Efficient Evaluation of Multi-Task Robot Policies With Active Experiment Selection

Abrar Anwar, Rohan Gupta, Zain Merchant, Sayan Ghosh, Willie Neiswanger, Jesse Thomason
University of Southern California,

Abstract: Evaluating learned robot control policies to determine their performance costs the experimenter time and effort. As robots become more capable in accomplishing diverse tasks, evaluating across all these tasks becomes more difficult as it is impractical to test every policy on every task multiple times. Rather than considering the average performance of a policy on a task, we consider the *distribution* of performance over time. In a multi-task policy evaluation setting, we actively model the distribution of robot performance across multiple tasks and policies as we *sequentially* execute experiments. We show that natural language is a useful prior in modeling relationships between tasks because they often share similarities that can reveal potential relationships in policy behavior. We leverage this formulation to reduce experimenter effort by using a cost-aware information gain heuristic to efficiently select informative trials. We conduct experiments on existing evaluation data from real robots and simulations and find a 50% reduction in estimates of the mean performance given a fixed cost budget. We encourage the use of our surrogate model as a scalable approach to track progress in evaluation. Code can be found at: github.com/AbrarAnwar/seq-eval

Keywords: Robot Evaluation, Active Testing, Language

1 Introduction

With the growth of large-scale robot datasets and pretrained policies, robot systems have become increasingly capable of carrying out a wide variety of tasks; however, this diversity makes evaluating these policies increasingly challenging. The combinatorial growth makes an exhaustive evaluation even more impractical. Language-guided manipulation [1, 2, 3] and navigation [4, 5, 6] approaches continue to improve. As such, there is a need for maintaining estimates of policy performance and efficient evaluation strategies that can enable systematic and scalable testing of multi-task robot policies in the real world. Unlike fields such as computer vision or natural language processing, physical robotics experiments are conducted sequentially, and each policy rollout requires significant experimenter time and effort. Our paper actively estimates the performance of a set of policies over tasks, and then uses this framework to explore cost-aware, informative experiment sampling.

In practice, experimenters are typically interested in selecting the best checkpoints, tuning hyperparameters, or comparing model architectures, which do not necessarily require a full evaluation across every policy-task combination. A robot policy that can "pick up an apple" is likely capable of "picking up an orange" in an otherwise similar scene. Our work explores this insight and considers the structural relationships between tasks by framing robot evaluation as a population parameter estimation problem. This formulation then lets us design efficient, active experiment sampling strategies.

When evaluating a robot policy, it is common to consider only *average-case* performance. However, robot performance often has high variance, so we instead consider the evaluation of a policy on a specific task as understanding the performance *distribution*. How do we learn these performance distributions in an effective and efficient manner?

To operationalize this problem, we characterize every policy-task pair by a parameterized distribution reflecting the experiment conditions. For example, we use a Bernoulli distribution to model performance for tasks with binary reward and a Gaussian distribution for a continuous reward. As an experimenter conducts evaluations sequentially, we learn a surrogate model that estimates parameters for the performance distribution of every policy-task pair.

To build an efficient evaluation strategy, we leverage the shared structure between tasks. As we sample new experiments, we learn a surrogate model conditioned on latent task and policy embeddings. We show that better representations of tasks, namely from language priors, improve estimates of the outcome distributions, indicating that there is shared information between tasks learnable from policy performance. Though surrogate models have been used in robotics to predict outcomes in human-robot interaction scenarios [7], they did not consider the cost of evaluating each scenario.

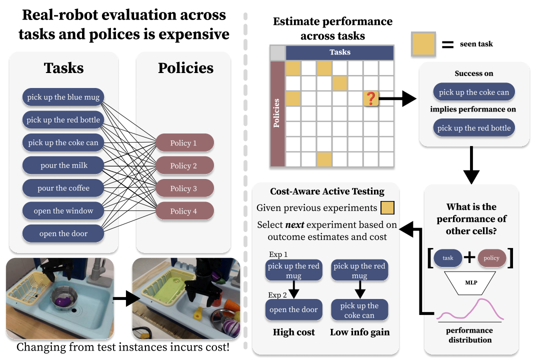

Figure 1: **Overview.** Exhaustively evaluating multiple robot policies across various tasks has high experimenter cost. In this work, we leverage latent relationships between tasks and policies to model performance distributions across all tasks and policies. These estimates are updated sequentially and used to implement cost-aware active experiment selection strategies.

Since evaluation is expensive, we want to minimize the cost of evaluation while still estimating the performance of all policies across all tasks of interest. Then, with our surrogate model, we leverage strategies from the active learning literature to integrate cost-efficient sampling heuristics.

2 Background and Related Work

Evaluation in Machine Learning. In computer vision and NLP, it is common to characterize the out-of-distribution performance of a single model [8, 9, 10, 11, 12, 13, 14] or create standards for comparing different models [12]. These approaches allow experimenters to quickly compare between models. However, in robotics, policy evaluation is difficult since each task is expensive to evaluate. We use methods from active learning to improve experiment selection during evaluation.

Active Testing. Similar to active learning, which aims to select training labels, active testing approaches [15, 16, 17] focus on selecting test instances to evaluate to better predict model performance. Though these settings focus on classification or regression labeling tasks, this formulation is important to robotics because evaluation is expensive. Various Bayesian optimization, active learning, and active testing approaches use surrogate models to estimate the value of a training or test instance [18, 19, 20, 21, 22, 23], often incorporating cost-aware sampling [24, 25]. In robotics, surrogate models have been used to predict outcomes of a human-robot interaction scenarios in simulation for policy learning [7]; however, this past work did not consider the cost evaluating each scenario. Since robot evaluation can have high variance, we take inspiration from past work [26] to focus on active learning of probabilistic models using a surrogate model.

Evaluation of Robot Policies. The goal of robot evaluation is to compare policies and gain insight into their behavior. Simulated evaluation [27, 28, 29, 30] is a common policy testing method, but often poorly correlates with real-world performance [31, 32]. We therefore focus on real-robot evaluation, which is costly and noisy. Recent efforts include selecting initial conditions [33], evaluating LLM-based planners [34], actively assessing black-box symbolic planners [35, 36, 37], or bounding policy performance using outcome distributions [38]. Other work examines how initial condition changes affect sensitivity [39, 40, 41, 42] or use these factors to guide data collection [43]. We instead actively evaluate multi-task policies and learn their underlying performance distributions.

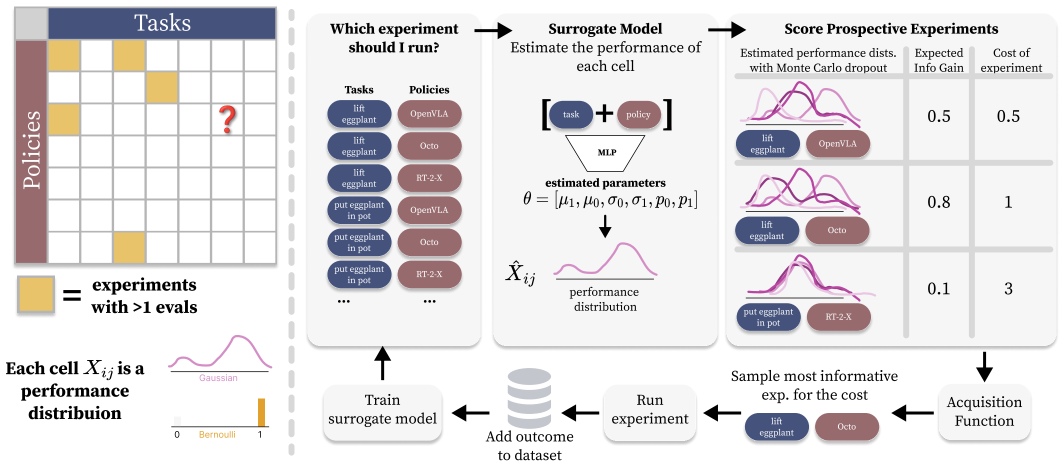
Tasks
Policies
?
= experiments with >1 evals
Each cell X_{ij} is a performance distribuion
Gaussian
Bernoulli
0
1
Which experiment should I run?
Tasks
Policies
lift eggplant
OpenVLA
Octo
RT-2-X
put eggplant in pot
Surrogate Model
Estimate the performance of each cell
task
policy
MLP
estimated parameters
$\theta = [\mu_1, \mu_0, \sigma_0, \sigma_1, p_0, p_1]$
$\hat{X}_{ij}$
performance distribution
Score Prospective Experiments
Estimated performance dists. with Monte Carlo dropout
Expected Info Gain
Cost of experiment
0.5
0.5
0.8
1
0.1
3
Acquisition Function
Sample most informative exp. for the cost
Run experiment
Add outcome to dataset
Train surrogate model

