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.**

I. 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 [27, 18, 6] and navigation [34, 33, 3] 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. 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.

II. Background and Related Work

Active Testing. Similar to active learning, which selects informative training labels, active testing [32, 31, 42] focuses on selecting test instances to better estimate model performance—especially relevant in robotics where evaluations are costly. Surrogate models are commonly used to guide such selections [8, 35], often with cost-aware sampling [21, 29]. Prior work in robotics has applied surrogate models to simulate outcomes [4], but without addressing evaluation cost. We instead follow work on active learning for probabilistic models [37] to model outcome uncertainty directly.

Evaluation of Robot Policies. The goal of robot evaluation is to compare policies and gain insight into their behavior. Simulated evaluation [7, 1, 17, 12] is a common policy testing method, but often poorly correlates with real-world performance [30, 22]. We therefore focus on real-robot evaluation, which is costly and noisy. Recent efforts include selecting initial conditions [19], evaluating LLM-based planners [16], actively assessing black-box symbolic planners [38, 39, 26], or bounding policy performance using outcome distributions [40]. Other work examines how initial condition changes affect sensitivity [28, 41, 2, 11] or use these factors to guide data collection [10]. We instead actively evaluate multi-task policies and learn their underlying performance distributions.

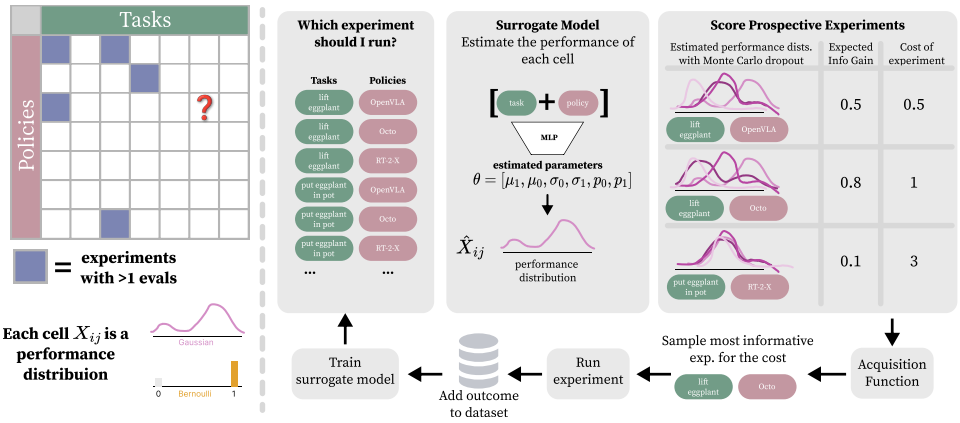

Fig. 1. **Method.** We build a surrogate parameter estimation model that learns task and policy embeddings to predict the outcome performance distribution for each policy on every task. We use Bernoulli distributions for binary outcomes or a bimodal Gaussian for continuous outcomes. Given this parameter estimation model, we develop an active testing strategy with cost-aware sampling based on expected information gain.

III. 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 [25]. 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, $x_{ij}^1, ..., x_{ij}^n \sim P_{ij}$, 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 $\Theta = \{\theta_{ij}\}_{i,j=1}^{i=M,j=N}$ for all policy-task pairs given a dataset $\mathcal{D} = \{x_{ij}^k\}$. These distributions can be visualized as a grid of policy-task pairs as shown in Figure 1.

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.

IV. 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.

A. Surrogate Model

As we evaluate robot policies across tasks, we collect outcomes in dataset $\mathcal{D}$, whose outcomes samples from the true distribution P_{ij}. Our goal is to learn a surrogate model f predicting population parameters θ_{ij} of the performance

distribution Q_{ij} from $\mathcal{D}$. With more rollouts, we update $\mathcal{D}$ and retrain the surrogate. To capture similarities, we define policy and task embeddings, e_{π_i} and e_{T_j}, which serve as inputs to an MLP predicting 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` [13] which we reduce to 32 dimensions using PCA over all tasks.

