SEAL: Safe and Efficient Abstraction Learning for Robotic Planning

Akhil R Kurup
Indian Institute of Science, Bengaluru

Ravi Prakash
Indian Institute of Science, Bengaluru

Abstract: Advances in Large Language Models have enabled the decomposition of complex tasks into subtasks for easier planning. A key challenge remains: how can we ensure a robot's task execution respects implicit task constraints and remains robust to real-world perturbations? We propose an active learning framework to efficiently learn a symbolic mode classifier that maps robot states to discrete modes, thereby establishing a grounded world model. By using active learning to iteratively improve our classification by querying experts at uncertain mode boundaries, we can match the performance of previous approaches but without any potentially unsafe perturbed trajectories or oracles, thereby enabling policies to become robust to task-level disturbances. Our findings underscore the potential for data-efficient, safe, and reliable abstraction learning to support long-term robot autonomy. https://akhilrkurup.github.io/

Keywords: Interactive Imitation Learning, Task Planning, Semantic Modes

1 Introduction

There has been growing interest in using Foundational models for robot manipulation, in particular for high-level task planning by decomposing complex, long-horizon tasks into sequential subtasks [[1],[2],[3],[4]], while using low-level policies for subtasks. While subtask policies learned using popular Learning From Demonstration methods, such as DMP [5], DS[6] are robust to perturbations at the motion scale, they may fail at the task scale as the policy is blind to the high-level modes and valid mode sequences necessary for successful completion of the task.

Linear Temporal Imitation [7] used Linear Temporal Logic formulas to make imitators robust to motion level and task level perturbations, and can guarantee task success. However, they require a logic formula and perfect sensors to detect mode transitions, which requires domain expertise in building the formulas and in engineering the sensors. GLiDE [8] built upon this concept by learning instead of engineering sensor models. To learn the mode classifier, they perturb an initial set of unsegmented human demonstrations and learn a classifier with only binary success/failure labels. While this approach of learning from sparse labels is powerful, there are several limitations to it in real-world applications. Perturbations may not be feasible outside a simulator, as failing trajectories may be dangerous and hard to reset autonomously . This approach also requires an oracle, which can identify invalid mode transitions in perturbed trajectories during the training phase which one has to manually engineer.

The main limitations of such an approach are a consequence of learning from sparse labels rather than dense mode labels. Our active learning paradigm - SEAL, uses dense labels from a few segmented expert demonstrations and an intelligent querying strategy to incrementally improve the classifier's performance. By focusing on uncertain regions near the boundaries between different modes, we can match the performance of GliDE with minimal human effort and no dangerous robot

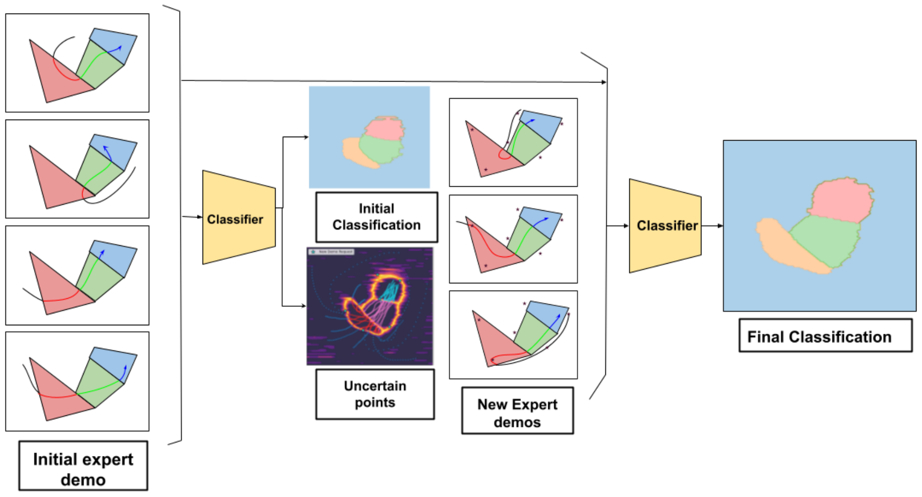

Figure 1: **Framework:** We start with a set of expert demonstrations of a trajectory segmented into the different semantic modes each part of the trajectory represents. A classifier is trained on these using the per-point labels. Using the uncertainty over predictions of the classifier, we select uncertain points and collect a few more expert demonstrations passing through those points. Finally, we train a classifier on the combined set of trajectories, which can then be used for downstream tasks.

rollouts. This method directly addresses a key question for the LEAP workshop: "When, where, and from what data should abstractions be learned ?"