Figure 2: **Method.** We build a surrogate parameter estimation model that learns task and policy embeddings to predict the outcome performance distribution for each policy on every task. We use Bernoulli distributions for binary outcomes or a bimodal Gaussian for continuous outcomes. Given this parameter estimation model, we develop an active testing strategy with cost-aware sampling based on expected information gain.

3 Problem Formulation and Notation

The objective of this work is to design an efficient strategy to evaluate robot policies across tasks while balancing the cost of experimentation. Consider a fixed set of M robot policies, denoted by $\mathcal{P} = \{\pi_1, \pi_2, \ldots, \pi_M\}$ and a set of N tasks $\mathcal{T} = \{T_1, T_2, ..., T_N\}$. Each task $T_j \in \mathcal{T}$ is a finite-horizon MDP defined by states, actions, and a high-level natural language instruction L_i. Our framework is policy-agnostic, does not assume access to policy model weights, and can be applied to engineered robot *systems* in addition to end-to-end models.

Population Parameter Estimation. We formulate the problem as population parameter estimation, similar to probabilistic matrix factorization [44]. Let the performance of a policy $\pi_i \in \mathcal{P}$ on a task $T_j \in \mathcal{T}$ be represented by the random variable X_{ij} with distribution P_{ij}, from which we can sample evaluations $x_{ij} \sim P_{ij}$. Here, P_{ij} represents the "true" performance distribution. Since the underlying distribution P_{ij} is unknown, the goal of population parameter estimation is to estimate a distribution Q_{ij} that models real-world evaluation outcomes from P_{ij}. We use θ_{ij} to represent the parameters of the learned distribution Q_{ij}. For example, $\theta_{ij} = [\mu, \sigma]$ if Q_{ij} is a Gaussian distribution. Given a limited number of observed samples from the true distribution, $x^1_{ij}, ..., x^n_{ij} \sim P_{ij}$, the goal is to estimate the parameters of an estimated distribution θ_{ij}. Our setting also has samples from other random variables, X_{kl} corresponding to different policy-task pairs. Therefore, in this work we want to estimate $\Theta = \{\theta_{ij}\}^{i=M,j=N}_{i,j=1}$ for all policy-task pairs given a dataset $\mathcal{D} = \{x^k_{ij}\}$. These distributions can be visualized as a grid of policy-task pairs as shown in Figure 2.

The aim is to estimate the parameters of Q_{ij} of all policy-task combinations by leveraging shared information across this matrix. However, it is infeasible to directly evaluate all policy-task pairs due to cost constraints. Therefore, we adopt an active testing approach, where the objective is to iteratively select the most informative experiments (π_i, T_j) to efficiently learn Θ.

Active Testing. We apply an active learning paradigm to learn a population parameter estimator $f(\pi_i, T_j)$. As such, we define acquisition functions to guide the selection of task-policy pairs or tasks alone, and then sample experiments that are most informative. First, we define an acquisition function $a(\pi_i, T_j)$, and the next experiment is selected by maximizing this function over all possible experiments: $(\pi^*_i, T^*_j) = \arg\max_{(\pi_i, T_j)} a(\pi_i, T_j)$. Although these acquisition functions are informative, we want a balance between selecting informative experiments and their costs.

Evaluation Cost. In real-world evaluation, each policy-task evaluation incurs a cost. Let $c_{\text{eval}}(T_j)$ denote the cost of a single evaluation of a policy on task T_j. We make a simplifying assumption that this cost is agnostic to changes in the policy under evaluation. This cost could include the policy execution time, the resources consumed during evaluation, or the manual work to reset the scene. Furthermore, switching between tasks typically incurs a larger cost involving reconfiguring the scene or robot. We define this switching cost $c_{\text{switch}}(T_j, T_k)$ as the cost associated with transitioning from task T_j to T_k. For a sequence of tasks that have been evaluated $T_{i_1}, \ldots, T_{i_L}$ (where each $i_j \in N$), we compute the total cost of evaluation $c_{\text{total}} = \sum_{j=1}^{N} c_{\text{eval}}(T_{i_j}) + \sum_{j=1}^{N-1} c_{\text{switch}}(T_{i_j}, T_{i_{j+1}})$. Given these costs, the problem is to design an evaluation strategy that minimizes the total cost of evaluation while learning the population parameters of test instances.

4 Method

We design a framework for estimating the performance of robot policies across tasks by using a surrogate model conditioned on task and policy representations. We then use this sequentially-learned surrogate model to inform cost-aware sampling of experiments using information gain.

4.1 Surrogate Model

As we evaluate our robot policies across tasks, we track the outcomes of each trial to aggregate a dataset $\mathcal{D}$ over time. Each of these outcomes are realizations of a true underlying distribution P_{ij}. Our goal is to learn a surrogate model from $\mathcal{D}$ that predicts the population parameters θ_{ij} of a performance distribution Q_{ij}. As more evaluation rollouts are conducted, we add the outcomes to $\mathcal{D}$ and update the surrogate model. To train an effective surrogate model f, we use notions of similarity between tasks and policies. Thus, we need a representation that captures the similarities between policies and tasks with respect to their performance distributions. We define a policy embedding e_{π_i} and task embedding e_{T_j}, where similar performance distributions in task and policy can be captured based on the embeddings. These policy and task representations are then provided as input to an MLP that predicts the estimated population parameters: $\hat{\theta}_{ij} = f(\pi_i, T_j) = \text{MLP}(e_{\pi_i}, e_{T_j})$.

Task and Policy Representation. To define the task and policy embeddings e_{π_i}, e_{T_j}, we design various types of embeddings. In practice, we cannot know the relationship between policies in advance while we are conducting evaluation. Therefore, we define the policy embedding to be a fixed, randomly initialized embedding to act as an identifier for the policy in a given experiment. For the task embedding e_{π_i}, we leverage language embeddings from `MiniLMv2` [45] which we reduce to 32 dimensions using PCA over all tasks.

Population Parameter Estimation. Outcomes in robot learning can take the form of continuous values like rewards, time to completion, or task progress, and binary values like task success. Thus, the underlying distribution from the surrogate model depends on the type of task. We consider two types of underlying distributions. When X_{ij} is continuous, Q_{ij} takes the form of a mixture of Gaussians with K components, $\hat{x}_{ij} \sim Q_{ij} = \sum_{k=1}^{K} p_k \mathcal{N}(\mu_k, \sigma_k)$, where π_k, μ_k, and σ_k are the mixing coefficients, means, and standard deviations of the Gaussian components respectively that are predicted from the surrogate model $\theta_{ij} = f(\pi_i, T_j)$. We thus train the surrogate model with a mixture density loss [46, 47] to minimize the negative log-likelihood of the observed data under the mixture model. In our experiments on continuous outcome distributions, we use $K = 2$ Gaussian components because of the intuition that robot policy performance is often bimodal; robots either fail catastrophically or they maintain non-zero performance. In the case where X_{ij} is binary, indicating success or failure, Q_{ij} takes the form of a Bernoulli distribution, where $\theta_{ij} = \{p \in [0, 1]\}$ is represented by the surrogate model trained using cross-entropy loss.

