Learning Modularity for Generalizable Robotic Behaviors

Corban Rivera [1] **Chace Ashcraft** [1] **Edward W. Staley** [1] **Katie M. Popek** [1] **Kapil D. Katyal** [1] **Bart L. Paulhamus** [1]

Abstract

Modularity in deep neural networks has provided scaling efficiencies leading to state of the art performance across multiple domains. A critical challenge for these networks is how to build and maintain a library of generalizable behavior modules. In this work, we propose a novel framework for building and maintaining a library of behavior primitives called Primitive Imitation for Control ($PICO$). Unlabeled demonstrations are automatically decomposed into existing or missing sub-behaviors which allows the framework to identify novel behaviors while not duplicating existing behaviors. We compared our results to several related approaches across two environments and achieve both better label accuracy and reconstruction accuracy as measured by action prediction mean squared error.

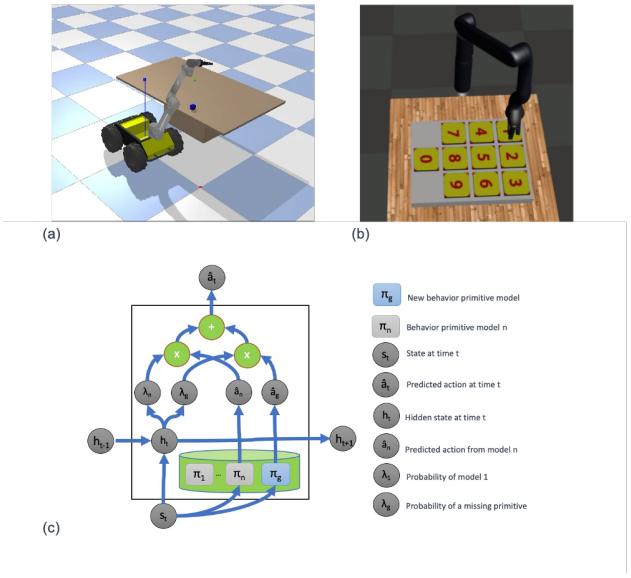


Figure 1. We explore the challenge of building and maintaining a behavior repository given unlabeled demonstrations. Our experiments include two manipulation task domains including (a) reach and grasp and (b) dial pad. (c) Illustration of our hierarchical recurrent deep network architecture for task decomposition, novel behavior primitive discovery, and behavior blending. Our unique approach directly minimizes reconstruction error while building and maintaining a library of behavior primitives.

1. INTRODUCTION

Neural networks have demonstrated extraordinary ability to control systems with high degrees of freedom. An important challenge is how to control such high-degree of freedom systems with only a few control inputs. One approach to address the scaling challenge is through modularity and hierarchical control mechanisms (Zhao et al., 2017; Akinola et al., 2017; Shazeer et al., 2017). These approaches use the limited number of inputs to select a primitive control policy, from a library of primitive behaviors, and potentially a target. Complex tasks are performed by chaining primitive behaviors.

We investigated how we might learn and maintain a primitive library from unlabeled demonstrations and, assuming the behavior primitive library exists, how would one know when to use, adapt, or create a new primitive behavior. We propose that the behavior library should be actively maintained to minimize redundancy and maximize the ability to reconstruct complex tasks through chains of the primitive behaviors. In this work, we explore techniques to directly optimize for these criteria.

2. PRELIMINARIES

Learning from demonstration (LfD) and imitation learning allow agents to execute a task by observing the task being performed (Hussein et al., 2017). In the robotics domain, a goal of imitation learning is to produce a mapping, π, from states to actions, known as a control *policy* (Argall et al., 2009; Schaal & Atkeson, 2010), that

*Equal contribution [1]Intelligent Systems Center
Johns Hopkins University Applied Physics Lab
11100 Johns Hopkins Rd. Laurel, MD 20723. Correspondence to: Corban Rivera <corban.rivera@jhuapl.edu>.