2 Method

The goal of our method is to efficiently learn a mode classifier $f : \mathcal{X} \to \mathcal{M}$ that maps a robot state $x \in \mathcal{X}$ to a discrete symbolic mode $m \in \mathcal{M}$. We assume a high-level task plan, consisting of a sequence of subtasks and their valid transitions.

2.1 Initial expert demonstrations

An expert first provides a few demonstrations of the complete task. A trajectory is represented as a sequence of states (robot state and observations) $\tau = \{x_t\}_{t=1}^{T}$. The expert provides dense, per-point labels, segmenting the trajectory into modes. This gives us an initial dataset $\mathcal{D}_0 = \{(x_t, m_t)\}_{t=1}^{N}$. Segmentation is easily achieved by asking the expert for a signal after each subtask, which we argue is not much more effort from the expert. We assume all human demonstrations follow legal mode transitions and successfully complete the task.

2.2 Active Learning for Classifier Training

We use a Gaussian Process Classifier (GPC) for its ability to provide explicit uncertainty estimates for its predictions, which is essential for our active learning framework. The GPC defines a probability distribution over functions. Given our dataset $\mathcal{D}_k = \{(x_i, m_i)\}_{i=1}^{N}$ at iteration k, the GPC provides a predictive distribution for a new test point x^*, which we approximate as a Gaussian distribution.

The posterior predictive probability for a Gaussian Process Classifier (GPC) is given by the integral:

$$P(m|x^*, \mathcal{D}_k) = \int p(m|f^*)p(f^*|\mathcal{D}_k)df^* = \int \sigma(f^*)p(f^*|\mathbf{f}, \mathcal{D}_k)df^*$$

where:

- $P(m|x^*, \mathcal{D}_k)$ is the probability of a new data point at x^* belonging to class m, given the training data $\mathcal{D}_k$.

- f^* is the latent function value at the test point x^*.
- $\sigma(f^*)$ is the sigmoid (or probit) link function, which maps the continuous latent function output to a probability.
- $p(f^*|\mathbf{f}, \mathcal{D}_k)$ is the posterior distribution over the latent function values. This distribution is non-Gaussian due to the non-linear link function.

This integral is computationally intractable, so GPC implementations rely on approximation methods. We use Variational Inference and the GPyTorch [9] library for GPU acceleration.

The GPC allows us to estimate the full predictive probability vector

$$\mathbf{p}^* = [P(m_1|x^*), P(m_2|x^*), \ldots, P(m_{|\mathcal{M}|}|x^*)]$$

Our active learning loop proceeds as follows:

1. Train Initial Classifier: Train a GPC on the initial dataset $\mathcal{D}_0$ and get predictions on a grid around the demonstrations.
2. Identify Uncertain Points: We define the uncertainty of a prediction at point x^* using the margin uncertainty metric. This is the difference between the probabilities of the two most probable modes:

   $$\text{Uncertainty}(x^*) = \mathbf{p}^*_{\text{top1}} - \mathbf{p}^*_{\text{top2}}$$

   where $\mathbf{p}^*_{\text{top1}}$ and $\mathbf{p}^*_{\text{top2}}$ are the highest and second-highest probabilities in $\mathbf{p}^*$. A smaller margin indicates higher uncertainty, as the classifier is less confident about which of the top two modes is correct. We select a set of query points $\mathcal{Q}$ with the highest uncertainty. To ensure broad coverage, we choose query points that are maximally distant from each other.
3. Query Expert: We request additional segmented expert demonstrations $\mathcal{D}_{\text{new}}$ that pass through these selected uncertain points. This directs the expert's effort to the most informative areas of the state space, particularly where mode boundaries are ambiguous.
4. Retrain Classifier: The new data is added to the total dataset, $\mathcal{D}_{k+1} = \mathcal{D}_k \cup \mathcal{D}_{\text{new}}$, and the classifier is retrained. This process can be repeated until a desired level of certainty is achieved.