4.2 Cost-aware Active Experiment Selection

We explore cost-aware, active-experiment acquisition functions that guide selection of experiments based on their expected utility while considering associated costs. To define the acquisition function, we first focus on how to measure the informativeness of a policy-task evaluation, which we capture through expected information gain.

Expected Information Gain. Expected Information Gain (EIG) quantifies the value of an experiment by estimating how much it reduces the predictive uncertainty of the performance distribution for a policy-task pair. Since the surrogate model estimates performance *distributions*, we define the EIG of a policy-task pair using a Bayesian Active Learning by Disagreement (BALD) [48] formulation for probabilistic models [26]:

$$\mathcal{I}(\pi_i, T_j) = \underbrace{\mathbb{H}[Q_{ij}]}_{\text{marginal entropy}} - \underbrace{\mathbb{E}_{\theta_{ij} \sim f(\theta_{ij}|\mathcal{D})}[\mathbb{H}[Q_{ij}|\theta_{ij}]]}_{\text{expected conditional entropy}}. \tag{1}$$

The first term represents the marginal entropy over Q_{ij}, which quantifies the total uncertainty in Q_{ij}. The second term corresponds to the expected conditional entropy over multiple samples of parameters θ_{ij}. Thus, $\mathcal{I}(\pi_i, T_j)$ captures the disagreement between multiple samples of distributions. For example, if 10 pairs of parameters for a Gaussian result in very different distributions, then their disagreement will be high. Because the entropy of a mixture of Gaussians generally lacks a closed-form solution, we estimate the entropy by discretizing the empirical distribution into $n = 25$ bins for which to compute entropy over. BALD ensures the EIG score is higher in test instances where there is disagreement in the predicted distributions across sampled parameters. In this case, we define the acquisition functions $a(\pi_i, T_j) = \mathcal{I}(\pi_i, T_j)$. Computing EIG requires multiple samples of Θ_{ij}; however, we only train a single MLP. Based on past literature [49, 50, 51], we apply dropout only at test-time to compute multiple samples of θ_{ij} from the surrogate model $f(\cdot)$.

Cost-Aware EIG. While EIG quantifies the informativeness of an experiment, it does not consider the costs of conducting evaluation. To make EIG cost-aware, we design the following acquisition function based on prior work that simply integrates cost with a multiplicative factor [25, 24]:

$$a_{\text{cost-aware}}(\pi_i, T_j, T_{\text{current}}) = \frac{\mathcal{I}(\pi_i, T_j)}{(\lambda \cdot c_{\text{switch}}(T_{\text{current}}, T_j)) + 1}, \tag{2}$$

where $\mathcal{I}(\pi_i, T_j)$ represents EIG for the policy π_i on task T_j, $c_{\text{switch}}(T_{\text{current}}, T_j))$ is the cost of switching from current task T_{current} to new task T_j, and λ is a cost sensitivity hyperparameter.

Active Experiment Selection. We use this acquisition function to iteratively sample experiments (see Algorithm 1 in Appendix B). To mitigate the cold-start problem in active learning, we initialize the dataset $\mathcal{D}$ with a single randomly-selected task, for which every policy is evaluated 3 times. We then train the surrogate model on this data. At each query step, the acquisition function $a(\pi_i, T_j)$ is computed for all policy-task pairs. To compute the entropy over model parameters for the EIG metric, we use MC dropout to sample 10 predicted outcome distributions. To balance exploration and exploitation, we use an epsilon-greedy strategy with a rate of $\epsilon = 0.1$. The selected experiment (π_i, T_j) is then executed 3 times, and the observed outcomes are added to the dataset $\mathcal{D}$. We found in preliminary experiments that 3 trials per selected experiment was often better for cost-efficient population parameter estimation. Given these new outcomes in the dataset, we keep training the surrogate model on the updated dataset to improve its predictions over time.

5 Experiments

To evaluate our active testing framework, we leverage evaluations that have already been conducted in simulated and the real world, which we then sample offline. We evaluate our method using four offline datasets, with more details in Appendix C. **HAMSTER** [52] provides evaluations of a VLA model and four baselines across 81 tasks, each with a single continuous outcome, which we model as a Gaussian with fixed variance. **OpenVLA** [2] includes 4 policies tested on 29 tasks across two embodiments; we incorporate switching costs for both task and embodiment changes. **MetaWorld Policies** [53] simulates 50 manipulation tasks with 10 diverse policies trained via different architectures and state noise; we collect 100 rollouts per policy-task pair and use both binary (success/fail) and continuous (reward) metrics. Switching cost reflects object-level differences between tasks. **MetaWorld Checkpoints** tracks a single policy through 11 training checkpoints. Each of these datasets can be modeled with different underlying distributions and have varying costs, semantic diversity, and skills.

5.1 Task and Policy Representation

Experiment Design and Baselines. As the ideal task or policy representation is unclear, we compute an upper bound by training learnable policy and task embeddings using all pre-evaluated outcomes to predict performance. We describe this in more detail in Appendix A.1. These **Optimal** embeddings are tuned specifically for this task but require full data access *a priori*.

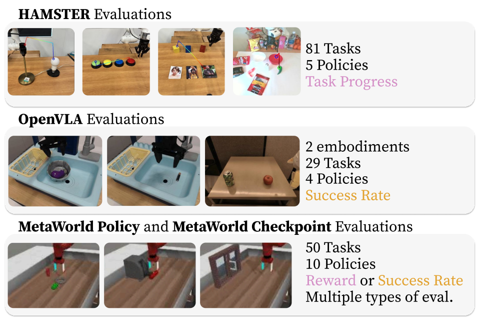

Figure 3: **Offline Datasets used for Experiments.** We consider 4 settings of offline evaluations, as denoted above.

To study the effect of different embeddings, we separately design representations for tasks and policies. As we describe in Section 4.1, we use language embeddings as a task representation. However, we found that language embeddings overly focus on nouns as opposed to verbs, which causes issues as actions with similar nouns but different verbs would be closer together than verbs with the same nouns. Thus, we apply the following procedure to mitigate this issue. We (1) use part-of-speech tagging to extract all verbs and verb phrases, (2) compute a language embedding for the verb $e_{T_j}^{\text{verb}}$ and for the entire task description $e_{t_j}^{\text{task}}$, and then (3) compute the task embedding $e_{T_j} = 0.8 \cdot e_{T_j}^{\text{verb}} + 0.2 \cdot e_{T_j}^{\text{task}} + 0.1 \cdot \mathcal{N}(0,1)$. We also found that the embeddings were often too close across multiple tasks, and we found that adding a slight noise term helped separate close embeddings. We call weighted language feature representation **Verb**.