Population Parameter Estimation. Robot learning outcomes may be continuous (e.g., rewards, completion time) or binary (e.g., task success), so the surrogate model must represent different distribution types. For continuous outcomes, we model Q_{ij} as a K-component Gaussian mixture, $\hat{x}_{ij} \sim Q_{ij} = \sum_{k=1}^{K} p_k \mathcal{N}(\mu_k, \sigma_k)$, with mixing coefficients p_k, means μ_k, and standard deviations σ_k predicted by the surrogate model $\theta_{ij} = f(\pi_i, T_j)$. We train this model using a mixture density loss [5, 14] to minimize the negative log-likelihood of observed outcomes. In our experiments, we use $K = 2$ to reflect the bimodal nature of robot performance—policies typically either fail or achieve partial to full success. For binary outcomes, we model Q_{ij} as a Bernoulli distribution, where $\theta_{ij} = \{p \in [0, 1]\}$ is trained via cross-entropy loss.

B. 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) [15] formulation for probabilistic models [37]:

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

The first term in Eq. 1 is the marginal entropy of Q_{ij}, representing overall uncertainty, while the second is the expected conditional entropy given sampled parameters θ_{ij}. Their difference, $\mathcal{I}(\pi_i, T_j)$, measures disagreement across predicted distributions—e.g., if multiple Gaussian parameter samples yield diverse outcomes, the information gain is high. Since the entropy of a Gaussian mixture lacks a closed-form solution, we approximate it using a discretized empirical distribution with $n = 25$ bins. We adopt BALD to prioritize high-disagreement queries, setting $a(\pi_i, T_j) = \mathcal{I}(\pi_i, T_j)$. To sample θ_{ij} efficiently, we apply dropout at test time [9, 24, 20] using a single trained MLP surrogate.

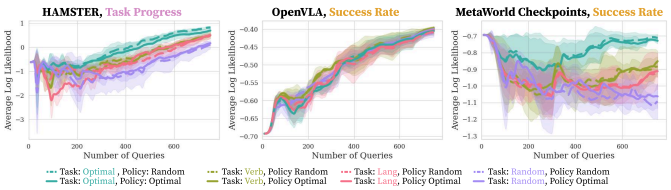

Fig. 2. **Task and Policy Representation Experiments.** We compute the average log likelihood of all outcomes under probability distribution represented by the predicted population parameters across various policy and task representations.

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 [29, 21]:

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

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 C). To avoid cold-start issues, we initialize $\mathcal{D}$ with one randomly chosen task evaluated 3 times across all policies, then train the surrogate model. At each step, we compute $a(\pi_i, T_j)$ for all pairs, using MC dropout to sample 10 predicted distributions. We apply an ϵ-greedy strategy ($\epsilon = 0.1$) to balance exploration and exploitation. The selected pair is evaluated 3 times, and outcomes are added to $\mathcal{D}$. We found 3 trials per query provided more reliable estimates. The surrogate is retrained continuously on the updated dataset to refine its predictions.

V. Experiments

We evaluate our active testing framework on four offline datasets (details in Appendix D). **HAMSTER** [23] includes 81 tasks with continuous outcomes modeled as Gaussians. **OpenVLA** [18] has 4 policies on 29 tasks across two embodiments, incorporating switching costs. **MetaWorld Policies** [43] covers 50 manipulation tasks with 10 policies, using binary and continuous metrics, with switching costs reflecting object differences. **MetaWorld Checkpoints** tracks one policy over 11 training checkpoints. These datasets vary in distributions, costs, and task diversity.

A. Task and Policy Representation

Experiment Design and Baselines. As the ideal task or policy representation is unclear, we compute an upper bound

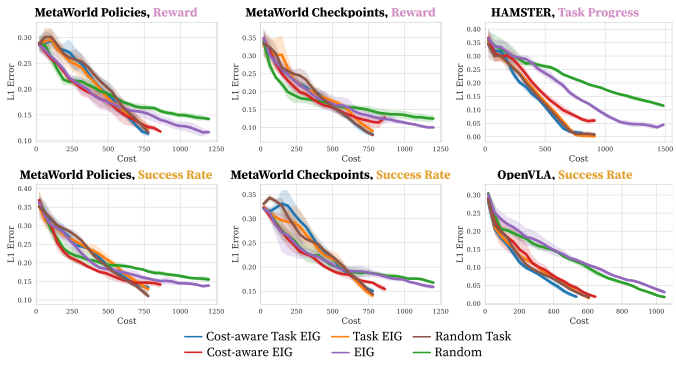

Fig. 3. **Average L1 Error of the Mean Over Cost.** EIG-based methods better estimate the means for both continuous and binary distributions. Task sampling methods are more cost-efficient than policy-task sampling methods at similar log likelihood.