DyNN workshop at the 39^{th} International Conference on Machine Learning, Baltimore, Maryland, USA, 2022.

has the maximum likelihood of producing the demonstration dataset $\mathcal{D} = \{\rho_1, \rho_2, \ldots, \rho_n\}$, where each $\rho = ((s_1, a_1), (s_2, a_2), \ldots, (s_T, a_T)$ is a demonstration trajectory of of state, action pairs. The demonstrations can be created by another control policy (Rusu et al., 2015), by a human expert (Konidaris et al., 2012), or in a simulated environment (Shiarlis et al., 2018; Kipf et al., 2019). Let π_θ parameterized by θ. The goal is then to optimize Equation 1 by varying θ and a library of behavior primitives $\mathcal{B} = (\pi_1, \pi_2, \ldots, \pi_K)$

$$\max \mathbb{E}_\rho[\sum_{t=1}^{T} \log \pi_\theta(a_t|s_t)] \tag{1}$$

3. Approach

In this work, we introduce Primitive Imitation for Control ($PICO$). Our contribution is a unique solution that simultaneously learns subtask decomposition from unlabeled task demonstrations, trains missing behavior primitives, and learns a hierarchical control mechanism that allows blending of primitive behaviors to create even greater behavioral diversity. Our approach directly optimizes the contents of the primitive library to maximize the ability to reconstruct unlabeled task demonstrations from sequences of primitive behaviors. Our approach takes inspiration the work on mixtures-of-experts (Shazeer et al., 2017; Jacobs et al., 1991).

$PICO$ aims to reconstruct the given trajectories as well as possible using the existing sub-task policy library. As shown in Equation 2, $PICO$ minimizes the sum of squared error between the observed action and the predicted action for all actions over all timepoints T and all trajectories $\rho \in \mathcal{D}$ which we refer to as *reconstruction error*. Let (s_{ρ_t}, a_{ρ_t}) be the state-action tuple corresponding to ρ_t timepoint t in trajectory ρ. The action prediction, equation 3, is the product of the probability $p(\pi|s_{\rho_t})$ of a sub-task policy π conditioned on the state s_{ρ_t} and the action predicted by policy $\pi(s_{\rho_t})$ for the state s_{ρ_t}. Substituting equation 3 into 2 results in Equation 4 which is the optimization problem for $PICO$.

$$\min \sum_{\rho \in \mathcal{D}} \sum_{t=0}^{T} (a_{\rho_t} - \hat{a}_{\rho_t})^2 \tag{2}$$

$$\hat{a}_{\rho_t} = \sum_{\pi \in \mathcal{B}} p(\pi|s_{\rho_t})\pi(s_{\rho_t}) \tag{3}$$

$$\min \sum_{\rho \in \mathcal{D}} \sum_{t=0}^{T} (a_{\rho_t} - \sum_{\pi \in \mathcal{B}} p(\pi|s_{\rho_t})\pi(s_{\rho_t}))^2 \tag{4}$$

3.1. Neural Network Architecture

Estimates of both $p(\pi|s_{\rho_t})$ and $\pi(s_{\rho_t})$ are given by a recurrent neural network architecture. Figure 2 gives an overview of the recurrent and hierarchical network architecture. We solve for the objective in Equation 4 directly by back propagation through a recurrent neural network with equation 2 as the loss function. The model architecture is composed of two branches that are recombined to compute the action prediction at each timepoint.

To more easily compare with other approaches that do not blend sub-task policies, we estimate the maximum likelihood sub-task policy label at each timepoint. We refer to sub-task policies as behavior primitives. The behavior primitive label prediction is given by the maximum likelihood estimate of π shown in Equation 5 for time t in trajectory ρ.

$$\underset{\pi \in \mathcal{B}}{\operatorname{argmax}}\, p(\pi|\rho_t) \tag{5}$$

Figure 1 (c) illustrates how we compute the predicted action $\hat{a}_t$ at time t. In the figure, the probability of π given state s_{ρ_t} is $\lambda_\pi = p(\pi|s_{\rho_t})$ for $\pi \in \mathcal{B}$. The latent representation h_t at the current timepoint t is a function of both the value of the latent representation of the previous state h_{t-1} and the current state s_t

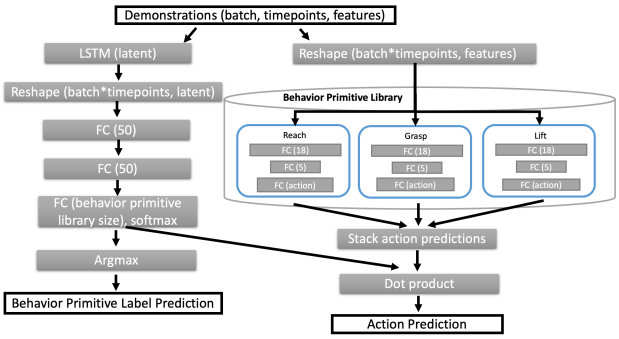


Figure 2. Neural network architecture for $PICO$. Given a set of input trajectories and a behavior primitive library, the core architecture follows two branches, the left most branch estimates a distribution over the behavior primitives. The right hand branch estimates the action prediction from each primitive behavior submodel. We compute the predicted action as a linear combination between the behavior primitive distribution and the set of predicted actions from all behavior primitives.