3 Experiments

We evaluate our method on two settings: (1) A simulated 2D navigation task and (2) an inspection task on a real robot. Each of these settings is designed to highlight the performance and application of a mode classifier in imitation tasks.

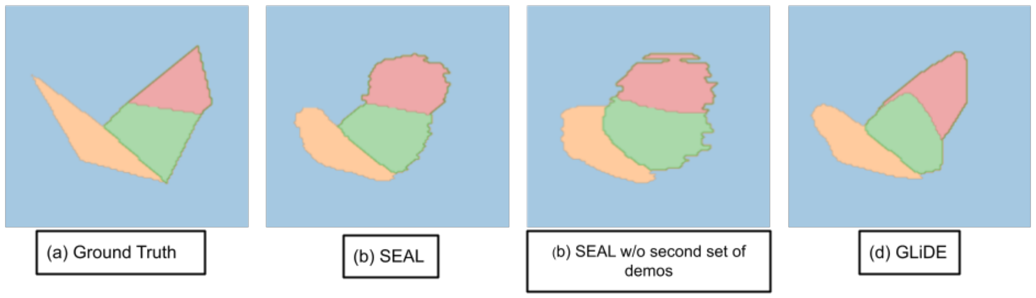

Figure 2: Comparison of the classification performance of the different methods.(b) uses 6+6 demos, (c) uses only the original 6 demos and (d) uses 6 demos + 1000 perturbations.

3.1 2D Navigation

This experiment, replicated from GLIDE, consists of connected polygons representing different modes. The objective is to move through each polygon in the correct sequence and reach the goal polygon. The agent starts in the outer region representing the start mode.

3.1.1 Results

For a fair comparison against the baseline, we use the same 6 demonstrations for both. For SEAL 6, additional demonstrations are provided, passing through the selected uncertain points. Figure 2 shows results for SEAL before and after additional demonstrations and GLiDE with a differing number of perturbed trajectories. As shown, our method (b) is close to the ground truth classification (a). The improvement in (b) over (c), which is without the second set of demonstrations, shows the utility of our active learning approach. Figure 4 shows the Mean Intersection over Union of different methods. Our performance matches that of GLiDE with 1000 perturbed trajectories. Also evident is how the performance of the other method drops when the number of perturbed trajectories used are reduced.

Figure 3 shows the utility of such a classifier to make multi-step tasks robust to disturbances. In this experiment a simple potential flow controller is used but this can be extended to any policy to detect mode changes and re-plan trajectories after perturbations.