by training learnable policy and task embeddings using all pre-evaluated outcomes to predict performance. We describe this in more detail in Appendix B-A. These **Optimal** embeddings are tuned specifically for this task but require full data access *a priori*. To study embedding effects, we design separate representations for tasks and policies. As discussed in Section IV-A, we use language embeddings for tasks. However, standard embeddings emphasize nouns over verbs, causing similar-noun actions with different verbs to be incorrectly grouped. To address this, we: (1) extract verbs via POS tagging, (2) compute embeddings for both the verb ($e^{\text{verb}}_{T_j}$) and full task ($e^{\text{task}}_{T_j}$), and (3) define $e_{T_j} = 0.8 \cdot e^{\text{verb}}_{T_j} + 0.2 \cdot e^{\text{task}}_{T_j} + 0.1 \cdot \mathcal{N}(0,1)$. We found that adding small noise helped differentiate similar tasks. We refer to this weighted representation as **Verb**.

We compare to standard **Language** and **Random** embeddings. Since there is no canonical policy representation, we use **Optimal** and **Random** embeddings, leaving further exploration for future work. Experiments run for 750 steps with three seeds, sampling by selecting a random task and testing each policy three times. To evaluate embeddings, we compute the average log-likelihood of outcomes under the surrogate model's predicted population parameters. More details on baselines and training for **Optimal** are in Appendix E.

Results. We find that random embeddings for tasks or policies consistently underperform, as they fail to encode meaningful structure or shared information. Although **Random** improves slightly as more experiments are sampled because the model can slowly better interpolate. The impact of representation is also task-dependent: HAMSTER, with its variations around object types (e.g., "pick up the milk" vs. "pick up the shrimp"), benefits significantly from language-based embeddings. In contrast, OpenVLA's task set is all more similar yields smaller differences between using language or not. Meanwhile, Metaworld Checkpoints, with wider task diversity, shows stronger gains from **Verb** over **Lang**.

B. Cost-Aware Experiment Selection

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 IV-A); 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 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 F for details).

We find EIG methods consistently outperform baselines in estimating mean performance, often at lower cost. For example, in OpenVLA, Cost-Aware Task EIG reduces L1 error by 50% over random sampling at the same budget. Improvements are most evident mid-way through evaluation since early stages lack coverage and late stages see all baselines converge. Fitting full performance distributions remains challenging: EIG slightly improves log-likelihoods over random, with dataset-dependent benefits. As shown in Figure 4, early predictions tend to center on the mean but improve with more data. Even without cost constraints, the surrogate model offers a scalable way to monitor and refine performance.

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.

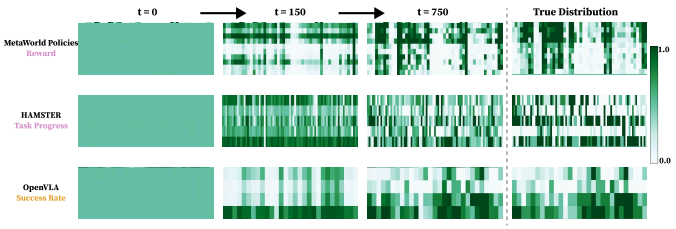

Fig. 4. **Predicted Mean Distributions.** We use random sampling with 3 evaluations per policy-task pair to show that our surrogate model can actively learn the full distribution of performance and learn the performance distribution over time. For example, for MetaWorld Policies at $t = 750$, $750/3 = 250$ 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.

REFERENCES

[1] Peter Anderson, Ayush Shrivastava, Joanne Truong, Arjun Majumdar, Devi Parikh, Dhruv Batra, and Stefan Lee. Sim-to-real transfer for vision-and-language navigation. *Conference on Robot Learning (CoRL)*, 2021.

[2] Abrar Anwar, Rohan Gupta, and Jesse Thomason. Contrast sets for evaluating language-guided robot policies. *Conference on Robot Learning (CoRL)*, 2024.

[3] Abrar Anwar, John Welsh, Joydeep Biswas, Soha Pouya, and Yan Chang. Remembr: Building and reasoning over long-horizon spatio-temporal memory for robot navigation. *International Conference on Robotics and Automation (ICRA)*, 2025.

[4] Varun Bhatt, Heramb Nemlekar, Matthew C Fontaine, Bryon Tjanaka, Hejia Zhang, Ya-Chuan Hsu, and Stefanos Nikolaidis. Surrogate assisted generation of human-robot interaction scenarios. *Conference on Robot Learning (CoRL)*, 2023.