Figure 2 details the architecture used for $PICO$ based on the Husky+UR5 dataset example. Unless otherwise specified, the fully connected (FC) layers have ReLU activations, except for the output layers from behavior primitive models. The last layer of behavior primitive models have linear activations to support diverse action predictions. While not

shown in Figure 2, the network architecture also returns the predicted latent embedding and behavior primitive distribution for additional visualization and analysis.

3.2. Discovering and Training New Behavior Primitives

An important aspect of our approach is the ability to discover and create new behavior primitives from a set of trajectories and a partial behavior primitive library. $PICO$ detects and trains new behavior primitive models simultaneously. As shown in figure 1(c), $PICO$ supports building new behavior primitive models by adding additional randomly initialized behavior models to the library prior to training. For our experiments, we assume that we know the correct number of missing primitives.

We define a *gap* in a trajectory as region within a demonstration where actions are not predicted with high probability using the existing behavior primitive models. A gap in a trajectory implies that the current library of behavior primitives is insufficient to describe a set of state-action tuples ρ in some part of the given trajectory. This also implies that the probability $p(\pi|\rho_t)$ that the data ρ_t for time point t was generated by the current library of behavior primitive models is low for all $\pi \in \mathcal{B}$. These low probabilities increase the likelihood that an additional randomly initialized behavior primitive policy π_{new} might have a higher probability $p(\pi_{new}|\rho_t) > p(\pi|\rho_t)$ for $\pi \in \mathcal{B}$. The data ρ_t is then used to train π_{new}. For nearby data in the same gap region ρ_{t+1}, it is now more likely that $p(\pi_{new}|\rho_{t+1}) > p(\pi|\rho_{t+1})$ for $\pi \in \mathcal{B}$. This mechanism allows π_{new} to develop in to a new behavior primitive that is not well covered by existing primitives.

4. EXPERIMENTS AND DISCUSSION

With our experiments, we aimed to (i) evaluate the ability of $PICO$ to identify parts of demonstrations that are not represented by existing behavior primitives and rebuild the missing behavior primitive, and (ii) demonstrate generalization of $PICO$ by comparison to several recent alternative approaches across two manipulation task domains.

4.1. Robotic Manipulation Task Domains

Husky-UR5 Reach and Grasp – A robotic reach and grasp environment. The position of the target block randomized. The dataset consists of 100 demonstrations of a Clearpath Husky robot with a UR5 manipulator performing a variety of reach, grasp, and lift tasks, see Figure 1(a). The number of time steps in the demonstrations varied from 1000 to 1800, but each used all three primitives: reach, grasp, and lift.

Dial Domain The dial domain (Shiarlis et al., 2018) is composed of demonstrations from a Jaco manipulator pressing 4 keys in sequence (e.g. 3,5,4,7) illustrated in Figure 1(b). The positions of the keys are randomly shuffled for each demonstration, but the position of each key is given in the state vector. In this environment, pressing an individual digit is the behavior primitive. Dialpad demonstrations were generated using default parameters (Shiarlis et al., 2018).

4.2. Training Details

As described in Equation 2, the loss used to train the model is mean squared error between the predicted and observed actions over all timepoints and all demonstrations. The approach illustated in Figure 2 is trained end to end using the the DART (Laskey et al., 2017) technique for imitation learning over 40 epochs with the Adam optimizer.

4.3. Evaluation Metrics

Two metrics were computed to estimate performance. First, we evaluated mean squared error (MSE) as shown in Equation 2 between the predicted and given action. Second, we computed the behavior primitive label accuracy which was a comparison between the predicted and given behavior primitive label. Label accuracy was computed as the number of matching labels divided by the total number of comparisons. Both metrics were computed over all timepoints and over all demonstrations in the test set.