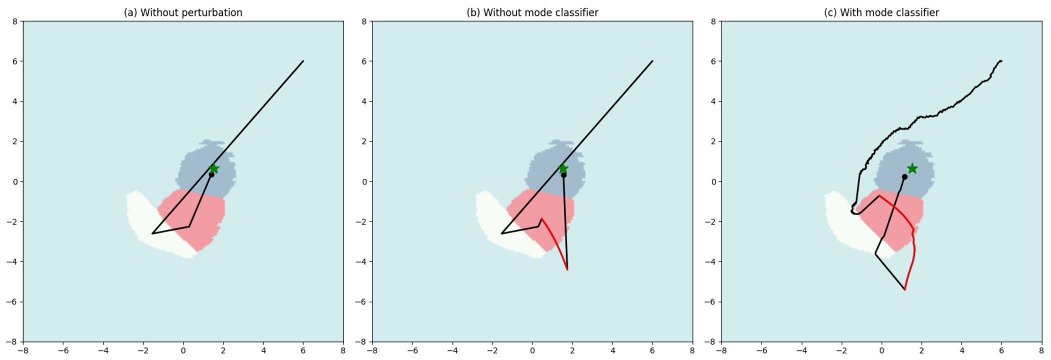

Figure 3: Rollout of a per-mode policy. The trajectories start at the top right and the red part of the trajectories denotes external perturbations. While the policy succeeds in (a) if there are no perturbations, it fails to obey valid mode transitions in (b) after perturbations are applied. In (c), using the mode classifier, we are able to enforce replanning through a valid sequence of modes. Additionally, we can use the mode boundaries as obstacles for planning.

3.1.2 Discussion

Note that SEAL never uses the ground truth information during training, unlike GLiDE, which uses an oracle in the form of ground truth information to verify the success of perturbed trajectories. We do not claim that our method surpasses GLiDE, which solves the harder task of learning from sparse labels, but only argue that we can match it with only a little more human effort and without the need for an oracle. The need for many perturbations makes GLiDE hard to deploy on real robots without a simulator or an expert demonstrating failure examples whereas our method does not require such examples.

While this test is useful to visualise the performance of the classifier, real-world tasks are unlikely to be as simple as 2D polygons, and performance depends on the quality of expert demonstrations. However, the general trends shown should hold in real world deployments.

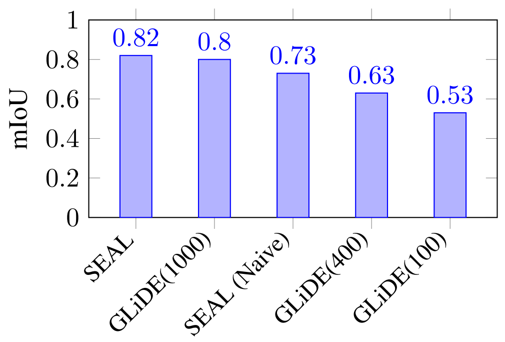

Figure 4: Mean IoU across modes for different methods. SEAL uses 6 initial+6 queried demonstrations while SEAL (Naive) uses only the 6 original demonstrations without re querying. The numbers in brackets denote the number of additional perturbed trajectories used by GLiDE.

3.2 3D Inspection

In this simulated inspection scenario the manipulator has to move an object through three zones to successfully inspect the object with the middle zone representing an ”inspection lane”. Figure 5 shows details of the task. Note that the robot ”state” in this case is the 3D position of the end-effector. Expert demos are collected through kinesthetic teaching, and an initial classifier is learned. Then additional demos are requested again at uncertain points (marked in black in (c)). The predictions of the learned classifier are seen to match the ground truth well only with 6+3 demonstrations. Note that the red lines in (a) are only to guide expert demos and this ground truth is not used during training.

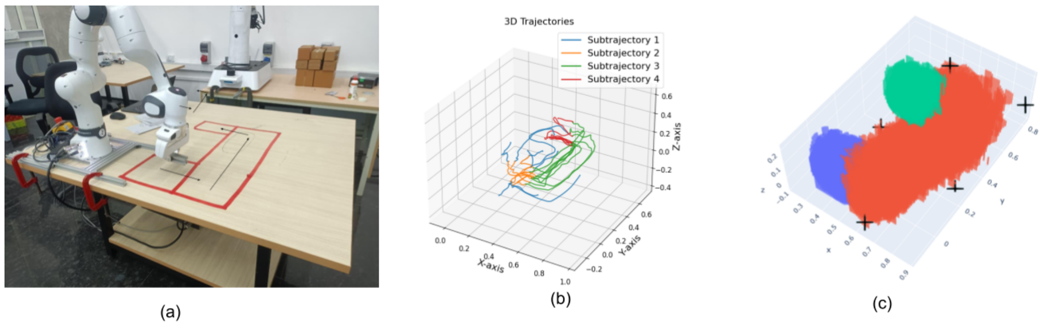

Figure 5: Setup of the inspection task. (a) shows the real world setup. (b) shows the trajectories collected after re-querying. (c) shows the predictions of the learned mode classifier.