We compare it to a standard **Language** embedding for the instruction and a **Random** embedding baselines. Unlike a task representation through language, there is no canonical policy representation. We use **Optimal** and **Random** embeddings, and leave broader exploration to future work. We run all experiments over 750 steps across three seeds. Each experiment is sampled similar to how researchers typically evaluate: we select a random task and test each policy three times. To assess each representation's impact, we compute the average log-likelihood of all outcomes in our offline dataset against a probability distribution represented by the predicted population parameters

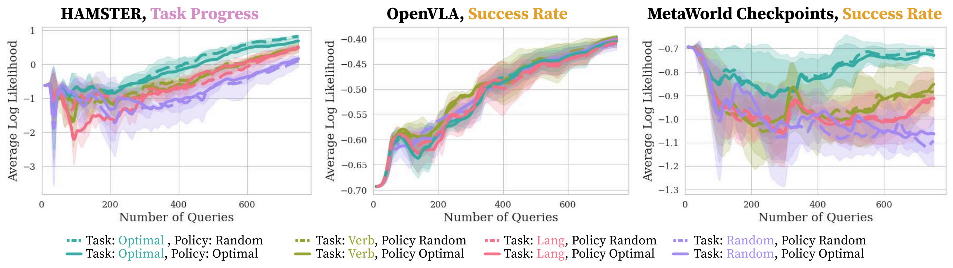

Figure 4: **Task and Policy Representation Experiments.** We compute the average log likelihood of all outcomes under probability distribution represented by the predicted population parameters across various policy and task representations. We evaluate these methods over three offline evaluation datasets over continuous and binary performance distributions. We find no large difference between random or optimal embeddings as a policy representation, indicating that there is not much shared information between policies. However, we find that for task representation, **Optimal** consistently performs the best, followed by **Verb**, then **Lang**, and lastly **Random**. Language-based embeddings is a good task representation that we can leverage for better active testing.

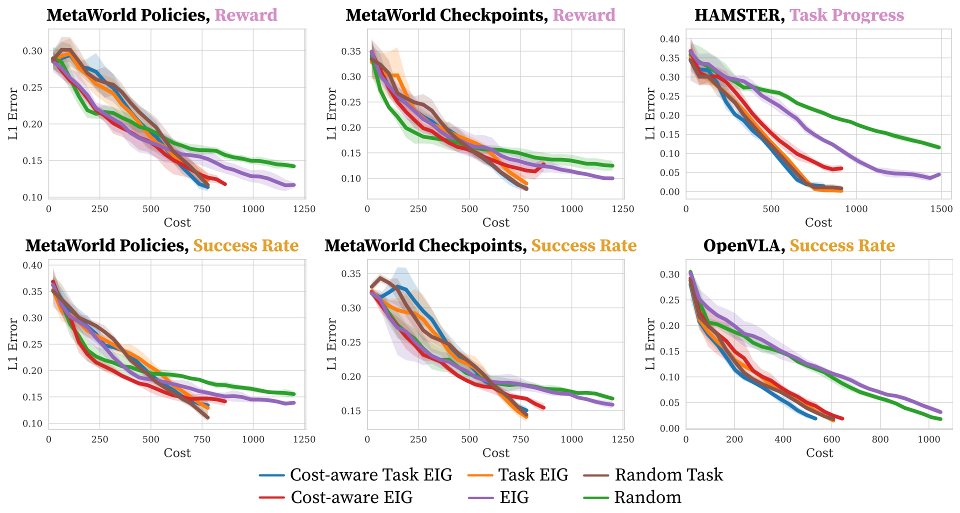

Figure 5: **Average L1 Error of the Mean Over Cost.** We compute the error between the ground truth means of a policy-task pair and the mean of the predicted performance distribution. EIG-based methods better estimate the means for both continuous and binary distributions. Task sampling methods are more cost-efficient than policy-task sampling methods at similar log-likelihoods.

from the surrogate model. Details on the baselines and how we train the **Optimal** policy and task representations can be found in Appendix D.

Random representations do not share information across policies and tasks. Our results indicate that random embeddings for tasks or policies consistently perform worse, as they fail to capture any meaningful structure or shared information between tasks. The increasing performance of **Random** is due to new experiments being sampled; however, minimal interpolation of outcomes occurred.

Task types impact performance estimation. The HAMSTER evaluations consist of many changes to objects rather than changes to the type of task itself such as "pickup the milk..." and "pickup the shrimp...". This structure leads to clearer benefits when using language-based representations. In contrast, OpenVLA has less separable tasks, thus it shows a much smaller separation between random, optimal, and language-based embeddings. MetaWorld Checkpoints, however, show a more stable improvement of **Verb** as opposed to simply **Lang** since there are many more tasks.

Optimal task and policy representations. Despite the improvement from using **Lang** or **Verb**, they do not fully bridge the gap in performance to optimal task embeddings. Thus, language-based representations do not represent all shared information between tasks. Additionally, the use of optimal policy embeddings did not provide additional sample-efficiency, likely due to over reliance on task embeddings during training or the limited number of policies in the dataset.

5.2 Cost-Aware Experiment Selection

To evaluate the effectiveness of our cost-aware active experiment selection methods, we assess the population parameter estimation capability of our framework across various datasets using continuous and binary performance distributions.

Sampling Strategies. We compare two families of strategies: (1) selecting a policy-task pair, and (2) selecting a task and evaluating all policies $d = 3$ times. For pairwise selection, we use: **Random Sampling**, which samples uniformly; **EIG**, which selects the pair with highest expected information gain (EIG, see Section 4.1); and **Cost-aware EIG**, which accounts for task-switching cost via Equation 2. For task-based sampling, we use: **Random Task**, sampling tasks uniformly; **Task EIG**, selecting the task with highest total EIG; across policies and **Cost-aware Task EIG**, which maximizes sum total cost-aware EIG across policies. The task-based sampling strategies are more

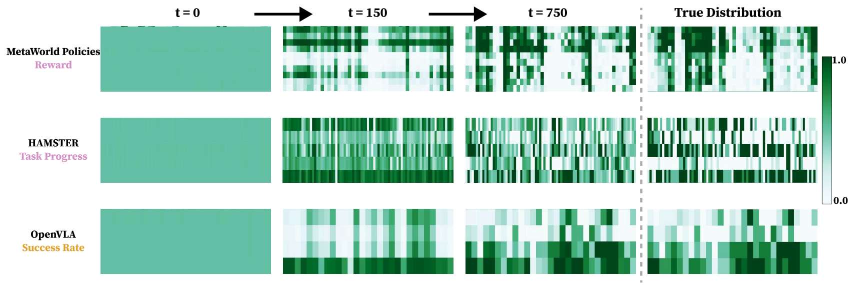

Figure 6: **Predicted Mean Distributions.** We provide a visualization of the means for the predicted continuous and binary distributions over 0, 150, and 750 sampled queries. We use random sampling with 3 evaluations per policy-task pair to show that our surrogate model can actively learn the full distribution of performance and learn the performance distribution over time. For example, for MetaWorld Policies at $t = 750$, $750/3 = 250$ policy-task pairs were sampled of the total $50 * 10 = 500$ possible policy-task pairs that could be evaluated, the estimated mean performance is qualitatively comparable to the true mean; Figure 5 reports these results quantitatively as L1 error.

realistic to how experimenters evaluate their robots today, as experimenters typically select a task and then evaluate every policy. All methods are run for 1500 steps across three seeds with **Random** policy and **Verb** task embeddings. We report the L1 distance between the true and estimated means derived from the estimated population parameters per policy-task pair (see Appendix E for details).

EIG-based approaches better estimate mean performance, but struggle to learn population parameters. Figure 5 shows EIG methods consistently outperform baselines in estimating the mean, often at lower cost. For example, for OpenVLA at a cost of 500, Cost-Aware Task EIG has an L1 error of 0.017 while random task has double that error at 0.036. This results in an approximate **50% reduction in error** when using EIG-based sampling for the same cost. We also note that qualitatively, we find that in the middle of an evaluation sequence, EIG-based approaches outperform random sampling baselines. This result is likely due to estimates converging as a larger number of policy-task pairs have been sampled at least once. Figure 7 shows the average log-likelihood of offline outcomes against the predicted population parameters of the model. EIG-based methods slightly outperform random baselines in fitting the full distribution, with stronger gains in some cases (e.g., MetaWorld Policies with success rate), though results vary across datasets. This result suggests that learning the performance distribution in a cost-effective manner remains challenging, especially in the early stages of evaluation when data is sparse.

