right] \\ &= C_{\text{loc}}\left[G_q^\top \Gamma^{-1} G_q + (C^{zz})^{-1}\right] C^{zz} G_q^\top S^{-1} \\ &= C_{\text{loc}} C_{\text{loc}}^{-1} C^{zz} G_q^\top S^{-1} \\ &= C^{zz} G_q^\top S^{-1}. \end{aligned} \tag{25}$$

A.1.4 LIMITATIONS OF EKI

In this work, we use EKI as an ensemble-based method to efficiently approximate the Bayesian update involved in the evaluation of information gain. It is much more efficient than diffusion models and other variants of stochastic interpolants, such as flow matching, since it avoids solving a multi-step reverse SDE/ODE, each step of which requires a forward model evaluation. The number of inference steps for diffusion models is generally much higher than that of EKI.

As a trade-off, EKI cannot generate samples from arbitrarily complex posteriors. Theoretically, EKI is exactly equivalent to score-based posterior sampling only under linear-Gaussian assumptions. Although many works have shown EKI can handle non-linear problems in practice, one should not expect a perfect match between the EKI-approximated posterior and the true one that is highly non-Gaussian. The posterior generated by EKI can be viewed as an optimal Gaussian approximation to the true posterior. In this work, we have shown that such an approximate posterior can be useful for active learning and, more specifically, Bayesian experimental design problems.

A.2 ALGORITHM

We provide the overall algorithm of active learning in Algorithm 1 and also the detailed algorithm to evaluate the EKI-based utility in Algorithm 2. For multi-step EKI, one can repeat Lines 2–10 using the updated ensemble from the previous step as the current ensemble, and then compute the information gain from the final updated ensemble.

Algorithm 1 Active Task Selection for Communication-Graph Optimization

Require: Training pool $\mathcal{D}_{\text{tr}} = \{(q_i, y_i)\}_{i=1}^{M}$; representative-pool size R; utility-evaluation budget K_{eval}; final training budget K; ensemble size J; prior distribution $p(\mathbf{z})$; target score l^*; observation-noise covariance Γ.

Ensure: Selected task subset $\mathcal{S}$.

1: Embed each task q_i into a vector $v_i = v(q_i)$.
2: Construct a representative pool $\mathcal{P}_R \subseteq \mathcal{D}_{\text{tr}}$ with $|\mathcal{P}_R| = R$ using greedy farthest-point selection in the embedding space.
3: Initialize the evaluated set $\mathcal{O} \leftarrow \emptyset$.
4: Select an initial subset $\mathcal{B}_0 \subseteq \mathcal{P}_R$ for utility evaluation.
5: **for** each task $(q, y) \in \mathcal{B}_0$ **do**
6: Estimate its utility $U(q)$ by Algorithm 2.
7: Update $\mathcal{O} \leftarrow \mathcal{O} \cup \{(q, U(q))\}$.
8: **end for**
9: **while** the total number of evaluated tasks is smaller than K_{eval} **do**
10: Fit or refit the surrogate model $\bar{\psi} : v(q) \mapsto U(q)$ on the evaluated set $\mathcal{O}$, where $\bar{\psi}$ is implemented by PLS dimension reduction followed by GP regression.
11: Use the surrogate acquisition rule, e.g., Thompson sampling, to select a new batch

$$\mathcal{B} \subseteq \mathcal{P}_R \setminus \{q : (q, U(q)) \in \mathcal{O}\}.$$

12: **for** each task $(q, y) \in \mathcal{B}$ **do**
13: Estimate its utility $U(q)$ by Algorithm 2.
14: Update $\mathcal{O} \leftarrow \mathcal{O} \cup \{(q, U(q))\}$.
15: **end for**
16: **end while**
17: Select the final training subset

$$\mathcal{S} = \text{TopK}_{(q,U(q))\in\mathcal{O}}\, U(q).$$

18: **return** $\mathcal{S}$.

A.3 NOTATION SUMMARY

We provide a notation summary in Table 2.

Algorithm 2 One-Step EKI Utility Estimation for a Task

Require: Task-answer pair (q, y); prior distribution $p(\mathbf{z})$; ensemble size J; target score l^*; observation-noise covariance Γ.
Ensure: Task utility $U(q)$.

1: Draw an initial ensemble
$$\{\mathbf{z}^{(j)}\}_{j=1}^{J} \sim p(\mathbf{z}).$$
2: **for** $j = 1, \ldots, J$ **do**
3: Run the MAS forward map:
$$g^{(j)} = G(q; \mathbf{z}^{(j)}, \omega^{(j)}).$$
4: Evaluate the MAS output:
$$l^{(j)} = \phi(g^{(j)}, q, y).$$
5: **end for**
6: Compute empirical means
$$\bar{\mathbf{z}} = \frac{1}{J}\sum_{j=1}^{J}\mathbf{z}^{(j)}, \qquad \bar{l} = \frac{1}{J}\sum_{j=1}^{J} l^{(j)}.$$
7: Compute the empirical cross-covariance and observation (score) covariance:
$$C^{zl} = \frac{1}{J-1}\sum_{j=1}^{J}\big(\mathbf{z}^{(j)} - \bar{\mathbf{z}}\big)\big(l^{(j)} - \bar{l}\big)^{\top},$$
$$C^{ll} = \frac{1}{J-1}\sum_{j=1}^{J}\big(l^{(j)} - \bar{l}\big)\big(l^{(j)} - \bar{l}\big)^{\top}.$$
8: **for** $j = 1, \ldots, J$ **do**
9: Update the ensemble particle:
$$\mathbf{z}_{\text{post}}^{(j)} = \mathbf{z}^{(j)} + C^{zl}\big(C^{ll} + \Gamma\big)^{-1}\big(l^* - l^{(j)}\big).$$
10: **end for**
11: Approximate the information gain using a Gaussian approximation of the ensembles:
$$U(q) = I(q) \approx D_{\mathrm{KL}}\left(\{\mathbf{z}_{\text{post}}^{(j)}\}_{j=1}^{J} \,\middle\|\, \{\mathbf{z}^{(j)}\}_{j=1}^{J}\right).$$
12: **return** $U(q)$.

A.4 Prediction-step reuse under exhaustive subset evaluation

This appendix presents a possible batch extension of the single-task EKI-based utility estimator. This formulation is not used in the main experiments, where task gains are computed at the single-task level. We include it to show how the same EKI-based update can be generalized to subset-level information gain and how forward evaluations can be reused when candidate subsets overlap. We focus here on the estimator and the associated reuse structure, rather than on the full combinatorial search problem over task subsets. Developing a practical acquisition strategy for efficiently exploring the batch-subset space is an interesting direction for future work.

A.4.1 Reuse formulation in batch EKI

We describe how the prediction-step reuse is incorporated into the batch EKI formulation. Consider a candidate pool $\mathcal{D}_{\text{tr}} = \{(q_i, y_i)\}_{i=1}^{M}$ and the initial ensemble $\{\mathbf{z}^{(j)}\}_{j=1}^{J}$. For each task q_i and each ensemble particle $\mathbf{z}^{(j)}$, we first run the forward map and compute the corresponding score
$$g_i^{(j)} = G(q_i; \mathbf{z}^{(j)}), \qquad l_i^{(j)} = \phi(g_i^{(j)}, q_i, y_i) \in \mathbb{R}.$$

Table 2: Summary of key notation.

Symbol	Meaning
$\mathcal{D}_{\text{tr}} = \{(q_i, y_i)\}_{i=1}^{M}$	Full training task pool, where q_i is a task and y_i is its ground-truth answer.
$\mathcal{S}$	Selected subset of tasks used for communication-graph optimization.
$\mathcal{G} = (V, E^S, E^T)$	Multi-agent communication graph, where V is the set of agents and E^S, E^T are the spatial and temporal adjacency matrices.
$\mathbf{z} = \{Z^S, Z^T\}$	Trainable graph logits that parameterize the spatial and temporal communication masks.
$A = \{A^S, A^T\}$	Random communication mask sampled from the logits $\mathbf{z}$.
$G(q; \mathbf{z}, \omega)$	MAS forward map for task q under graph logits $\mathbf{z}$ and internal stochasticity ω.
g	Output of the MAS, such as a choice, numerical answer, or code output.
$\phi(g, q, y)$	Evaluator that maps the MAS output, task, and ground truth to a performance score.
l	Performance score produced by the evaluator.
$U(q)$	Utility of a task q, measured by the information gain induced by the task.
$U(\mathcal{S})$	Utility of a selected task subset $\mathcal{S}$.
$I(q)$	Information gain of a single task, approximated by the KL divergence between the EKI-updated ensemble and the prior ensemble.
$p(\mathbf{z})$	Prior distribution over graph logits.
$p(\mathbf{z} \mid q, y)$	Posterior distribution over graph logits after conditioning on a task-answer pair (q, y).
$D_{\text{KL}}(\cdot\|\cdot)$	Kullback–Leibler divergence used to quantify information gain.
$\{\mathbf{z}^{(j)}\}_{j=1}^{J}$	EKI ensemble representing uncertainty over graph logits.
J	EKI ensemble size.
l^*	Desired target score used in the EKI update.
Γ	Observation-noise covariance (in score space) in the EKI update.
$v(q)$	Embedding vector of task q.
$\mathcal{P}_R$	Representative task pool selected from the original training pool.
$\bar{\psi}$	Surrogate model, e.g., a GP, that approximates the map from reduced task embeddings to utilities.