4 Conclusions and Limitations

We present the SEAL framework and show how a mode classifier can be learned efficiently from a few human demonstrations and help multi-step LfD policies become robust to task-level perturbations. Further, we demonstrate the advantages of our work over previous approaches in this domain.

While our approach is promising, it has limitations that warrant future work.

High-Dimensional State Spaces: Our current method is limited to low-dimensional states. To extend it to high-dimensional observations such as RGB-D images, a robust feature extractor is required. Furthermore, there is the question of how to query expert demonstrations for such abstract, non-physical states.

Generalisation: The learned abstractions are specific to a single task. A key question for future research is whether our active learning approach can be used to learn composable abstractions that generalise across different but related tasks.

Additionally, recent works have shown the ability of Vision Language Models to automatically segment human demonstration video into subtasks [10], which can relieve the burden of providing segmentation from the expert and opens up the possibility of learning from pre existing demonstrations.

References

[1] M. Dalal, M. Liu, W. Talbott, C. Chen, D. Pathak, J. Zhang, and R. Salakhutdinov. Local policies enable zero-shot long-horizon manipulation. *International Conference of Robotics and Automation*, 2025.

[2] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, K. Gopalakrishnan, K. Hausman, A. Herzog, D. Ho, J. Hsu, J. Ibarz, B. Ichter, A. Irpan, E. Jang, R. M. J. Ruano, K. Jeffrey, S. Jesmonth, N. J. Joshi, R. C. Julian, D. Kalashnikov, Y. Kuang, K.-H. Lee, S. Levine, Y. Lu, L. Luu, C. Parada, P. Pastor, J. Quiambao, K. Rao, J. Rettinghouse, D. M. Reyes, P. Sermanet, N. Sievers, C. Tan, A. Toshev, V. Vanhoucke, F. Xia, T. Xiao, P. Xu, S. Xu, and M. Yan. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022.

[3] W. Huang, F. Xia, T. Xiao, H. Chan, J. Liang, P. Florence, A. Zeng, J. Tompson, I. Mordatch, Y. Chebotar, P. Sermanet, T. Jackson, N. Brown, L. Luu, S. Levine, K. Hausman, and brian ichter. Inner monologue: Embodied reasoning through planning with language models. In *6th Annual Conference on Robot Learning*, 2022.

[4] J. Wu, R. Antonova, A. Kan, M. Lepert, A. Zeng, S. Song, J. Bohg, S. Rusinkiewicz, and T. Funkhouser. Tidybot: Personalized robot assistance with large language models. *Autonomous Robots*, 2023.

[5] A. Ijspeert, J. Nakanishi, P. Pastor, H. Hoffmann, and S. Schaal. Dynamical movement primitives: Learning attractor models for motor behaviors. *Neural Computation*, (25):328–373, 2013. clmc.

[6] A. Billard, S. S. Mirrazavi Salehian, and N. Figueroa. *Learning for Adaptive and Reactive Robot Control: A Dynamical Systems Approach*. Intelligent Robotics and Autonomous Agents Series,. MIT Press, Cambridge, USA, 2022.

[7] Y. Wang, N. Figueroa, S. Li, A. J. Shah, and J. A. Shah. Temporal logic imitation: Learning plan-satisficing motion policies from demonstrations. In *Conference on Robot Learning*, 2022.

[8] Y. Wang, T.-H. Wang, J. Mao, M. Hagenow, and J. Shah. Grounding language plans in demonstrations through counterfactual perturbations. In *The Twelfth International Conference on Learning Representations*, 2024.

[9] J. R. Gardner, G. Pleiss, D. Bindel, K. Q. Weinberger, and A. G. Wilson. Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration. In *Advances in Neural Information Processing Systems*, 2018.

[10] B. Wang, J. Zhang, S. Dong, I. Fang, and C. Feng. Vlm see, robot do: Human demo video to robot action plan via vision language model, 2024.