Tradeoffs between task- and policy-task sampling. Both Figure 5 and Figure 7 show that task-based sampling is generally better in OpenVLA and HAMSTER, but cost-aware EIG generally estimates the L1 error better than its task-based counterpart on MetaWorld. Policy-task sampling approaches are likely more efficient in MetaWorld experiments as there are a large number of experiments where there is a high cost to switch, and evaluating 10 policies over a single task may not be as informative. In contrast, HAMSTER and OpenVLA have fewer policies, meaning the cost of evaluating all policies for a single task is lower. Additionally, we found that policy-task sampling methods are more likely to switch tasks, causing a faster accumulation of cost.

Learning the Performance Landscape. Figure 6 illustrates how our formulation of sequentially sampling experiments refines the predictions of the performance landscape. Early in the evaluation process, predictions tend to center around the mean and are misaligned with the true distribution. As more experiments are selected, the estimates begin to resemble the true means. Even if evaluation cost is not a constraint, the surrogate model is a scalable way to track evaluation progress and monitor trends during the evaluation process.

Concluding Statement. By framing robot evaluation as an active testing problem, we investigate the relationships between tasks to predict policy performance distributions. The surrogate model not only informs cost-aware sampling but also serves as a scalable tool for tracking evaluation progress. We hope this ability enables more informed decision-making in the robot development lifecycle.

6 Limitations

Cost-effective performance distribution estimation. While our approach better estimates the mean performance of policy-task pairs (see Section 5.2), the surrogate model does not learn the parameters of the performance distribution in a more cost-effective manner than random sampling, as reflected in the lower log-likelihoods in Figure 7. This is likely because cost-aware strategies favor repeating low-cost experiments. As a result, task coverage decreases, and the model sees fewer diverse or uncommon items. We focused on ensuring that the surrogate model is able to estimate the landscape of performance across tasks and policies at low cost, but in practice, experimenters care about policy comparisons. Our framework can be combined with concurrent work on optimal stopping for experiments during policy comparisons [54], or focus on other applications such as finding the best average policy, finding a ranked ordering of policies, or finding the worst performing tasks. Each of these would require different active sampling strategies.

Cost-aware, myopic experiment selection. We represented robot execution costs naively at a fixed cost; however, different tasks may have different execution costs that may depend on whether a policy fails on its task or not, such as having to clean up spilled milk. When execution or switching costs are non-uniform, single-step look-ahead is typically not sufficient for cost-aware experiment selection. More optimal cost-aware solutions must *plan* future evaluations with respect to cost and potential information gain. Future work can extend our methods by developing myopic, look-ahead algorithms that can select longer sequences of experiments at a time. If we can myopically estimate the most informative tasks at lower costs, we hope our work can act as a learned reward function [55] that encourages autonomous, reset-free, multi-task reinforcement learning in the real world.

Language-based task representations. For computing our language representations, the design of **Verb** involved a weighted sum between verb and full-instruction embeddings. We found this heuristic term to perform better, and the weightings of these terms to not impact performance. Most language embeddings emphasize objects, as they act like bag-of-words models, but verbs are more indicative of the task in robotics. This finding motivates future work in learning robotics-specific *task* representations that are grounded in actions and are available *a priori*. There are also hierarchical

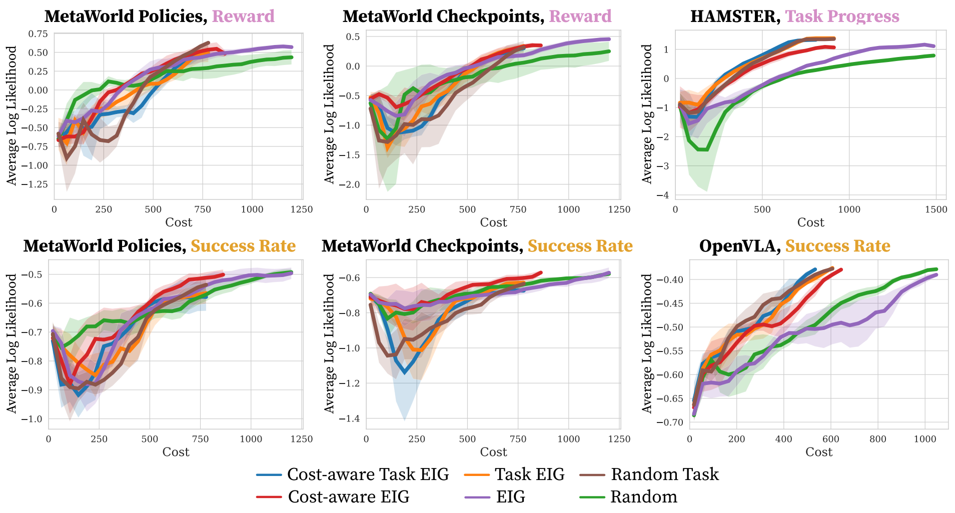

Figure 7: **Average Log Likelihood Over Cost.** We show the average log likelihood of all the outcomes in our offline dataset against the cost of evaluation for MetaWorld Policies, MetaWorld Checkpoints, HAMSTER, and OpenVLA over continuous and binary performance distributions. Each set of experiments is run for 1500 trials. We find that EIG-based approaches struggle to model the true distribution in a more cost-efficient manner than Random Task sampling. Task-based sampling strategies are more cost-efficient than policy-task approaches.

relationships between tasks such as “pour milk” likely depending on being able to “pick up the milk” that this work does not consider.

Policy representation. We used simple random or optimal embeddings for policy representations and found minimal differences between the two, but learning policy embeddings may better predict performance. Policy embedding priors might be formed by encoding the training data of those policies or their predictions to offline data.

Acknowledgments

This work was supported in part by a grant from the Army Research Lab (ARL) Army AI Innovations Institute (A2I2), award number W911NF-23-2-0010. The claims and findings of this work do not necessarily represent the views of the ARL.

References

[1] Octo Model Team, D. Ghosh, H. Walke, K. Pertsch, K. Black, O. Mees, S. Dasari, J. Hejna, C. Xu, J. Luo, T. Kreiman, Y. Tan, L. Y. Chen, P. Sanketi, Q. Vuong, T. Xiao, D. Sadigh, C. Finn, and S. Levine. Octo: An open-source generalist robot policy. *Robotics: Science and Systems (RSS)*, 2024.

[2] M. Kim, K. Pertsch, S. Karamcheti, T. Xiao, A. Balakrishna, S. Nair, R. Rafailov, E. Foster, G. Lam, P. Sanketi, Q. Vuong, T. Kollar, B. Burchfiel, R. Tedrake, D. Sadigh, S. Levine, P. Liang, and C. Finn. Openvla: An open-source vision-language-action model. *Conference on Robot Learning (CoRL)*, 2024.

[3] K. Black, N. Brown, D. Driess, A. Esmail, M. Equi, C. Finn, N. Fusai, L. Groom, K. Hausman, B. Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[4] D. Shah, B. Osiński, S. Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. *Conference on Robot Learning (CoRL)*, 2023.

[5] D. Shah, A. Sridhar, N. Dashora, K. Stachowicz, K. Black, N. Hirose, and S. Levine. Vint: A foundation model for visual navigation. *Conference on Robot Learning (CoRL)*, 2022.

[6] A. Anwar, J. Welsh, J. Biswas, S. Pouya, and Y. Chang. Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation. *International Conference on Robotics and Automation (ICRA)*, 2025.