These task-level scores are cached once and reused for different candidate subsets. Specifically, for each ensemble particle, we define the cached score vector over the full candidate pool as

$$\mathbf{l}^{(j)} = [l_1^{(j)}, l_2^{(j)}, \ldots, l_M^{(j)}]^\top \in \mathbb{R}^M.$$

For a candidate subset $\mathcal{S} = \{q_{i_1}, q_{i_2}, \ldots, q_{i_m}\}$, the batch score vector for particle j is obtained by selecting the corresponding entries from the cached pool-level score vector:

$$\mathbf{l}_{\mathcal{S}}^{(j)} = [l_{i_1}^{(j)}, l_{i_2}^{(j)}, \ldots, l_{i_m}^{(j)}]^\top \in \mathbb{R}^m.$$

Thus, forming $\mathbf{l}_{\mathcal{S}}^{(j)}$ only requires indexing and stacking cached task-level scores, rather than rerunning the forward map.

The subset-specific ensemble mean of the score vectors is

$$\bar{\mathbf{l}}_{\mathcal{S}} = \frac{1}{J}\sum_{j=1}^{J} \mathbf{l}_{\mathcal{S}}^{(j)}, \qquad \bar{\mathbf{z}} = \frac{1}{J}\sum_{j=1}^{J} \mathbf{z}^{(j)}.$$

Then the cross-covariance and score covariance used in the EKI update are computed as

$$C_{\mathbf{z}l}^{\mathcal{S}} = \frac{1}{J-1}\sum_{j=1}^{J} (\mathbf{z}^{(j)} - \bar{\mathbf{z}})(\mathbf{l}_{\mathcal{S}}^{(j)} - \bar{\mathbf{l}}_{\mathcal{S}})^\top,$$

and

$$C_{ll}^{\mathcal{S}} = \frac{1}{J-1}\sum_{j=1}^{J} (\mathbf{l}_{\mathcal{S}}^{(j)} - \bar{\mathbf{l}}_{\mathcal{S}})(\mathbf{l}_{\mathcal{S}}^{(j)} - \bar{\mathbf{l}}_{\mathcal{S}})^\top.$$

Here, $C_{\mathbf{z}l}^{\mathcal{S}} \in \mathbb{R}^{d_z \times m}$ and $C_{ll}^{\mathcal{S}} \in \mathbb{R}^{m \times m}$, where d_z is the dimension of the graph parameter vector.

Given the batch-level target score vector $\mathbf{l}_{\mathcal{S}}^{\text{obs}} \in \mathbb{R}^m$ and the observation noise covariance $\Gamma_{\mathcal{S}} \in \mathbb{R}^{m\times m}$, the EKI update for subset $\mathcal{S}$ is

$$\mathbf{z}_{\text{post},\mathcal{S}}^{(j)} = \mathbf{z}^{(j)} + C_{\mathbf{z}l}^{\mathcal{S}} \left(C_{ll}^{\mathcal{S}} + \Gamma_{\mathcal{S}}\right)^{-1} \left(\mathbf{l}_{\mathcal{S}}^{\text{obs}} - \mathbf{l}_{\mathcal{S}}^{(j)}\right).$$

The resulting posterior ensemble $\{\mathbf{z}_{\text{post},\mathcal{S}}^{(j)}\}_{j=1}^{J}$ is then used to compute the batch information gain $I(\mathcal{S})$.

This formulation shows that the prediction step is performed at the task level, while the EKI update step is performed at the subset level. Therefore, when different candidate subsets overlap, their shared tasks reuse the same cached scores $l_i^{(j)}$, whereas the covariance matrices $C_{\mathbf{z}l}^{\mathcal{S}}$, $C_{ll}^{\mathcal{S}}$, the posterior ensemble, and the resulting information gain remain specific to each subset $\mathcal{S}$.

A.4.2 Cost analysis

If batch information-gain evaluation involves exhaustive subset enumeration, the reuse strategy can substantially reduce repeated forward-model evaluations. Consider a candidate pool of M tasks and suppose that all size-m subsets are evaluated under the same current ensemble of size J. A naive exhaustive evaluation would compute the prediction step separately for each subset, requiring

$$\binom{M}{m} mJ$$

forward-model evaluations. Here, $\binom{M}{m}$ denotes the number of possible size-m subsets that can be formed from a pool of M candidate tasks. However, the prediction associated with a task does not depend on the subset in which the task appears, as long as all subsets are evaluated from the same current ensemble. Therefore, task-level predictions can be cached once for all M unique tasks, reducing the prediction-step cost to

$$MJ.$$

The resulting reduction factor is

$$\frac{\binom{M}{m} mJ}{MJ} = \binom{M-1}{m-1}.$$

For example, selecting 5 tasks from a pool of 10 already gives $\binom{10}{5} = 252$ candidate subsets, resulting in a $\binom{9}{4} = 126$-fold reduction in prediction-step forward evaluations. Selecting 10 tasks from a pool of 50 gives

$$\binom{50}{10} = 10{,}272{,}278{,}170,$$

which is on the order of 10^{10} possible subsets. The corresponding prediction-step reduction factor is

$$\binom{49}{9} = 2{,}054{,}455{,}634.$$

This large factor comes from the combinatorial repetition of the same tasks across different subsets. These values are meant only to illustrate the potential savings under exhaustive subset enumeration, since our method does not enumerate all possible subsets in practice. Nevertheless, the same reuse principle applies whenever the evaluated candidate subsets overlap: shared tasks can reuse cached prediction results, reducing the number of repeated forward rollouts.

A.4.3 Remark

One subtlety of aggressive prediction reuse is that all candidate subsets are evaluated conditional on the same finite ensemble. In the large-ensemble limit, this is harmless, since the empirical ensemble distribution converges to the target uncertainty distribution and reuse only avoids duplicated forward evaluations. With a small ensemble, however, the ensemble provides only a low-rank and random representation of parameter uncertainty. Consequently, all subset utilities are evaluated through the same empirical covariance directions and share the same ensemble-induced approximation error. This might make the ranking of candidate subsets sensitive to the particular ensemble realization:

subsets that are informative along the sampled directions may be favored, while subsets whose informativeness lies in poorly represented directions may be undervalued. The severity of this effect depends on the ensemble size, the posterior geometry, and the diversity of the candidate subsets. A systematic characterization of this effect is left for future work.

However, since reuse reduces repeated forward rollouts, part of the saved computational budget could be allocated to increasing the ensemble size. Besides, in practice, one may reuse task-level predictions only within a limited collection of candidate subsets. Designing finite-reuse strategies that balance computational savings and finite-ensemble approximation error is also an interesting direction for future work.

Overall, this formulation highlights a key advantage of the batch EKI view: the expensive prediction step can be organized at the task level, while subset-specific information gains are obtained through lightweight recombination in the update step.

A.5 Multi-objective extension of EKI-based utility estimation

A.5.1 Vector-valued utility estimation

It is also possible to incorporate multiple objectives explicitly, rather than collapsing them into a single scalar score. In a traditional scalarized multi-objective formulation, the loss is often written as

$$l = \phi(\cdot) = \alpha_1 l_1 + \alpha_2 l_2 + \cdots + \alpha_n l_n \in \mathbb{R}, \tag{26}$$

where l_i denotes the i-th objective and α_i is its corresponding weight. In graph optimization, these objectives may represent, for example, downstream accuracy, graph sparsity, and regularization Zhang et al. (2025b); Wang et al. (2025c). However, such scalarization mixes the objective signals before the EKI update is formed, so improvement in one component may offset degradation in another.

Instead, we can define a vector-valued pseudo-observation as

$$l = [\alpha_1 l_1, \alpha_2 l_2, \cdots, \alpha_n l_n]^\top \in \mathbb{R}^n, \tag{27}$$

so that each objective enters the update as a separate component. Under the standard EKI formulation, this corresponds to matching a weighted quadratic residual in the observation space, which allows the posterior update to respond to each objective more explicitly instead of only their pre-aggregated sum. As a result, the update is driven by the component-wise residual structure rather than by a single pre-aggregated scalar score. This does not remove trade-offs among objectives, but it avoids premature cancellation caused by scalarization and yields a more structured update signal.

A.5.2 Remark

The multi-objective extension (Appendix A.5) and the batch extension (in Sec 4.3) act on different axes of the pseudo-observation and are therefore complementary rather than redundant. The batch extension increases the observation dimension across tasks, while the multi-objective extension increases the dimension within each task by keeping different objectives as separate components. When both are used together, each task q_i is associated with an n-dimensional pseudo-observation vector

$$l_i = [\alpha_1 l_{i,1}, \alpha_2 l_{i,2}, \cdots, \alpha_n l_{i,n}]^\top \in \mathbb{R}^n, \tag{28}$$

and a batch $\mathcal{S} = \{q_i, y_i\}_{i=1}^m$ is represented by concatenating these task-wise vectors into a single observation vector

$$l_{\mathcal{S}} = [l_1^\top, l_2^\top, \cdots, l_m^\top]^\top \in \mathbb{R}^{mn}. \tag{29}$$

This unified form can be directly used in the EKI update, allowing the method to simultaneously account for multiple tasks and multiple objectives.

B Experimental details

In this section, we provide detailed experimental settings for the main MMLU and GSM8K experiments, followed by the settings for the broader benchmarks in Appendix B.7.

B.1 DATASET

For GSM8K, we use the `train` split as the task pool for communication-structure optimization and the `test` split for evaluation. We report results on the first 460 tasks in the `test` split, which provides a sufficiently large evaluation set for stable empirical comparison across runs.