[5] Christopher M Bishop. Mixture density networks. *Technical Report*, 1994.

[6] Kevin Black, Noah Brown, Danny Driess, Adnan Esmail, Michael Equi, Chelsea Finn, Niccolo Fusai, Lachy Groom, Karol Hausman, Brian Ichter, et al. π_0: A vision-language-action flow model for general robot control. *arXiv preprint arXiv:2410.24164*, 2024.

[7] Matt Deitke, Winson Han, Alvaro Herrasti, Aniruddha Kembhavi, Eric Kolve, Roozbeh Mottaghi, Jordi Salvador, Dustin Schwenk, Eli VanderBilt, Matthew Wallingford, Luca Weihs, Mark Yatskar, and Ali Farhadi. RoboTHOR: An Open Simulation-to-Real Embodied AI Platform. *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[8] Katharina Eggensperger, Frank Hutter, Holger Hoos, and Kevin Leyton-Brown. Efficient benchmarking of hyperparameter optimizers via surrogates. In *Proceedings of the AAAI conference on artificial intelligence*, 2015.

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

[10] Jensen Gao, Annie Xie, Ted Xiao, Chelsea Finn, and Dorsa Sadigh. Efficient Data Collection for Robotic Manipulation via Compositional Generalization. *Proceedings of Robotics: Science and Systems (RSS)*, 2024.

[11] Jensen Gao, Suneel Belkhale, Sudeep Dasari, Ashwin Balakrishna, Dhruv Shah, and Dorsa Sadigh. A taxonomy for evaluating generalist robot policies. 2025.

[12] Theophile Gervet, Soumith Chintala, Dhruv Batra, Jitendra Malik, and Devendra Singh Chaplot. Navigating to objects in the real world. *Science Robotics*, 2023.

[13] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Minillm: Knowledge distillation of large language models. In *International Conference on Learning Representations (ICLR)*, 2024.

[14] David Ha and Jürgen Schmidhuber. World models. *Conference on Neural Information Processing System (NeurIPS)*, 2018.

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

[16] Zichao Hu, Francesca Lucchetti, Claire Schlesinger, Yash Saxena, Anders Freeman, Sadanand Modak, Arjun Guha, and Joydeep Biswas. Deploying and Evaluating LLMs to Program Service Mobile Robots. *IEEE Robotics and Automation Letters (RA-L)*, 2024.

[17] Abhishek Kadian, Joanne Truong, Aaron Gokaslan, Alexander Clegg, Erik Wijmans, Stefan Lee, Manolis Savva, Sonia Chernova, and Dhruv Batra. Sim2Real Predictivity: Does Evaluation in Simulation Predict Real-World Performance? *IEEE Robotics and Automation Letters (RA-L)*, 2020.

[18] Moo Jin Kim, Karl Pertsch, Siddharth Karamcheti, Ted Xiao, Ashwin Balakrishna, Suraj Nair, Rafael Rafailov, Ethan Foster, Grace Lam, Pannag Sanketi, Quan Vuong, Thomas Kollar, Benjamin Burchfiel, Russ Tedrake, Dorsa Sadigh, Sergey Levine, Percy Liang, and Chelsea Finn. Openvla: An open-source vision-language-action model. *Conference on Robot Learning (CoRL)*, 2024.

[19] Hadas Kress-Gazit, Kunimatsu Hashimoto, Naveen Kuppuswamy, Paarth Shah, Phoebe Horgan, Gordon Richardson, Siyuan Feng, and Benjamin Burchfiel. Robot learning as an empirical science: Best practices for policy evaluation. *arXiv*, 2024.

[20] Emanuele Ledda, Giorgio Fumera, and Fabio Roli. Dropout injection at test time for post hoc uncertainty quantification in neural networks. *Information Sciences*, 2023.

[21] Eric Hans Lee, Valerio Perrone, Cedric Archambeau, and Matthias Seeger. Cost-aware bayesian optimization. *arXiv preprint arXiv:2003.10870*, 2020.

[22] Xuanlin Li, Kyle Hsu, Jiayuan Gu, Karl Pertsch, Oier Mees, Homer Rich Walke, Chuyuan Fu, Ishikaa Lunawat, Isabel Sieh, Sean Kirmani, Sergey Levine, Jiajun Wu, Chelsea Finn, Hao Su, Quan Vuong, and Ted Xiao. Evaluating real-world robot manipulation policies in simulation. *Conference on Robot Learning (CoRL)*, 2024.