[7] V. Bhatt, H. Nemlekar, M. C. Fontaine, B. Tjanaka, H. Zhang, Y.-C. Hsu, and S. Nikolaidis. Surrogate assisted generation of human-robot interaction scenarios. *Conference on Robot Learning (CoRL)*, 2023.

[8] D. Wang, N. Ding, P. Li, and H.-T. Zheng. CLINE: Contrastive Learning with Semantic Negative Examples for Natural Language Understanding. *Association for Computational Linguistics (ACL)*, 2021.

[9] D. Hendrycks, X. Liu, E. Wallace, A. Dziedzic, R. Krishnan, and D. Song. Pretrained transformers improve out-of-distribution robustness. *Association for Computational Linguistics (ACL)*, 2020.

[10] B. Recht, R. Roelofs, L. Schmidt, and V. Shankar. Do imagenet classifiers generalize to imagenet? In *International Conference on Machine Learning (ICML)*, 2019.

[11] D. Hendrycks and T. Dietterich. Benchmarking neural network robustness to common corruptions and perturbations. *International Conference on Learning Representations (ICLR)*, 2019.

[12] P. Liang, R. Bommasani, T. Lee, D. Tsipras, D. Soylu, M. Yasunaga, Y. Zhang, D. Narayanan, Y. Wu, A. Kumar, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research (TLMR)*, 2022.

[13] Y. Chang, X. Wang, J. Wang, Y. Wu, L. Yang, K. Zhu, H. Chen, X. Yi, C. Wang, Y. Wang, et al. A survey on evaluation of large language models. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 2024.

[14] M. Gardner, Y. Artzi, V. Basmov, J. Berant, B. Bogin, S. Chen, P. Dasigi, D. Dua, Y. Elazar, A. Gottumukkala, N. Gupta, H. Hajishirzi, G. Ilharco, D. Khashabi, K. Lin, J. Liu, N. F. Liu, P. Mulcaire, Q. Ning, S. Singh, N. A. Smith, S. Subramanian, R. Tsarfaty, E. Wallace, A. Zhang, and B. Zhou. Evaluating models' local decision boundaries via contrast sets. *Findings of Empirical Methods in Natural Language Processing (EMNLP Findings)*, 2020.

[15] C. Sawade, N. Landwehr, S. Bickel, and T. Scheffer. Active risk estimation. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 951–958, 2010.

[16] T. Rainforth, A. Foster, D. R. Ivanova, and F. Bickford Smith. Modern bayesian experimental design. *Statistical Science*, 39(1):100–114, 2024.

[17] E. Yilmaz, P. Hayes, R. Habib, J. Burgess, and D. Barber. Sample efficient model evaluation. *arXiv preprint arXiv:2109.12043*, 2021.

[18] K. Eggensperger, F. Hutter, H. Hoos, and K. Leyton-Brown. Efficient benchmarking of hyperparameter optimizers via surrogates. In *Proceedings of the AAAI conference on artificial intelligence*, 2015.

[19] E. Brochu, V. M. Cora, and N. De Freitas. A tutorial on bayesian optimization of expensive cost functions, with application to active user modeling and hierarchical reinforcement learning. *arXiv preprint arXiv:1012.2599*, 2010.

[20] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.

[21] A. Cozad, N. V. Sahinidis, and D. C. Miller. Learning surrogate models for simulation-based optimization. *AIChE Journal*, 60(6):2211–2227, 2014.

[22] Z. Qian, C. C. Seepersad, V. R. Joseph, J. K. Allen, and C. Jeff Wu. Building surrogate models based on detailed and approximate simulations. *Journal of Mechanical Design*, 2006.

[23] J. Kossen, S. Farquhar, Y. Gal, and T. Rainforth. Active testing: Sample-efficient model evaluation. *International Conference on Machine Learning (ICML)*, 2021.

[24] E. H. Lee, V. Perrone, C. Archambeau, and M. Seeger. Cost-aware bayesian optimization. *arXiv preprint arXiv:2003.10870*, 2020.

[25] B. Paria, W. Neiswanger, R. Ghods, J. Schneider, and B. Póczos. Cost-aware bayesian optimization via information directed sampling. In *Adaptive Experimental Design and Active Learning in the Real World Workshop at ICML*, 2020.

[26] C. Tosh, M. Tec, and W. Tansey. Targeted active learning for probabilistic models. *arXiv preprint arXiv:2210.12122*, 2022.

[27] M. Deitke, W. Han, A. Herrasti, A. Kembhavi, E. Kolve, R. Mottaghi, J. Salvador, D. Schwenk, E. VanderBilt, M. Wallingford, L. Weihs, M. Yatskar, and A. Farhadi. RoboTHOR: An Open Simulation-to-Real Embodied AI Platform. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[28] P. Anderson, A. Shrivastava, J. Truong, A. Majumdar, D. Parikh, D. Batra, and S. Lee. Sim-to-real transfer for vision-and-language navigation. *Conference on Robot Learning (CoRL)*, 2021.

[29] A. Kadian, J. Truong, A. Gokaslan, A. Clegg, E. Wijmans, S. Lee, M. Savva, S. Chernova, and D. Batra. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[30] T. Gervet, S. Chintala, D. Batra, J. Malik, and D. S. Chaplot. Navigating to objects in the real world. *Science Robotics*, 2023.

[31] W. Pumacay, I. Singh, J. Duan, R. Krishna, J. Thomason, and D. Fox. THE COLOSSEUM: A Benchmark for Evaluating Generalization for Robotic Manipulation. *Robotics: Science and Systems (RSS)*, 2024.

[32] X. Li, K. Hsu, J. Gu, K. Pertsch, O. Mees, H. R. Walke, C. Fu, I. Lunawat, I. Sieh, S. Kirmani, S. Levine, J. Wu, C. Finn, H. Su, Q. Vuong, and T. Xiao. Evaluating real-world robot manipulation policies in simulation. *Conference on Robot Learning (CoRL)*, 2024.

[33] H. Kress-Gazit, K. Hashimoto, N. Kuppuswamy, P. Shah, P. Horgan, G. Richardson, S. Feng, and B. Burchfiel. Robot learning as an empirical science: Best practices for policy evaluation. *arXiv*, 2024.

[34] Z. Hu, F. Lucchetti, C. Schlesinger, Y. Saxena, A. Freeman, S. Modak, A. Guha, and J. Biswas. Deploying and Evaluating LLMs to Program Service Mobile Robots. *IEEE Robotics and Automation Letters (RA-L)*, 2024.

[35] P. Verma, S. R. Marpally, and S. Srivastava. Discovering user-interpretable capabilities of black-box planning agents. *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2021.

[36] P. Verma, R. Karia, and S. Srivastava. Autonomous capability assessment of sequential decision-making systems in stochastic settings. *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.

[37] R. K. Nayyar, P. Verma, and S. Srivastava. Differential assessment of black-box ai agents. *AAAI Conference on Artificial Intelligence*, 2022.

[38] J. A. Vincent, H. Nishimura, M. Itkina, P. Shah, M. Schwager, and T. Kollar. How Generalizable Is My Behavior Cloning Policy? A Statistical Approach to Trustworthy Performance Evaluation. *IEEE Robotics and Automation Letters (RA-L)*, 2024.

[39] A. Parekh, N. Vitsakis, A. Suglia, and I. Konstas. Investigating the Role of Instruction Variety and Task Difficulty in Robotic Manipulation Tasks. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.

[40] A. Xie, L. Lee, T. Xiao, and C. Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. *International Conference on Robotics and Automation (ICRA)*, 2024.

[41] A. Anwar, R. Gupta, and J. Thomason. Contrast sets for evaluating language-guided robot policies. *Conference on Robot Learning (CoRL)*, 2024.