For MMLU, the original implementation in Zhang et al. (2025b) uses the 'dev' split to randomly select tasks for graph optimization. However, the 'dev' split is less suitable as an active-learning candidate pool, since it contains only five tasks for each subject. Such a small and highly structured pool limits task diversity and is less favorable for representative selection based on clustering or task embeddings. Therefore, in our setting, we use the 'test' split as the candidate pool for communication-structure optimization, since it is large enough to support active learning over a more diverse set of tasks. For evaluation, we use the first 153 tasks in the `val` split, following Zhang et al. (2025b). In this way, the task pool used for communication-structure optimization is separated from the evaluation set.

Unless otherwise stated, all the other settings are the same for both datasets.

B.2 ACTIVE LEARNING PIPELINE AND RANDOM TRAINING PIPELINE

For active learning, we use the same selection pipeline in both datasets and settings (with and without attack). We first randomly sample 1000 tasks from the full training set, then select 50 representative tasks based on task embeddings, and finally choose training tasks from this 50-task pool.

The 1000-to-50-to-10 pipeline is used as an illustrative default configuration, rather than as a tuned or problem-specific choice. The final number (10) of selected tasks is determined by the training budget (i.e., how many task usages one can afford during graph optimization). The size of the intermediate candidate pool is determined by the selection budget (i.e., how many tasks can be evaluated for informativeness). In practice, these three quantities can be adjusted independently to accommodate different computational settings and resource constraints.

For active learning, we select 10 informative tasks and use each task twice during training, resulting in 20 total task usages. Here, a "task usage" means one pass where the LLM-MAS is executed on a task to produce an answer, the answer is scored by the task-specific metric, and this score is used to compute the training signal (e.g., a policy-gradient estimate) for updating the communication-graph parameters. We adopt this repeated-use design because using each selected task only once may not provide a sufficiently reliable training signal. Due to stochasticity in the forward map and the resulting noisy graph-optimization signal, the effect of an informative task can be weakened if it is used only once.

For random selection, we randomly select tasks from the whole training pool and match the total number of task usages in training. On GSM8K, for consistency with the active-learning training schedule, we randomly sample 10 tasks and use each task twice. On MMLU, we notice that the original training implementation of Zhang et al. (2025b) uses each selected task only once. To keep the forward model and training procedure as black-boxes with minimal changes, we follow this design and randomly sample 20 tasks from the whole pool, using each once, so that the total number of task usages remains 20. Overall, across the two commonly used ways to allocate an equal training budget, we consistently observe that active learning outperforms random task training.

In addition, on MMLU we performed a sanity check by swapping the random-training repetition schedule from 20 tasks × 1 to 10 tasks × 2 while keeping the total task usages fixed. The random baseline remains very similar (76.87 ± 3.72 as shown in the main text vs. 76.42 ± 2.55 over 15 runs), indicating that this choice does not materially affect the random-training performance.

B.3 EKI HYPERPARAMETERS

We use a one-step EKI update with ensemble size 6, prior perturbation standard deviation 2.0, observation noise 0.1, and damping factor 0.7.

Since our experiments use a single interaction round, temporal communication is inactive: E^T is fixed to the zero matrix and is not optimized. Therefore, only the spatial graph logits $Z^S \in \mathbb{R}^{6\times 6}$ are optimized, resulting in a 36-dimensional graph-parameter space. An ensemble size of 6 is small relative to the parameter dimension. We use a prior perturbation standard deviation of 2.0 to cover a

relatively broad range of initial perturbations. Since the forward system does not include explicit observation noise, we use a small fixed value of 0.1 as a regularizing noise level in the EKI update. The damping factor 0.7 is used to make the update more conservative and numerically stable.

For the performance score l and target value l^*, we use the log-probability of the correct answer (A/B/C/D) for MMLU and therefore $l^* = 0$. We use 0/1 correctness for GSM8K and set $l^* = 1$.

B.4 KL DIVERGENCE ESTIMATION

For the empirical KL computation, we use a diagonal Gaussian approximation rather than a full-covariance Gaussian approximation. This choice is motivated by the small ensemble size used in our experiments. With $J = 6$ particles and a 36-dimensional graph-logit parameter, the empirical full covariance matrix would be rank-deficient and therefore unsuitable for a standard Gaussian KL computation. We therefore estimate only the coordinate-wise empirical means and variances of the prior and updated ensembles. We also apply a small variance floor of 10^{-8} to avoid numerical instability from zero or near-zero empirical variances.

This diagonal approximation is an implementation choice for the small-ensemble setting, rather than a restriction of the proposed framework. When the ensemble size is large relative to the parameter dimension and the covariance estimate is well-conditioned, we recommend using a full-covariance Gaussian KL instead. This is consistent with similar ensemble-based Gaussian KL estimation practice in Yang et al. (2026a).

B.5 THOMPSON SAMPLING AND GAUSSIAN PROCESS SURROGATE

After representative selection, Thompson sampling is performed on the 50-task pool in a sequential batched manner. We first randomly evaluate 10 or 12 tasks to form an initial observed set. Then we perform additional acquisition rounds, selecting 5 or 6 new tasks in each round, giving an overall schedule of $10 + 5 \times 3$ for MMLU and $12 + 6 \times 4$ for GSM8K. After each round, the newly evaluated tasks and their true utility values are added to the observed set, and the surrogate model is refit. The purpose of these rounds is to progressively improve the surrogate using newly acquired labeled tasks, so that later selections are guided by a more accurate model.

To make surrogate fitting more stable, we first use the 10 or 12 initially evaluated tasks and their utility values to supervise a partial least squares (PLS) projection. Concretely, the task embeddings from representative selection are originally 384-dimensional, and PLS is used to reduce them to a much smaller latent space, typically 2 or 3 dimensions. This step uses the observed utility values to identify directions in the embedding space that are most relevant to task informativeness, rather than performing unsupervised dimensionality reduction. We use a small latent dimension because the representative pool contains only 50 tasks and the number of labeled tasks is limited in the early rounds, so a higher-dimensional surrogate is not expected to be beneficial in this setting.

A Gaussian process (GP) surrogate is then fit on the reduced features to predict utility values for the remaining tasks. Thompson sampling is implemented by drawing samples from the GP posterior and selecting the tasks with the highest sampled values. In this way, PLS provides a utility-aware low-dimensional representation, while the GP supplies the predictive model used for sequential task acquisition. The GP uses a Matérn kernel with smoothness parameter $\nu = 2.5$, combined with a learnable output scale and an additive white-noise term.

By applying this sequential decision-making process, we avoid evaluating the true utility of all 50 tasks in the representative pool. Instead, we only evaluate 25 tasks for MMLU and 36 tasks for GSM8K, reducing the utility-evaluation cost by 50% and 28%, respectively. In our experiments, this reduced evaluation budget still achieves more than 80% overlap with the top tasks selected by full enumeration. This suggests that the surrogate-assisted Thompson-sampling procedure captures most of the useful acquisition signal while substantially reducing the number of expensive utility evaluations. The benefit of this strategy is expected to become more important for larger candidate pools, where exhaustive enumeration becomes increasingly infeasible.

More advanced acquisition schedules, more carefully tuned surrogate models, or more advanced decision-making strategies may further reduce the evaluation cost or improve the overlap with full enumeration.

B.6 Training setup

Unless otherwise stated, we follow the default training setup in Zhang et al. (2025b) for communication-structure optimization.

B.7 Broader benchmark settings

Table 3: Experimental settings for the broader benchmarks.

Benchmark	Selection pipeline	Agents / adv.	Test tasks	Metric	l^*
HumanEval	$94 \to 50 \to 10$	$5/3$	50	Accuracy	1
InterCode-SQL	$834 \to 50 \to 10$	$5/1$	50	Success@5, AUC@5	1
ALFWorld	$20 \to 12 \to 10$	$4/2$	20	Success@30, AUC@30	1

The selection pipeline reports the candidate-pool size, representative-pool size, and final number of selected training tasks, respectively. Each selected task is used twice during graph optimization.

For HumanEval, we use the binary code-execution as performance score for graph optimization, with $l = 1$ if the generated code passes execution and $l = 0$ otherwise, and set $l^* = 1$. For InterCode-SQL and ALFWorld, we combine task completion, early success, and intermediate progress as

$$l = 0.6\,\mathrm{Success@}k + 0.2\,\mathrm{AUC@}k + 0.2\,\mathrm{BestReward@}k,$$

where $k = 5$ for InterCode-SQL and $k = 30$ for ALFWorld. $\mathrm{Success@}k$ measures the fraction of tasks solved within k interaction steps, while $\mathrm{AUC@}k$ averages success over the interaction horizon and therefore also reflects how early a task is solved. $\mathrm{BestReward@}k$ denotes the highest environment reward obtained within the first k interaction steps. Since the maximum possible score is 1, we set $l^* = 1$ for both benchmarks.

C MAS setup details

C.1 General setup

We directly adopt the MAS execution framework of AgentPrune Zhang et al. (2025b) in all experiments. Although the present experiments are built on this specific system, the proposed active learning method is largely independent of the particular MAS construction, since it mainly treats the MAS as a black-box forward model under different communication graphs. Below we summarize the execution details most relevant to our experiments.

The multi-agent system is represented by a spatial-temporal communication graph, where the spatial graph describes information exchange within each interaction round and the temporal graph describes how information is passed across adjacent rounds. Each node corresponds to an agent. In our setting, the graph structure is parameterized by trainable logits, represented in our notation by the $\mathbf{z}$. These logits are first transformed by a sigmoid function into edge-retention probabilities in $[0, 1]$. A Bernoulli variable is then sampled for each edge to determine whether that communication edge is active in the current graph realization. In this way, the logits do not directly define a deterministic graph; rather, they parameterize a distribution over communication graphs.