[23] Yi Li, Yuquan Deng, Jesse Zhang, Joel Jang, Marius Memmel, Caelan Reed Garrett, Fabio Ramos, Dieter Fox, Anqi Li, Abhishek Gupta, and Ankit Goyal. Hamster: Hierarchical action models for open-world robot manipulation. *International Conference on Learning Representations (ICLR)*, 2025.

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

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

[26] Rashmeet Kaur Nayyar, Pulkit Verma, and Siddharth Sri-

vastava. Differential assessment of black-box ai agents. *AAAI Conference on Artificial Intelligence*, 2022.
[27] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Lawrence Yunliang Chen, Pannag Sanketi, Quan Vuong, Ted Xiao, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. *Robotics: Science and Systems (RSS)*, 2024.
[28] Amit Parekh, Nikolas Vitsakis, Alessandro Suglia, and Ioannis Konstas. Investigating the Role of Instruction Variety and Task Difficulty in Robotic Manipulation Tasks. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2024.
[29] Biswajit Paria, Willie Neiswanger, Ramina Ghods, Jeff Schneider, and Barnabás Póczos. Cost-aware bayesian optimization via information directed sampling. In *Adaptive Experimental Design and Active Learning in the Real World Workshop at ICML*, 2020.
[30] Wilbert Pumacay, Ishika Singh, Jiafei Duan, Ranjay Krishna, Jesse Thomason, and Dieter Fox. THE COLOSSEUM: A Benchmark for Evaluating Generalization for Robotic Manipulation. *Robotics: Science and Systems (RSS)*, 2024.
[31] Tom Rainforth, Adam Foster, Desi R Ivanova, and Freddie Bickford Smith. Modern bayesian experimental design. *Statistical Science*, 39(1):100–114, 2024.
[32] Christoph Sawade, Niels Landwehr, Steffen Bickel, and Tobias Scheffer. Active risk estimation. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 951–958, 2010.
[33] Dhruv Shah, Ajay Sridhar, Nitish Dashora, Kyle Stachowicz, Kevin Black, Noriaki Hirose, and Sergey Levine. Vint: A foundation model for visual navigation. *Conference on Robot Learning (CoRL)*, 2022.
[34] Dhruv Shah, Błażej Osiński, Sergey Levine, et al. Lm-nav: Robotic navigation with large pre-trained models of language, vision, and action. *Conference on Robot Learning (CoRL)*, 2023.
[35] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando De Freitas. Taking the human out of the loop: A review of bayesian optimization. *Proceedings of the IEEE*, 104(1):148–175, 2015.
[36] David Snyder, Asher James Hancock, Apurva Badithela, Emma Dixon, Patrick Miller, Rares Andrei Ambrus, Anirudha Majumdar, Masha Itkina, and Haruki Nishimura. Is your imitation learning policy better than mine? policy comparison with near-optimal stopping. *arXiv preprint arXiv:2503.10966*, 2025.
[37] Christopher Tosh, Mauricio Tec, and Wesley Tansey. Targeted active learning for probabilistic models. *arXiv preprint arXiv:2210.12122*, 2022.
[38] Pulkit Verma, Shashank Rao Marpally, and Siddharth Srivastava. Discovering user-interpretable capabilities of black-box planning agents. *International Conference on Principles of Knowledge Representation and Reasoning (KR)*, 2021.
[39] Pulkit Verma, Rushang Karia, and Siddharth Srivastava. Autonomous capability assessment of sequential decision-making systems in stochastic settings. *Conference on Neural Information Processing Systems (NeurIPS)*, 2023.
[40] Joseph A Vincent, Haruki Nishimura, Masha Itkina, Paarth Shah, Mac Schwager, and Thomas Kollar. How Generalizable Is My Behavior Cloning Policy? A Statistical Approach to Trustworthy Performance Evaluation. *IEEE Robotics and Automation Letters (RA-L)*, 2024.
[41] Annie Xie, Lisa Lee, Ted Xiao, and Chelsea Finn. Decomposing the generalization gap in imitation learning for visual robotic manipulation. *International Conference on Robotics and Automation (ICRA)*, 2024.
[42] Emine Yilmaz, Peter Hayes, Raza Habib, Jordan Burgess, and David Barber. Sample efficient model evaluation. *arXiv preprint arXiv:2109.12043*, 2021.
[43] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *Conference on Robot Learning (CoRL)*, 2020.