[42] J. Gao, S. Belkhale, S. Dasari, A. Balakrishna, D. Shah, and D. Sadigh. A taxonomy for evaluating generalist robot policies. 2025.

[43] J. Gao, A. Xie, T. Xiao, C. Finn, and D. Sadigh. Efficient Data Collection for Robotic Manipulation via Compositional Generalization. *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[44] A. Mnih and R. R. Salakhutdinov. Probabilistic matrix factorization. *Conference on Neural Information Processing Systems (NeurIPS)*, 2007.

[45] Y. Gu, L. Dong, F. Wei, and M. Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

[46] C. M. Bishop. Mixture density networks. *Technical Report*, 1994.

[47] D. Ha and J. Schmidhuber. World models. *Conference on Neural Information Processing System (NeurIPS)*, 2018.

[48] N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel. Bayesian active learning for classification and preference learning. *arXiv preprint arXiv:1112.5745*, 2011.

[49] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. *International Conference on Machine Learning (ICML)*, 2016.

[50] A. Loquercio, M. Segu, and D. Scaramuzza. A general framework for uncertainty estimation in deep learning. *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[51] E. Ledda, G. Fumera, and F. Roli. Dropout injection at test time for post hoc uncertainty quantification in neural networks. *Information Sciences*, 2023.

[52] Y. Li, Y. Deng, J. Zhang, J. Jang, M. Memmel, C. R. Garrett, F. Ramos, D. Fox, A. Li, A. Gupta, and A. Goyal. Hamster: Hierarchical action models for open-world robot manipulation. *International Conference on Learning Representations (ICLR)*, 2025.

[53] T. Yu, D. Quillen, Z. He, R. Julian, K. Hausman, C. Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *Conference on Robot Learning (CoRL)*, 2020.

[54] D. Snyder, A. J. Hancock, A. Badithela, E. Dixon, P. Miller, R. A. Ambrus, A. Majumdar, M. Itkina, and H. Nishimura. Is your imitation learning policy better than mine? policy comparison with near-optimal stopping. *Robotics: Science and Systems (RSS)*, 2025.

[55] J. Zhang, Y. Luo, A. Anwar, S. A. Sontakke, J. J. Lim, J. Thomason, E. Biyik, and J. Zhang. Rewind: Language-guided rewards teach robot policies without new demonstrations. *Conference on Robot Learning (CoRL)*, 2025.

A Surrogate Model

A.1 Computing optimal representations

To compute the optimal embeddings, we take the MLP surrogate model, add a learnable task and policy embedding, and then supervise against its respective loss against all the evaluation data *a priori*. This approach quickly learns to estimate the performance distribution parameters, conditioned on the learnable embeddings. Then, once training has converged, we stop training, reset the surrogate model's weights, and freeze the policy and task embedding layers. This surrogate model is then used for actively learning the policy distribution parameters.

A.2 Surrogate model details

We actively train the surrogate model, which is a 2-layer MLP that takes in the policy and task embeddings and outputs the number of parameters. We train with a learning rate of 1e-4 and a weight decay of 1e-4. We train with a dropout of 10% and also use dropout during parameter sampling. For computing metrics such as L1 error, log-likelihoods, and others, we do not use dropout.

B Active Experiment Selection Procedure

Below, we provide the active experiment selection procedure in detail.

Algorithm 1 Active Experiment Selection Procedure

```
Require: A set of policies $\pi_i \in \mathcal{P}$ to evaluate over tasks $T_j \in \mathcal{T}$, an empty dataset of outcomes $\mathcal{D}$,
    an untrained surrogate model $f(\pi_i, T_j)$, exploration rate $\epsilon = 0.1$
 1: Randomly sample a single task $T_j$ and evaluate every policy 3 times. Add outcomes $x_{ij}^k$ to $\mathcal{D}$
 2: Set $T_{\text{current}} = T_j$
 3: Increment $C_{\text{total}} = C_{\text{total}} + c_{\text{eval}} \cdot |\mathcal{P}| \cdot 3$
 4: Train the surrogate model $f(\cdot)$ on $\mathcal{D}$ for $k$ epochs
 5: for each query step do
 6:     Use MC dropout to sample 10 predicted distributions from the surrogate model for every
        policy-task pair
 7:     Use sampled distributions to compute scores $s_{ij} = a(\pi_i, T_j, T_{\text{current}})$ according to Eq. 2
 8:     With probability $\epsilon$, select a random $(\pi_i, T_j)$
 9:     Otherwise, select $(\pi_i, T_j) = \arg\max_{(\pi_i, T_j)} s_{ij}$
10:     Conduct 3 evaluations and observe $x_{ij}^1, x_{ij}^2, x_{ij}^3 \sim P_{ij}$
11:     Add these outcomes to $\mathcal{D}$
12:     Train $f(\cdot)$ on $\mathcal{D}$ for $k$ epochs
13:     Increment $C_{\text{total}} = C_{\text{total}} + c_{\text{eval}} \cdot 3$
14:     if $T_j \neq T_{\text{current}}$ then                    ▷ Task switching cost applies
15:         Increment $C_{\text{total}} = C_{\text{total}} + c_{\text{switch}}(T_{\text{current}}, T_j)$
16:         Update $T_{\text{current}} = T_j$
17:     end if
18: end for
```

Experiment sampling. We run each policy-task pair three times, since preliminary experiments showed that estimating binary performance distributions required more trials to estimate the population parameters. Using 1 trial per policy-task selection led to more costly task switches with poor parameter estimations. Additionally, The OpenVLA evaluation dataset used 10 trials for each task, so we used 3 trials per policy-task pair as we could execute that experiment 3 times if needed.

Expected Information Gain Computation Though our approach to mitigating the cold-start problem with test-time dropout inspired by past work [50, 51] appears to have improved performance during sampling, this approach has not been rigorously tested by the Bayesian optimization community in particular. We had also tried other approaches, such as ensembling and variational prediction, but these approaches also overfit to the small size of the dataset early in the evaluation procedure.

C Offline Dataset Details

C.1 HAMSTER

We use evaluations from HAMSTER [52], which compares a hierarchical VLA model against 4 other policies across 81 diverse tasks with varying objects, complexity, and linguistic variation. Each policy-task pair is evaluated once using a continuous progress metric. We model each outcome as the mean of a Gaussian distribution with fixed variance. For HAMSTER, we have a cost of 0.5 per execution of an experiment, then an additional switching cost of +1 if a task is of the same task type but requires adding/removing objects. If a new task type is selected, we then add a cost of +2 for requiring new, often large, objects to be brought into the scene.

C.2 OpenVLA

From OpenVLA [2], we use evaluations of 4 policies across 29 tasks. Partial successes (e.g., 0.5) are rounded down to binary outcomes. We have a cost of 0.5 per execution of an experiment. If a task is changed, such as moving an eggplant to lifting a battery, a cost of 1 is applied. OpenVLA also has multiple embodiments available, Bridge and the Google Robot. If there is an embodiment change, we set the switching cost to 3, as this change is relatively large.

C.3 MetaWorld Policies/Checkpoints

MetaWorld [53] is an open-source simulated benchmark containing a set of 50 different manipulation environments for multi-task learning. With MetaWorld, we use both binary success and continuous reward normalized between 0 and 1.

For MetaWorld evaluation, we have a cost of 0.5 per execution of an experiment. In MetaWorld tasks, some tasks keep the same objects in the same scene such as opening or closing a window, while others would require new objects like a faucet or a door. Because these changes are easier to enumerate, we apply only a task switching cost of +1 if the primary object changes, and a switching cost of 0 in the case of the same object being manipulated.