Within each interaction round, agent execution must follow a valid order. Therefore, the spatial communication graph is executed as a directed acyclic graph (DAG), so that agents can be processed in topological order. Intuitively, an agent can only read messages that have already been produced by its spatial predecessors in the same round. The temporal graph, in contrast, determines which outputs from the previous round are passed forward to the current round. Thus, spatial edges control same-round communication, while temporal edges control cross-round memory and dialogue-history propagation.

Given an input task q, the system runs for multiple rounds. At round t, each agent receives the task together with messages from its temporal in-neighbors from round $t - 1$ and its spatial in-neighbors from the current round t. The agent then produces its response conditioned on this collected context, along with its predefined role and internal state. Therefore, the communication graph affects the

MAS entirely through message routing: it determines which agent outputs are visible to which other agents, both within a round and across rounds.

After the final round, a decision agent reads the resulting dialogue trajectory and produces the final answer. In our experiments, this entire execution pipeline, including the prompt family and role assignment, is inherited from AgentPrune; our contribution does not modify the MAS execution mechanism itself, but instead studies how to select informative tasks for optimizing its communication structure under limited training budgets.

C.2 Agent roles, prompts, and attack details

C.2.1 MMLU

In this dataset, agent roles, prompts and attack details follow AgentPrune. We briefly introduce the main settings here.

Agents are assigned role-specific prompts from the following role set: **Knowledgeable Expert, Critic, Mathematician, Psychologist, Historian, Doctor, Lawyer, Economist**, and **Programmer**. These role prompts are used to induce diverse reasoning perspectives across agents on the same multiple-choice question. More specifically, Knowledgeable Expert is prompted to identify key entities relevant to the question and mark them using the @...@ format. Critic is prompted to identify potential issues in other agents' analyses. The remaining roles are defined through domain-oriented descriptions: Mathematician emphasizes mathematical reasoning and calculation; Psychologist emphasizes psychology, sociology, and philosophy; Historian emphasizes historical, political, economic, and social analysis; Doctor emphasizes medical reasoning; Lawyer emphasizes law, politics, and history; Economist emphasizes economics, finance, business, and chart interpretation; and Programmer emphasizes computer science, engineering, and physics.

All answering agents are given the same answer-format constraint. Each agent receives a four-option multiple-choice question with exactly one correct answer and is required to output one option from A, B, C, D. The response must be fewer than 100 words, with the first line containing only the selected option letter, followed by a brief step-by-step analysis. Agents are allowed to refer to other agents' responses, but the analysis prompt explicitly instructs them to use such information only as additional advice and not to imitate other agents' reasoning processes.

Answering agents are assigned roles by cycling through a fixed role pool. In the agent-attack setting, half of the agents are assigned the **Fake** role. The roles are allocated in an alternating manner: if one agent is assigned a regular role, the next agent is assigned Fake, and so on. The regular agents continue to cycle through the standard role pool described above.

For agent-attack experiments, adversarial agents are prompted with a dedicated attack template rather than the standard answering prompt. The attack prompt explicitly instructs the agent to provide an incorrect answer together with a false analysis process for the given question. It further specifies that, even if outputs from other agents are available, the adversarial agent should continue to produce deceptive content and try to mislead the other agents.

C.2.2 GSM8K

For GSM8K without agent attack, answering agents are assigned role-specific prompts from a fixed role pool consisting of **Math Solver, Mathematical Analyst, Programming Expert**, and **Inspector**. These roles are assigned in a cyclic manner across agents. The role prompts are designed to induce complementary solution styles for arithmetic reasoning: **Math Solver** is prompted to provide a direct step-by-step mathematical solution; **Mathematical Analyst** is prompted to first express the solution process in symbolic form and then substitute numerical values; **Programming Expert** is prompted to solve the problem using Python code only, with the final result assigned to the variable `answer`; and **Inspector** is prompted to check the correctness of other agents' reasoning or code and then provide its own step-by-step solution. In addition, the answer prompt prepends role-specific few-shot exemplars for **Math Solver**, **Mathematical Analyst**, and **Programming Expert** before the target question.

The decision agent in GSM8K is prompted separately as a top decision-maker. It receives the mathematical problem together with analyses or code from other agents, and is instructed to provide the final answer by following the most common or most confident conclusion among the candidate

outputs. Its response is required to end with a final line containing only the numerical result in the format "The answer is ...", without units.

In the agent-attack setting, some of the agents are replaced with **Fake** agents, while the remaining regular agents continue to use the standard GSM8K role pool described above. Fake agents are prompted to produce a convincing but incorrect step-by-step solution, to make the reasoning appear logical and confident, and to keep the final answer wrong. The prompt also explicitly requires the agent not to reveal that it is being misleading. They are required to follow or modify the incorrect answer from previous dialogue, so that their incorrect answers become more convincing.

C.2.3 Clarification for attack agents

In our current adversarial experiments, the adversarial agents are pre-specified in each experiment run and remain fixed across tasks and dialogue rounds within that run. We do not provide the identity of the adversarial agents to the other agents. Evaluating settings where the adversarial identity varies across tasks or even dialogue rounds is an important direction for future study.

D Additional results

D.1 Scalability to larger MAS and generalization across LLM backbones

D.1.1 Scalability

To evaluate whether the proposed method remains effective beyond the system scale considered in the main experiments, we further test adversarial MMLU with 10 and 20 agents. One-third of the agents are adversarial, and the EKI ensemble sizes are 10 and 15, respectively.

Table 4: Performance comparison across agent scales under agent attacks. Accuracy is reported as mean $\pm$ std, first quartile (Q1), and mean below Q1. Token cost is reported as mean.

		Accuracy / (%)			Token cost / (M)		
Case	Setup	Mean $\pm$ std	Q1	Worst-25%	Select	Train	Test
MMLU 10 agents	No train	76.39 ± 2.26	74.83	73.62	-	-	2.40
	Random	77.27 ± 2.64	75.81	74.51	-	0.20	1.20
	Active	78.13 ± 1.85	**76.80** (+0.99)	**75.61** (+1.10)	2.80	0.20	1.20
MMLU 20 agents	No train	79.55 ± 2.36	77.12	76.47	-	-	7.40
	Random	78.43 ± 2.41	76.47	75.82	-	0.60	3.40
	Active	80.91 ± 2.17	**79.73** (+3.26)	**78.75** (+2.93)	13.00	0.60	3.60

Run counts for no training/random/active are 6/13/11 for 10-agent MMLU and 7/5/5 for 20-agent MMLU.

Active selection outperforms random training at both larger MMLU scales, improving mean accuracy from 77.27% to 78.13% with 10 agents and from 78.43% to 80.91% with 20 agents. These results show that the benefit of active task selection is not restricted to the original six-agent system and can also be observed in larger systems.

For larger systems beyond the evaluated scale, EKI itself is not the primary computational bottleneck; the dominant scaling comes from the increased cost of a single MAS rollout itself. For a system with N agents, the number of graph parameters scales as $O(N^2)$, while the token cost of one MAS rollout scales from $O(N)$ for sparse communication to $O(N^2)$ for dense communication. If the EKI ensemble size grows approximately linearly with N, the overall active-selection cost therefore scales between $O(N^2)$ and $O(N^3)$. Empirically, the selection cost increases from 0.8M tokens for 6 agents to 2.8M for 10 agents and 13.0M for 20 agents, approximately following quadratic growth over the tested range. The $O(N)$–$O(N^2)$ per-rollout cost is intrinsic to executing the MAS, while EKI contributes only the additional ensemble-size factor. For substantially larger systems, techniques such as localization or graph/ensemble dropout could further reduce the required ensemble size.

D.1.2 GENERALIZATION ACROSS LLM BACKBONES

We also evaluate HumanEval with a different LLM backbone to examine whether the observed benefit depends on the underlying LLM. We compare GPT-4.1-nano with GPT-5-nano and use five agents with three adversarial agents.

On HumanEval, active selection improves mean accuracy from 77.80% to 78.80% with GPT-4.1-nano and from 85.60% to 86.40% with GPT-5-nano. These results show that our method and its benefits are not limited to any specific LLM backbone.

Table 5: Performance comparison across LLM backbones under agent attacks. Accuracy is reported as mean ± std, first quartile (Q1), and mean below Q1. Token cost is reported as mean.

		ACCURACY / (%)			TOKEN COST / (M)		
CASE	SETUP	MEAN ± STD	Q1	WORST-25%	SELECT	TRAIN	TEST
HUMAN-EVAL 4.1-NANO	NO TRAIN	75.20 ± 4.44	73.00	69.33	-	-	0.23
	RANDOM	77.80 ± 3.94	76.00	74.00	-	0.07	0.17
	ACTIVE	78.80 ± 4.44	76.00	74.67	0.64	0.09	0.17
HUMAN-EVAL 5-NANO	NO TRAIN	85.80 ± 2.20	84.50	83.33	-	-	0.28
	RANDOM	85.60 ± 3.50	82.50	82.00	-	0.18	0.20
	ACTIVE	86.40 ± 3.98	84.00	82.67	0.80	0.23	0.20

Run counts for no training/random/active are 10/10/10 for both HumanEval configurations.

D.2 ADDITIONAL RESULTS FOR ADVERSARIAL SETTING

In this section, we provide additional results for the adversarial setting. We include a representative example of graph change before and after optimization. We also report the context, EKI statistics, and simple comments on some of the tasks from the representative pool, in order to give an intuitive idea of what the tasks are, what kinds of tasks are informative, and how they affect the EKI process.