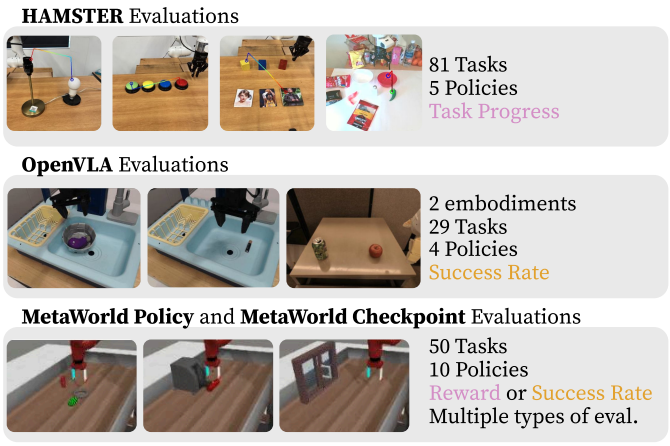

Fig. 5. **Offline Datasets used for Experiments.** We consider 4 settings of offline evaluations, as denoted above.

APPENDIX A
LIMITATIONS

Cost-effective performance distribution estimation. While our approach better estimates the mean performance of policy-task pairs (see Section V-B), 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 6. 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 [36], 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.

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 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.

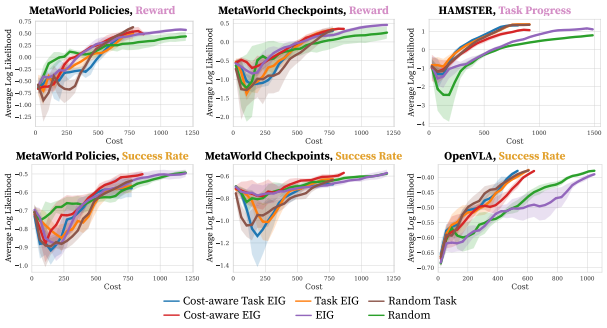

Fig. 6. **Average Log Likelihood Over Cost.** We show the average log likelihood of all the outcomes in our offline dataset against the cost of evaluation for MetaWorld Policies, MetaWorld Checkpoints, HAMSTER, and OpenVLA over continuous and binary performance distributions. Each set of experiments 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.

APPENDIX B
SURROGATE MODEL

A. 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.

B. 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.

APPENDIX C
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 $p(\pi_i, T_j)$, exploration rate $\epsilon = 0.1$

Randomly sample a single task T_j and evaluate every policy 3 times. Add outcomes x_{ij}^k to $\mathcal{D}$
Set $T_{\text{current}} = T_j$
Increment $C_{\text{total}} = C_{\text{eval}} + c_{\text{eval}} \cdot |\mathcal{P}| \cdot 3$
Train the surrogate model $p(\cdot)$ on $\mathcal{D}$ for k epochs
for each query step **do**
    Use MC dropout to sample 10 predicted distributions from the surrogate model for every policy-task pair
    Use sampled distributions to compute scores $s_{ij} = a(\pi_i, T_j, T_{\text{current}})$ according to Eq. 2
    With probability ϵ, select a random (π_i, T_j)
    Otherwise, select $(\pi_i, T_j) = \arg\max_{(\pi_i, T_j)} s_{ij}$
    Conduct 3 evaluations and observe $x_{ij}^1, x_{ij}^2, x_{ij}^3 \sim P_{ij}$
    Add these outcomes to $\mathcal{D}$
    Train $f(\cdot)$ on $\mathcal{D}$ for k epochs
    Increment $C_{\text{total}} = C_{\text{total}} + c_{\text{eval}} \cdot 3$
    if $T_j \neq T_{\text{current}}$ **then** ▷ Task switching cost applies
        Increment $C_{\text{total}} = C_{\text{total}} + c_{\text{switch}}(T_{\text{current}}, T_j)$
        Update $T_{\text{current}} = T_j$
    end if
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 [24, 20] 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.

APPENDIX D
OFFLINE DATASET DETAILS

A. HAMSTER

We use evaluations from HAMSTER [23], 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.

B. OpenVLA

From OpenVLA [18], 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 changing cost to 3, as this change is relatively large.

C. MetaWorld Policies/Checkpoints

MetaWorld [43] 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.

APPENDIX E
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.

a) Task Representations.: We define 4 different task reprsentations.

- **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 IV-A).
- **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.

b) 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.

APPENDIX F
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 IV-A. 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.