We train 10 multi-task policies with varying architectures and noise to ensure diverse behaviors, and then evaluate 100 times in each environment to serve as an approximation of the true performance population distribution. For training these policies, we rollout an expert policy for 100 episodes for the 50 tasks to build our training set. We then train a state-based, language-conditioned behavior cloning policy. The policy takes in a 768-dimensional language embedding, a 39-dimensional state vector, and outputs a 4-dimensional action. For MetaWorld Checkpoints, we train a single MLP-based policy for 100 epochs, recording the policy performance at epoch $1, 10, 20, ..., 100$ for a total of 11 checkpoints. For MetaWorld Policies, we instead train 10 policies on random MLP architecture sizes and also apply different amounts of noise to the proprioceptive inputs to the policy to mimic a noisy understanding of state information. We do this procedure to produce policies that vary more in performance while still having a systematic "flaw" in understanding the scene, which we hope would be captured in our policy embeddings. Then, for each policy and environment, we sample 50 evaluations each and store them offline for sampling.

D Task and Policy Representation Experiment Design

We evaluate how different task and policy representations affect the quality of the surrogate model's predictions. Because each experiment involves both a task and a policy, we define embedding strategies for each separately. Below, we detail the representations evaluated in our experiments.

Task Representations. We define 4 different task representations.

- **Optimal:** Learned task embeddings trained to directly predict performance using all available data. These serve as an upper bound but are not feasible in real settings due to their reliance on full data access.
- **Verb:** Our primary method, which constructs a representation from a weighted combination of the task's instruction embedding and its extracted verbs. This captures both linguistic and action-related structure (see Section 4.1).
- **Language:** A baseline that uses only the sentence embedding of the full task description, without decomposition.
- **Random:** Randomly initialized vectors for each task. These break any meaningful structure and serve as a naive control.

Policy Representations. Since we cannot compute policy representations apriori, we use the two following approaches, and leave the question on discovering new policy representations to future work.

- **Optimal:** Learned policy embeddings trained using full data to predict outcomes. As with task embeddings, these are used only to establish a performance upper bound.
- **Random:** Random vectors assigned to each policy, used as a baseline in the absence of structured policy descriptors. We leave the design of more informed policy representations (e.g., based on architecture, behavior, or training data) to future work.

All embedding configurations are evaluated over 750 experiment steps, across three random seeds. In each step, a task is sampled uniformly at random, and all policies are evaluated three times on that task. To assess surrogate model quality, we compute the average log likelihood of all outcomes in the offline dataset under the predicted distribution derived from the model's estimated population parameters.

E Sampling Strategy Details

We explore two main families of sampling strategies for selecting experiments: (1) selecting a specific policy-task pair, and (2) selecting a task and evaluating all policies on that task. Below, we detail each method:

- **Random Sampling:** Select a policy-task pair (π_i, T_j) uniformly at random. Acquisition function: $a(\pi_i, T_j) = 1/(|\mathcal{P}| \times |\mathcal{T}|)$.
- **EIG:** Select the policy-task pair with the highest expected information gain (EIG), as described in Section 4.1. Acquisition function: $a(\pi_i, T_j) = \mathcal{I}(\pi_i, T_j)$.
- **Cost-aware EIG:** Incorporate task-switching costs by selecting the pair that maximizes cost-adjusted EIG. See Equation 2 for the full formulation.
- **Random Task:** Select a task T_j uniformly at random and evaluate all policies on it, $d = 3$ times each. Acquisition function: $a(T_j) = 1/|\mathcal{T}|$.
- **Task EIG:** Select a task by summing the EIG across all policies and choosing the task with the highest total information gain. Acquisition function: $a(T_j) = \sum_i \mathcal{I}(\pi_i, T_j)$.
- **Cost-aware Task EIG:** Like Task EIG, but incorporates cost-awareness by summing the cost-adjusted EIG across all policies relative to the current task. Acquisition function: $a(T_j) = \sum_i a_{\text{cost-aware}}(\pi_i, T_j, T_{\text{current}})$.

These strategies are evaluated over 1500 experiment steps across three random seeds. We use **Random** policy embeddings and **Verb** task embeddings throughout. Our main evaluation metric is the L1 error between the true mean outcome of a policy-task pair and the surrogate's predicted mean.

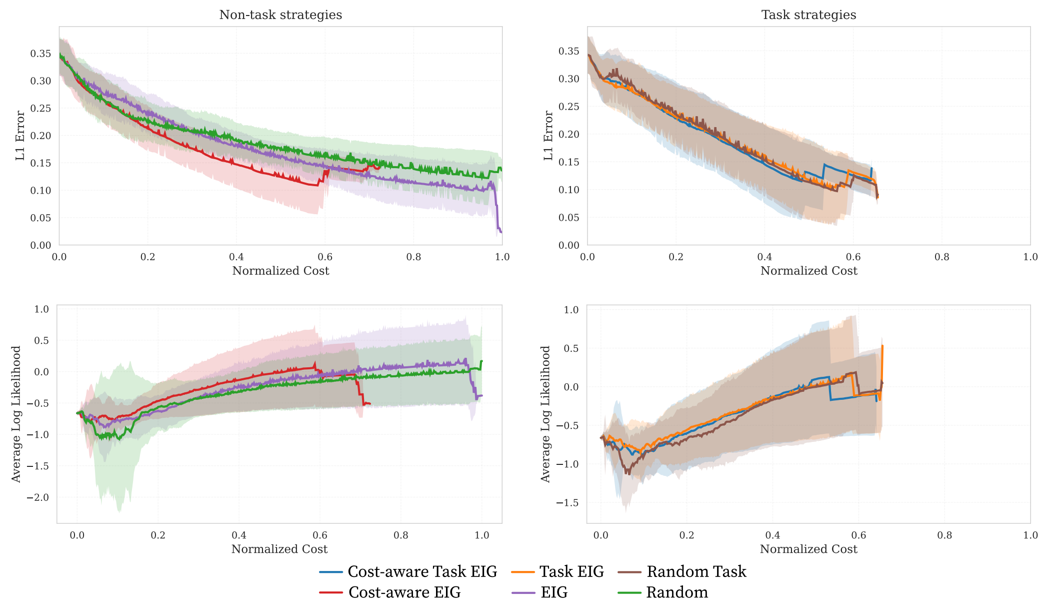

Figure 8: We average metrics across all datasets and split by sampling type. We also normalize the cost on the x-axis by the highest cumulative cost accrued in our evaluations. We find that for non-task-based sampling approaches, our cost-aware sampling is consistently more efficient, while random sampling is at a higher cost.

E.1 Relationship to probabilistic matrix factorization (PMF).

We model outcomes $x_{ij} \sim Q_{ij}$ where Q_{ij} is parameterized by distribution params θ_{ij}. This parallels to PMF, where ratings $r_{ij} \sim \mathcal{N}(\mu_i^T v_j, \sigma^2)$, but instead we use a neural network to estimate θ_{ij} rather than a linear inner product. We assume conditional independence of outcomes given these parameters.

E.2 Additional Results

We re-plot an averaged set of results across all offline datasets and for continuous and binary distributions in Figure 8. The disjoint jumps at the ends are typically due to the changing number of evaluations at certain costs (i.e. some runs finish at lower costs while others finish at higher costs, leading to higher values that are averaged). Cost-aware EIG consistently outperforms random and standard EIG, converging with approximately 150 fewer evaluations (raw cost of $\sim$ 900 vs $\sim$ 1200). For task-based sampling, the gap is not as large in aggregate, and Figure 5 shows improvements vary based on the dataset. Beyond raw estimation performance, our surrogate model can be used as a tool to track performance estimates *during* the evaluation process (as shown in Figure 6).