D.2.1 A REPRESENTATIVE GRAPH CHANGE ON MMLU

Figure 6 illustrates the practical effect of graph optimization in an adversarial setting. The left panel shows the initial communication graph as a fully connected topology at the parameter level, while the right panel shows the executable DAG after optimization. The optimized structure exhibits a clear pattern: normal agents no longer read messages from fake agents, indicating that the optimization process has learned to suppress harmful information flow from adversarial nodes. This behavior is consistent with the intended role of structure optimization under agent attack.

At the same time, the optimized DAG is not maximally sparse. Some fake agents still retain incoming edges from other normal or fake agents. We emphasize, however, that graph optimization is performed under a limited training budget rather than until full convergence. Under such a lightweight training setup, the learned graph already captures the main meaningful trend, namely, blocking fake-to-normal information flow while preserving a workable communication structure among useful agents.

D.2.2 SELECTED TASKS

We show the details of several tasks to provide an intuitive understanding of the features of different tasks. We enumerate 50 tasks and evaluate information gain (KL-based) for each of them. We then rank these 50 tasks by their information gain. We first show the content of a few tasks in Fig. 7. We then show their corresponding information gain and other statistics within the EKI process in Table 6. We finally give a brief analysis of the difficulty of these tasks in Table 7, which is generated by ChatGPT-5.4.

In Table 6, *Pred. var.* denotes the predictive variance across the ensemble, which reflects how differently the ensemble members respond to the same task. *Cosine* measures the directional alignment between the ensemble before and after the update; a value close to one indicates that the update preserves a highly similar direction, while a smaller value suggests a more noticeable

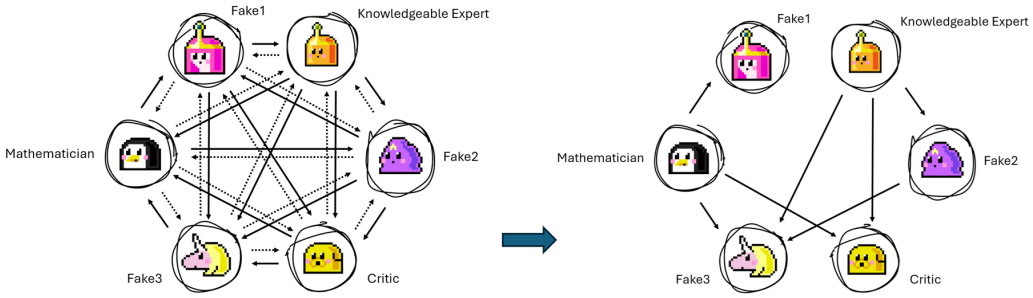

Figure 6: A representative example of graph change on MMLU under agent attack. The left panel shows the initial fully connected communication graph at the parameter level, while the right panel shows the optimized executable DAG after graph optimization. The learned structure suppresses several fake-to-normal communication paths while preserving a workable sparse topology.

directional change. *Var. sum* summarizes the overall ensemble variance after the update. These quantities provide complementary views of how each task interacts with the ensemble during the EKI update. Overall, a task with the largest information gain is often one for which the LLM-MAS exhibits high uncertainty. However, high uncertainty alone does not necessarily imply high informativeness, since the direction of the posterior update also matters.

The diagnostics show that tasks with large information gain tend to have large predictive variance and a smaller posterior ensemble variance. This suggests that the LLM-MAS is uncertain about these tasks, and that the corresponding EKI update substantially contracts the posterior ensemble. However, predictive uncertainty alone is not sufficient to determine informativeness. For example, the task at KL rank 16 still has a relatively large predictive variance, but its update direction remains close to that of the original ensemble and its posterior variance remains relatively large. This indicates that although the task induces substantial disagreement in the predicted outputs, this disagreement does not translate into a strong contraction of the graph-parameter posterior. Therefore, information gain depends not only on the magnitude of predictive uncertainty, but also on whether the task induces an informative update direction and effectively reduces posterior uncertainty.

Table 7 gives a qualitative characterization of the same examples. The column *Fact.* indicates whether the task requires domain-specific factual knowledge. *Multi-concept* describes whether the task involves combining multiple concepts, rules, or conditions. *Fine-grained* measures whether the answer depends on distinguishing closely related alternatives. *Direct recall* indicates whether the task can be answered mainly by recalling a standard definition, term, or fact. This qualitative analysis helps connect the numerical EKI diagnostics with visible task structures. The interpretation column is intended to summarize the overall requirement of each task, including the reasoning steps it requires, the source of challenge, and whether the answer is mainly obtained through rule application, fine-grained comparison, or direct recall. Overall, this table provides a rough indication of task difficulty.

Figure 7: Representative MMLU tasks at different EKI ranking positions from the same 50-task representative pool. The examples are shown to illustrate visible task structure rather than intrinsic task difficulty.

KL rank 1, task index 11106

Question. A state enacts a statute that prohibits "anyone over 60 years of age to run for public office." A state senator has been in office for three terms and wishes to seek re-election. The senator, who is 61, brings suit challenging the constitutionality of the state statute. Which of the following best states the burden of persuasion?
Choices. (A) Since a fundamental right is involved, the state must show the regulation is necessary to vindicate a compelling government interest.
(B) Since no fundamental right is involved, the petitioner must show the age restriction is not rationally related to a legitimate government interest.
(C) The state must show the age regulation substantially furthers an important government objective and does not impair the fundamental right to vote.
(D) The petitioner must show the statute violates due process by depriving her of the right to be a candidate.
Correct answer: B.

KL rank 6, task index 12110

Question. Which of the following goods is likely to have the most elastic demand curve?
Choices. (A) Demand for white Ford minivans
(B) Demand for automobiles
(C) Demand for Ford automobiles
(D) Demand for American-made automobiles
Correct answer: A.

KL rank 11, task index 9510

Question. Mary, a wealthy St. Petersburg widow, executed her first and only will on May 15, 1990 and died on August 18, 1990. Her will provided that her estate be divided equally between her only child, Joan, and the Salvation Army of Largo. How will Mary's estate actually be distributed?
Choices. (A) 100% to Joan.
(B) 100% to Joan if she files a timely petition requesting that the devise to the Salvation Army be avoided.
(C) 50% to Joan and 50% to the Salvation Army.
(D) 50% to Joan and the income from the remaining 50% to Joan for life, remainder to the Salvation Army, if Joan files a timely petition protesting the devise to the Salvation Army.
Correct answer: C.

KL rank 16, task index 8221

Question. Which of the following could result in autocorrelated residuals?
(i) Slowness of response of the dependent variable to changes in the values of the independent variables
(ii) Over-reactions of the dependent variable to changes in the independent variables
(iii) Omission of relevant explanatory variables that are autocorrelated
(iv) Outliers in the data
Choices. (A) (ii) and (iv) only
(B) (i) and (iii) only
(C) (i), (ii), and (iii) only
(D) (i), (ii), (iii), and (iv)
Correct answer: C.

KL rank 21, task index 1323

Question. In Ward-Leonard system, the lower limit of the speed imposed by

Choices. (A) Field resistance.
(B) Armature resistance.
(C) Residual magnetism of the generator.
(D) None of above.

Correct answer: C.

KL rank 26, task index 1918

Question. Which of the following anatomical regions of abdomen lies just distal to the sternum?

Choices. (A) Epigastric
(B) Hypochondriac
(C) Hypogastric
(D) Lumbar

Correct answer: A.

KL rank 31, task index 10592

Question. The vibrations in a transverse wave move in a direction

Choices. (A) along the wave
(B) perpendicular to the wave
(C) Both of these
(D) Neither of these

Correct answer: B.

KL rank 46, task index 12025

Question. Max was typically out of control whenever he attended preschool. Teachers tried time-outs and other punishments to no avail. His parents and the school decided to work with Max by giving him a sticker each time he behaved for a full hour. Once he accumulated ten stickers, he could present them to his parents who would give him a reward. The method the school and parents chose to employ is referred to as

Choices. (A) negative reinforcement
(B) a token economy
(C) a point value system
(D) negative punishment

Correct answer: B.

Table 6: Diagnostic comparison of tasks at different EKI ranking positions.

KL rank	Task index	KL	Pred. var.	Cosine	Var. sum
1	11106	3804.28	86.32	0.056	131.36
6	12110	91.31	49.52	0.526	121.54
11	9510	29.08	54.70	0.822	162.18
16	8221	12.44	76.60	0.862	206.03
21	1323	3.70	9.125×10^{-9}	0.959	220.80
26	1918	0.01	1.291×10^{-7}	1.000	257.92
31	10592	0.00	3.706×10^{-7}	1.000	261.38
46	12025	0.00	5.271×10^{-9}	1.000	261.09

Table 7: Qualitative characterization of representative tasks at different EKI ranks. Scores use a 0–2 scale, where 0 means the property is not salient, 1 means moderately present, and 2 means clearly present.

KL rank	Task index	Fact.	Multi-concept	Fine-grained	Direct recall	Task-structure interpretation
1	11106	2	2	2	0	This legal reasoning task requires more than recognizing a constitutional-law term. The solver must identify the relevant legal category, determine whether candidacy is treated as a fundamental right, and then match that classification to the correct burden of persuasion. The main challenge is the multi-step rule selection and the distinction among closely related levels of scrutiny.
6	12110	2	1	2	1	This economics task tests a standard concept, but the answer depends on comparing several nested market definitions. The challenge is fine-grained rather than computational: the solver must recognize that demand becomes more elastic when the good is defined more narrowly and then distinguish the most specific option.
11	9510	2	2	2	0	This legal rule-application task combines factual details about timing, beneficiary type, and estate distribution. The solver must decide which legal rule applies to the given factual setting before selecting the distribution outcome. The challenge lies in applying a domain-specific rule to a concrete scenario rather than recalling an isolated fact.
16	8221	2	2	2	0	This econometrics task requires evaluating several proposed causes of autocorrelated residuals and then combining the valid statements. The challenge is that the answer cannot be obtained from a single keyword; each statement must be judged separately, and the final option depends on a correct aggregation of multiple conditions.
21	1323	2	0	1	2	This engineering task mainly depends on specialized factual knowledge about the Ward-Leonard system. The problem provides limited contextual information from which the answer can be inferred, so the challenge is primarily whether the solver has memorized or previously learned the relevant technical property.
26	1918	2	0	1	2	This anatomy task asks the solver to map a spatial description to the corresponding anatomical region. The main requirement is domain-specific terminology recall, with only a modest spatial distinction among the options. The challenge is therefore localized to recognizing the correct anatomical term.
31	10592	2	0	0	2	This physics task is close to a definition-matching question. Once the solver recalls the definition of a transverse wave, the answer follows directly. The options are also organized as broad opposites, which makes the required distinction relatively direct.
46	12025	2	1	1	1	This psychology task presents a concrete behavioral scenario and asks the solver to identify the corresponding concept. The challenge is to map the described reward-token procedure to the correct behavioral term, but the scenario contains strong cues that directly point to the answer.

D.2.3 DETAILED STATISTICS FOR BROADER BENCHMARKS

Table 8 reports the detailed performance statistics corresponding to the broader-benchmark results in Fig. 4.

Table 8: Performance statistics for the broader benchmarks under adversarial settings. HumanEval results correspond to GPT-4.1-nano. Performance is reported as mean ± std, first quartile (Q1), and mean below Q1.

Benchmark	Metric	Setup	Runs	Mean ± std	Q1	Worst-25%
HumanEval	Accuracy	Random	10	77.80 ± 3.94	76.00	74.00
		Active	10	78.80 ± 4.44	76.00	74.67
InterCode-SQL	Success@5	Random	11	51.45 ± 5.15	50.00	45.33
		Active	11	57.27 ± 4.22	53.00	52.00
	AUC@5	Random	11	28.84 ± 2.91	27.20	24.93
		Active	11	31.67 ± 2.20	30.40	28.93
ALFWorld	Success@30	Random	5	45.00 ± 17.80	33.75	30.00
		Active	4	51.25 ± 14.36	48.75	30.00
	AUC@30	Random	5	18.12 ± 9.12	13.17	8.17
		Active	4	20.88 ± 5.61	19.54	12.67

D.3 RUN COUNTS AND BOOTSTRAP CONFIDENCE INTERVALS OF MMLU AND GSM8K

Table 9 reports the number of independent runs used to compute the accuracy statistics in the main table (Table 1). Each run uses a different random seed. For random training, the random seed controls the selection of training tasks. For active learning, it controls the initial 1000-task pool realization and the stochastic components in the active-learning procedure.

Table 10 reports bootstrap 95% confidence intervals for the main active-versus-random comparisons. The reported differences are computed as active learning minus random training for three metrics: mean accuracy, first quartile (Q1), and worst-25% mean. These intervals quantify the uncertainty of the estimated differences across independent runs.

We compute the intervals using a nonparametric bootstrap. For each comparison, we resample runs with replacement within each method while keeping the original number of runs in that method. We then evaluate the target difference for each bootstrap sample. This procedure is repeated 10,000 times, and the 2.5th and 97.5th percentiles of the resulting bootstrap distribution are reported as the 95% confidence interval.

Overall, the bootstrap intervals provide the strongest evidence under adversarial conditions. For GSM8K with attack, the 95% confidence intervals for all three metrics exclude zero, while for MMLU with attack, the interval for worst-25% mean excludes zero. The remaining intervals include zero. These results support particularly robust improvements in lower-tail performance under attack, while improvements whose intervals include zero are interpreted as observed effect-size trends rather than statistically decisive differences under the current number of independent runs.

Table 9: Number of independent runs used for the accuracy statistics reported in the main tables.

Setting	Dataset	No train	Random	Active
Benign	MMLU	13	17	17
Benign	GSM8K	11	14	13
Attack	MMLU	11	20	12
Attack	GSM8K	10	23	23

Table 10: Bootstrap 95% confidence intervals for EKI/active-learning improvements over random training. Differences are computed as EKI/active learning minus random training.

Dataset	Setting	Metric	Difference	95% bootstrap CI
MMLU	Without attack	Mean accuracy	+0.69	$[-0.58, 1.91]$
		Q1	+0.68	$[-1.30, 2.64]$
		Worst-25%	+1.30	$[-0.12, 2.99]$
GSM8K	Without attack	Mean accuracy	+0.28	$[-0.14, 0.71]$
		Q1	+0.32	$[-0.11, 0.98]$
		Worst-25%	+0.70	$[-0.11, 0.92]$
MMLU	With attack	Mean accuracy	+1.45	$[-0.38, 3.34]$
		Q1	+3.27	$[-0.65, 5.23]$
		Worst-25%	+4.25	$[1.53, 6.32]$
GSM8K	With attack	Mean accuracy	+0.93	$[0.41, 1.45]$
		Q1	+0.98	$[0.33, 1.67]$
		Worst-25%	+1.23	$[0.72, 1.71]$

D.4 Paired statistical analysis on InterCode-SQL

Unlike the MMLU and GSM8K experiments, the InterCode-SQL runs are paired by random seed; we therefore use a paired statistical test for this benchmark.

For InterCode-SQL, results are reported over 11 paired runs, with the compared methods matched by random seed. We therefore assess pairwise differences using a two-sided exact paired permutation test. Under the null hypothesis of no systematic difference between two methods, we enumerate all possible swaps of the two method labels within each paired run and compute the resulting distribution of the mean difference.

Table 11 reports the pairwise comparisons. Active learning improves Success@5 over random training by 5.82 percentage points ($p = 0.037$), while the corresponding AUC@5 improvement is 2.84 percentage points ($p = 0.066$). Relative to no training, active learning improves Success@5 by 9.45 percentage points ($p = 0.093$) and AUC@5 by 6.29 percentage points ($p = 0.047$).

Overall, these results provide additional statistical evidence that active task selection improves performance on InterCode-SQL, particularly in Success@5 relative to random training.

Table 11: Two-sided exact paired permutation tests for InterCode-SQL over 11 paired runs. Differences are computed as the first method minus the second method.

Metric	Comparison	Mean difference	p-value
Success@5	Random − No train	+3.64 pp	0.373
	Active − No train	+9.45 pp	0.093
	Active − Random	+5.82 pp	0.037
AUC@5	Random − No train	+3.45 pp	0.094
	Active − No train	+6.29 pp	0.047
	Active − Random	+2.84 pp	0.066

D.5 Sensitivity study

D.5.1 Ensemble size

We check the sensitivity of the task selection to ensemble size, using the MMLU dataset without agent attack. We fix the task pool of 50, select the top-10 tasks using different ensemble sizes and check the overlap ratio between the selected top-10 tasks and the reference top-10 tasks obtained with ensemble size 20. We repeat this experiment nine times with different initial ensembles and task pools, and error bars denote one standard deviation across repeated runs. For a baseline, a random

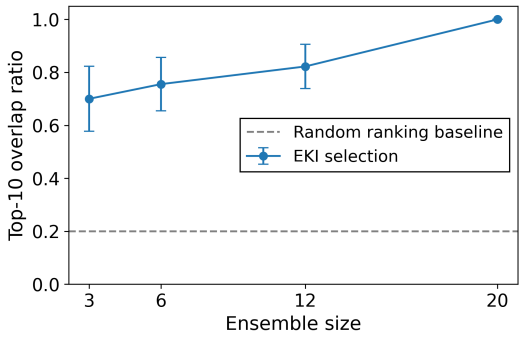

Figure 8: Sensitivity to ensemble size: Top-10 overlap with the ensemble-size-20 ranking (set as reference). The dashed horizontal line denotes the expected top-10 overlap ratio under a random ranking, equal to $10/50 = 0.2$.

top-10 ranking from the 50-task pool would have an expected overlap ratio of $10/50 = 0.2$ with the reference top-10 set. The overall results can be found in Fig. 8.

As expected, the overlap ratio increases with ensemble size, indicating that larger ensembles lead to more stable task identification. At the same time, even a relatively small ensemble already recovers most of the tasks selected by the larger-ensemble reference. In particular, ensemble size 6 achieves an average overlap ratio of around 0.75. This does not mean that only 7.5 out of the 10 selected tasks are truly valuable. Some tasks ranked just below the top 10 under ensemble size 20 may still be informative and can be selected when using a smaller ensemble. Therefore, the overlap ratio should be interpreted as a measure of agreement with the reference selection, rather than a strict count of useful tasks. Overall, the results suggest that **EKI-based task selection remains effective even with a small ensemble**, while larger ensembles further improve stability.

Note that we use ensemble size 20 as a larger-budget reference to study the stability trend, rather than to define a ground-truth ranking. Our goal here is not to fully characterize convergence with respect to ensemble size, but to examine whether a very small ensemble already provides effective task identification at substantially lower cost. This is consistent with prior observations in the Bayesian experimental design literature Callahan et al. (2025) that accurate inner-loop inference is not always necessary for useful EIG estimation, and that small inner sample sizes can substantially reduce computation while preserving practical utility.

D.5.2 EKI ITERATION STEPS

We further check the sensitivity of the task selection to the EKI iteration step, using the MMLU dataset without agent attack. We fix the task pool of 50, select the top-10 tasks using different EKI iteration steps and check the overlap ratio between the selected top-10 tasks and the reference top-10 tasks obtained with three EKI steps. We repeat this experiment nine times with different initial ensembles and task pools, and error bars denote one standard deviation across repeated runs.

The overall results are shown in Fig. 9. The tasks selected by one-step EKI and three-step EKI exhibit a high degree of overlap: on average, 8–9 out of the top-10 selected tasks are the same. We further plot the KL trajectories over EKI steps for all 50 tasks in Fig. 10. The top-ranked tasks maintain consistently larger KL values than the remaining tasks across EKI steps. Their rankings at step 1 and step 3 are also generally consistent, as shown in Fig. 11.

These results indicate that (i) **task selection is robust to the choice of the EKI iteration step**, and (ii) **one-step EKI is sufficient to identify the most informative tasks**.

D.6 COMPARISON WITH OTHER ACTIVE LEARNING METHODS

In this section, we introduce how we implement other active learning methods and how we balance the selection cost. For EGL and EMC, the computational cost is largely determined by the number of forward samples used to approximate the expectation in the REINFORCE estimator. In our

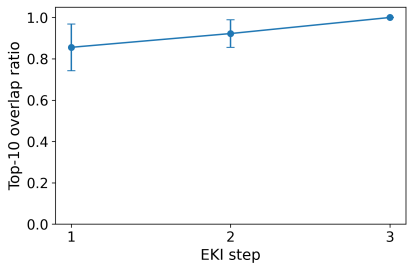

Figure 9: Top-10 overlap with the final three-step ranking as a function of the number of EKI iterations.

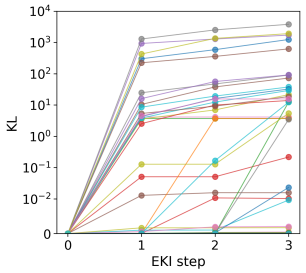

Figure 10: KL trajectory over EKI iteration steps for one representative pool.

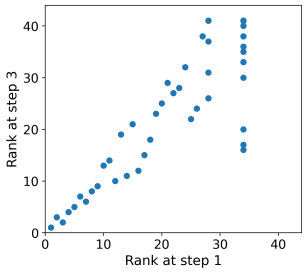

Figure 11: Rank comparison between step 1 and step 3 for one representative pool.

experiments, we choose this sampling budget such that the per-task gradient-based score estimation remains affordable, which makes direct enumeration over the 50-task pool feasible, without the need for Thompson sampling. For the Fisher-based methods, enumerating the same pool also yields a pool-level Fisher matrix, which enables a greedy subset selection strategy that partially accounts for batch interactions rather than selecting tasks independently.

D.6.1 EKI TOTAL ROLLOUTS

Based on the setup described in B.5, we use EKI to measure information gain for 25 tasks out of 50 and choose the top-10. Since each task evaluation involves 6 ensemble members, the total number of rollouts is $6 \times 25 = 150$. We keep the number of rollouts for the other methods the same.

D.6.2 EGL-BASED SELECTION

Expected gradient length quantifies task informativeness by the magnitude of the gradient, that is, by how strongly a task is expected to change the graph parameters. We use pointwise estimates rather than full expectations for the same reason as discussed in Section 4.2. In our setting, this gradient is the derivative of the communication-structure optimization utility with respect to the graph logits, estimated by policy gradient (REINFORCE). This is the same type of gradient used for graph optimization:

$$h = \nabla_{\mathbf{z}} \mathbb{E}_A \, \mathbb{E}_{\{q_i, y_i\} \in \mathcal{D}_{\text{tr}}} \big[\phi(g, q_i, y_i) \big]. \tag{30}$$

For task selection, we score each candidate task by the ℓ_2 norm of its estimated gradient,

$$U(q_i) = \|h_i\|_2, \tag{31}$$

and then rank all candidate tasks by this score.

In the optimization implementation of AgentPrune, the expectation over A is typically approximated using only one sample, which greatly reduces computational cost but also leads to a noisy gradient estimate. In our EGL implementation, we use 3 rollout samples for each task and enumerate all 50 tasks in the representative pool. Therefore, the selection uses a total of 150 rollouts.

D.6.3 EMC-BASED SELECTION

Expected model change quantifies task informativeness by the magnitude of the parameter change induced after several gradient-descent steps. For each candidate task, we start from the same initial graph, perform a short single-task training procedure, and measure how much the graph logits change before and after optimization.

Let $\mathbf{z}^{\text{before}}$ and $\mathbf{z}^{\text{after}}$ denote the flattened graph logits before and after the short training procedure, respectively. We define the task score as

$$U(q_i) = \|\Delta \mathbf{z}_i\|_2, \qquad \Delta \mathbf{z}_i = \mathbf{z}_i^{\text{after}} - \mathbf{z}_i^{\text{before}}. \tag{32}$$

We then rank all candidate tasks by this score and select the top ones.

In our implementation, we use one rollout sample in the REINFORCE gradient estimator for each update step, but run 3 gradient-descent steps for each task. The optimizer is the same as in training, namely Adam with learning rate 0.1. The selection uses a total of 150 rollouts.

D.6.4 FISHER-BASED SELECTION

Formula. The Fisher-based method quantifies task informativeness through a rollout-level empirical Fisher matrix. Fisher information measures how sensitively the model response changes with respect to the parameters, and is widely used as a proxy for the amount of information that an observation provides about those parameters. In our setting, each rollout on a candidate task induces a gradient with respect to the graph logits, and the corresponding empirical Fisher matrix captures the magnitude and directional structure of these task-induced updates.

For each candidate task q_i, suppose we obtain m rollout gradients $h_{i,1}, h_{i,2}, \ldots, h_{i,m}$ with respect to the graph logits. We construct the empirical Fisher matrix of this task as

$$F_i = \frac{1}{m} \sum_{r=1}^{m} h_{i,r} h_{i,r}^\top. \tag{33}$$

Relationship to EGL. It should be noted that when $m = 1$, the task-wise Fisher matrix reduces to

$$F_i = h_i h_i^\top, \tag{34}$$

so that

$$\mathrm{tr}(F_i) = \|h_i\|_2^2. \tag{35}$$

Therefore, for single-rollout single-task scoring, the trace criterion is closely related to EGL, whose score is proportional to $\|h_i\|_2$, and differs only by a monotone transformation. In this sense, the two criteria induce the same ranking over tasks in the single-rollout case.

It should also be noted that, in this case, the determinant of the Fisher matrix is zero whenever the graph-parameter dimension is greater than one, since Eq. (34) is a rank-one matrix. As a result, the determinant of a single-task Fisher matrix does not provide a meaningful ranking criterion in the single-rollout setting.

With multiple rollouts, the empirical Fisher matrix becomes an average of gradient outer products across rollouts, which incorporates second-moment information beyond the averaged-gradient magnitude used in EGL. However, this trace-based or determinant-based score remains closely related in spirit to EGL, and in our view does not constitute a sufficiently distinct task-selection criterion to warrant a separate standalone comparison.

Greedy subset selection. To address this limitation, we do not score each task independently and then directly take the top-k. Instead, we perform greedy subset selection so as to construct a subset whose accumulated Fisher information better covers the representative pool. First, we build a baseline Fisher matrix by averaging the task-wise Fisher matrices over the representative pool:

$$B_0 = \frac{1}{N} \sum_{i=1}^{N} F_i + \lambda I, \tag{36}$$

where $N = 50$ is the representative-pool size and λ is a small regularization constant.

Starting from $B = B_0$, at each greedy step we add one task that gives the largest marginal gain. In the trace-based version, the score of candidate task q_i is

$$U_{\mathrm{trace}}(q_i) = \mathrm{tr}(B^{-1} F_i). \tag{37}$$

In the determinant-based version, the score is

$$U_{\det}(q_i) = \log\det(B + F_i) - \log\det(B). \tag{38}$$

After selecting a task, we update

$$B \leftarrow B + F_i, \tag{39}$$

and repeat this procedure until 10 tasks are selected.

We use 3 rollout samples per task and enumerate 50 tasks. This selection uses a total of 150 rollouts.

The current formulation is closely related to A-optimality and D-optimality in Bayesian experimental design. For convenience, we refer to this baseline as *Coreset* in the figures. More precisely, it is a coreset-style greedy subset-selection method based on task-wise empirical Fisher information, designed to select a compact subset that better covers the representative pool.

Table 12: Numerical comparison of different task-selection baselines on MMLU under agent attack.

METHOD	MEAN ACCURACY (%)	STD. (%)
NO TRAIN	75.68	2.10
RANDOM TRAIN	76.87	3.72
FISHER CORESET (TRACE)	76.25	1.17
EGL	77.02	1.93
EMC	77.21	2.58
EKI	78.32	1.80
FISHER CORESET (DETERMINANT)	78.59	1.91

D.6.5 DETAILED STATISTICS FOR FIG. 5

Table 12 reports the numerical mean accuracy and standard deviation corresponding to the baseline comparison shown in Fig. 5.

The Fisher coreset comparison should be interpreted under the fixed selection-cost budget used in our experiments. We do not claim that task-wise EKI scoring generally dominates a fully optimized Fisher coreset method. Instead, the comparison reflects a practical cost trade-off. Fisher coreset construction requires information from the entire 50-task pool, so under the same total selection cost, each task-level Fisher matrix is estimated with a limited rollout budget and may be noisy. In contrast, the EKI-based method avoids explicit pool-level coreset construction and directly estimates task-level utility. This may lead to a more reliable task-level informativeness estimate in the cost-limited black-box setting considered here.

Table 13 reports bootstrap 95% confidence intervals for the auxiliary baseline comparison, using random training as the common reference. The results show that EKI and Fisher det have the largest observed improvements over random training, while EGL, EMC, and Fisher trace show smaller or negative observed differences. We interpret this comparison as descriptive evidence of the overall trend, rather than as a significance-ranked ordering among methods.

Overall, these results suggest that EKI provides competitive performance in the cost-limited setting.

Table 13: Bootstrap 95% confidence intervals for mean-accuracy differences in the baseline comparison. Improvements are computed relative to random training.

METHOD	IMPROVEMENT	95% BOOTSTRAP CI
NO TRAIN	-1.19	$[-3.28, 0.83]$
FISHER CORESET (TRACE)	-0.62	$[-2.46, 1.24]$
EGL	+0.16	$[-1.96, 2.34]$
EMC	+0.34	$[-1.96, 2.99]$
EKI	+1.45	$[-0.38, 3.34]$
FISHER CORESET (DETERMINANT)	+1.72	$[-0.45, 3.99]$

D.6.6 QUALITATIVE COMPARISON BETWEEN EKI AND FISHER-DETERMINANT SELECTION

Although EKI and Fisher-determinant selection achieve similar downstream performance, they characterize task informativeness differently. EKI measures the realized finite change in the graph-parameter distribution induced by a task. In contrast, the Fisher matrix describes local parameter sensitivity, and the Fisher-determinant coreset further accounts for redundancy and coverage across the candidate pool. As a result, the two methods can assign substantially different priorities to individual tasks.

We illustrate this difference using four representative MMLU tasks:

- **Task 13719 [virology_test]:** How is the parvovirus family targeted to reduce disease? Correct answer: *screen transfusion blood*.

- **Task 12110 [high_school_microeconomics_test]:** Which good likely has the most elastic demand curve? Correct answer: *white Ford minivans*.
- **Task 8221 [econometrics_test]:** Which factors could cause autocorrelated residuals (slow response, over-reaction, omitted autocorrelated variables, or outliers)? Correct answer: *i, ii, iii*.
- **Task 11587 [high_school_world_history_test]:** A question based on a passage from Felipe Guaman Poma de Ayala, *The First New Chronicle and Good Government* (ca. 1610), asking which change in Spanish policy toward Native Americans resulted. Correct answer: *royal decrees for more humane treatment*.

Table 14: Representative task-level diagnostics comparing EKI-based information gain and Fisher-determinant coreset selection. KL rank is the rank according to the EKI-based information gain.

Task index	KL rank	KL value	Prediction variance	Cosine	Fisher-det rank
13719	2	13.31	30.00	0.56	44
12110	9	3.82	72.52	0.81	12
8221	28	1.91	53.99	0.94	1
11587	40	0	7.68×10^{-10}	1.00	35

Note: The values differ from those in Table 6 because the two tables report results from different random-seed runs.

The examples show several distinct cases. Task 13719 is ranked second by EKI but only 44th by the Fisher-determinant coreset. Its relatively low update-direction cosine (0.56) indicates a distinctive parameter update, resulting in a large change in the parameter distribution and therefore a high EKI information gain. In contrast, Task 8221 is ranked first by the Fisher-determinant coreset but only 28th by EKI. This illustrates the pool-level nature of coreset selection: a task can be valuable because its local information complements that of other tasks even when its individual EKI information gain is moderate.

Task 12110 is ranked highly by both methods, showing that the two criteria can also agree on informative tasks. Task 11587 provides the opposite example: its negligible predictive variation produces essentially zero EKI information gain, and neither method assigns it a high priority. Overall, these examples show that EKI emphasizes task-induced finite changes in the graph-parameter distribution, whereas the Fisher-determinant coreset additionally emphasizes complementary coverage of the candidate pool.

The two methods also differ computationally. Fisher-based selection requires gradients with respect to the graph parameters, whereas EKI only requires forward evaluations. For a dense N-agent graph with $O(N^2)$ graph parameters, explicitly storing a dense Fisher matrix requires $O(N^4)$ memory, whereas EKI stores an ensemble with memory $O(JN^2)$. Thus, EKI is particularly attractive when gradients are unavailable or noisy, or when storing a dense Fisher matrix becomes expensive.

D.7 Ablation studies

We conduct an ablation study on MMLU under the adversarial setting to isolate the contributions of the two key components in our active learning pipeline: representative selection and informative selection. The full method first selects a 50-task representative pool from the 1000-task candidate set, and then chooses 10 tasks from this pool using the proposed informativeness criterion. To understand the role of each component, we remove one component at a time while keeping the overall training budget and evaluation protocol unchanged. Specifically, "Representative" keeps the representative-selection stage but replaces informative selection with random choice from the representative pool. "Informative" replaces the representative pool with a random 50-task subset from the 1000-task candidate set, and then applies informative selection to choose 10 tasks. "Representative + Informative" denotes the full method.

As shown in Table 15, the three ablated variants exhibit a clear pattern. Representative selection alone does not lead to a consistent improvement over the baselines, while informative selection alone already provides a noticeable gain. The full method, which combines representative and informative selection, achieves the best mean accuracy and the smallest variability across runs.

These results indicate that informative selection is the primary source of performance improvement. This is because the informative stage directly selects tasks that are estimated to be beneficial for the downstream objective from the 50-task pool. As long as the pool contains sufficiently useful tasks, informative selection can extract them even when the pool itself is constructed randomly, which helps explain the improved minimum accuracy of the "Informative" setting. At the same time, without representative selection, the 50-task pool may provide weaker coverage of the broader candidate set and may contain more tasks with overlapping update directions, which can limit the strongest downstream outcomes.

While informative selection accounts for most of the gain, representative selection still provides a measurable additional benefit when combined with it, and is crucial for computational tractability. Its role is not only to improve accuracy, but also to make the informative stage practical. Compared with using a random 50-task subset before informative selection, using representative selection to construct the 50-task pool leads to better final performance. This shows that the representative stage does not merely reduce the pool size, but meaningfully improves the quality of the candidate tasks passed to the informative stage. In addition, this reduction is crucial for efficiency: applying informative selection directly to the full candidate pool would be computationally impractical. The representative stage thus plays a dual role, serving both as a quality-improving filter and as an efficiency-enabling step for the subsequent informative selection.

Table 15: Ablation study on different training setups. Accuracy is reported as mean ± std, together with the minimum and maximum values across runs.

SETUP	VARIANT	ACCURACY / (%)	MIN	MAX
NO TRAIN		75.68 ± 2.10	70.58	79.08
RANDOM TRAIN		76.87 ± 3.72	68.62	82.35
	REPRESENTATIVE SELECTION ONLY	75.74 ± 3.64	70.58	81.04
ACTIVE LEARNING	INFORMATIVE SELECTION ONLY	77.12 ± 2.45	72.54	79.73
	REPRESENTATIVE + INFORMATIVE	78.32 ± 1.80	75.81	82.35

E SCOPE AND GENERALIZABILITY

E.1 FAILURE MODES OF INFORMATION-GAIN-BASED TASK SELECTION

To address potential concerns about whether high information gain translates into downstream utility, we identify three possible failure modes and discuss how they are mitigated in our framework. First, an atypical or outlier task may exhibit high information gain without being broadly useful. Our representative-selection stage reduces this risk by restricting informative selection to tasks that represent non-negligible regions of the task distribution. Second, different task groups may favor conflicting communication structures. Our formulation follows the shared-graph setting of the adopted graph optimizer and therefore assumes that the task pool is not so heterogeneous that different groups require fundamentally different graphs. When distinct task groups do require different communication structures, the same task-selection framework can instead be applied within each group to identify informative tasks for group-specific graph optimization. Third, noisy evaluations may spuriously inflate the estimated information gain. This is primarily an estimation issue, for which the ensemble-based update provides a stable estimate under a limited evaluation budget, as further examined in Appendix D.5.

E.2 SCOPE

Our goal in this paper is not to propose a new LLM-MAS architecture or a new graph optimization algorithm, but to study active task selection for communication-structure optimization. Accordingly, our method is designed as a modular layer on top of an existing communication-graph optimization pipeline: it evaluates candidate tasks according to how informative they are for updating the communication structure, while treating the underlying multi-agent system and graph optimizer as black boxes. In this sense, the proposed framework is not inherently tied to a specific number of

agents, a particular LLM backbone, or the REINFORCE-based graph optimization method used in our experiments.

More broadly, the proposed framework requires a parameterized communication structure that can be evaluated under different graph parameters, together with a task-dependent scalar or vector performance score. Under these conditions, the proposed EKI-based utility estimator can be applied without differentiating through either the LLM-MAS or the graph optimizer.

Our active task-selection layer is separate from the communication-structure optimization layer: it identifies informative training tasks, while the selected tasks can in principle be used with different graph optimizers. Therefore, the proposed active learning framework is not tied to the REINFORCE-based optimizer adopted in our experiments. Empirical evaluation with other communication-structure optimization methods is left for future work.

Finally, newly obtained information can affect communication-structure optimization at two different levels. First, within each MAS rollout, we keep the communication graph fixed. In multi-step tasks, intermediate environment feedback may change subsequent agent messages or actions, but the graph itself is not re-optimized during the rollout. Second, after a batch of informative tasks is selected and used to optimize the graph, the graph parameters change, and the tasks that are most informative for further optimization may also change. Our task-selection procedure can therefore be reapplied to the updated graph to select the next batch of training tasks. In this work, we focus on fixed-graph execution within each rollout and single-round task selection, leaving within-rollout graph adaptation and multi-round task selection for future study.