4.4. Comparison Baselines

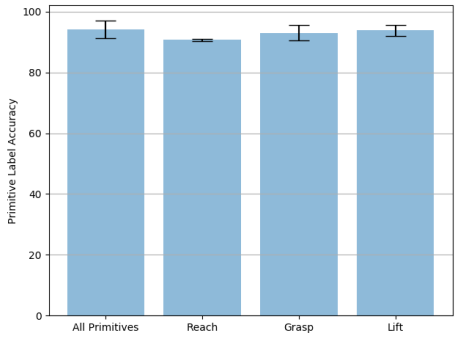
We compared $PICO$ to TACO (Shiarlis et al., 2018) and the extended version of CTC (Graves et al., 2006) using an open source implementations [1] and default parameters. Both CTC and TACO baseline approaches were tested with MLP and RNN architectures. Due to space constraints, details of CTC and TACO are described in more detail in Appendix A.4.

4.5. Behavior Primitive Discovery

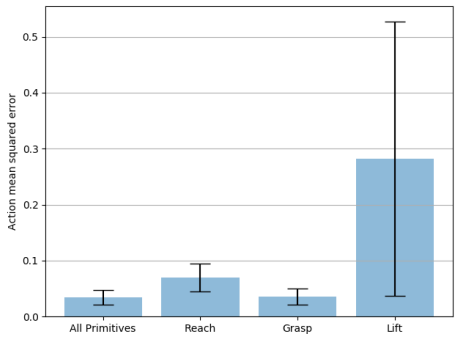
We evaluated the ability of $PICO$ to recognize and build a missing behavior primitive model focusing on the Husky-UR5 Reach and Grasp task. We structured the experiment as a leave-one-behavior-out experiment where one of the three primitives (i.e. reach, grasp, lift) was replaced with a randomly-initialized behavior primitive. An 80/20 split between training and validation sets. Five trials were run with the training and validation sets randomly chosen. The label accuracy and action prediction MSE are shown in 3. The leftmost bar shows the results with all primitives pre-trained with behavior cloning. The remaining bars show the accuracy when reach, grasp and lift, respectively, were replaced with the gap primitive. Note, the gap primitive was updated throughout the training with back-propagation

[1] https://github.com/KyriacosShiarli/taco

such that the final primitive ideally would perform as well as the original pre-trained, behavior-cloned version; this comparison is shown with the action prediction MSE. The error bars show the standard deviation across the five trials. While the label accuracy across all three replaced primitives is approximately the same, the action prediction for the lift primitive is significantly worse. We believe this is due to the larger variance in lift trajectories. Unlike the reach and grasp which have restrictions placed on their final target position (it needs to be near the block), the final position of lift is randomly placed above the block's starting position. As shown in Table 1, over all of the test trajectories, the average label classification accuracy was 96% for our approach.



(a) behavior label accuracy



(b) action prediction MSE

Figure 3. Accuracy of $PICO$ to correctly identify a primitive's label on the validation set (twenty randomly selected trajectories). (a) The leftmost bar shows performance when all primitives are in the library, successive bars denote accuracy when the *reach*, *grasp*, and *lift* primitives are dropped out and learned from a randomly generated "gap" primitive. Error bars represent the standard deviation across five validation trials. (b) Mean squared error between the ground truth action and the learned model's estimate averaged across twenty randomly selected test trajectories five times.

4.6. Multi-domain comparison of label accuracy

The goal of this comparison was to evaluate the label prediction accuracy given an existing library of behavior primitives. To isolate the label predictions of each approach, the behavior primitive library was pretrained on the training dataset including 1200 demonstrations and frozen. Label classification and action prediction accuracy was then evaluated on the test set including 280 demonstrations.

Table 1. Method comparisons using the Husky UR5 Reach and Grasp dataset.

Husky UR5	Label Accuracy	MSE Action Prediction
$PICO$	**96%**	**0.053**
TACO (MLP)	74%	3.59
TACO (RNN)	73%	3.75
CTC (MLP)	25%	4.20
CTC (RNN)	33%	2.68

Table 2. Method comparisons using the Jaco Pinpad dataset. *TACO (RNN) resulted in NaN loss after repeated attempts.

Jaco Pinpad	Label Accuracy	MSE Action Prediction
$PICO$	**65%**	**0.0061**
TACO (MLP)	47%	0.55
TACO (RNN)	*	*
CTC (MLP)	31%	0.57
CTC (RNN)	29%	0.58

The average results of 5 runs are shown for TACO and CTC. We evaluate each approach using the same label accuracy and action prediction metrics. The summary of results are shown in Table 2. We found that our approach achieves the highest label accuracy at 65%. The overall label accuracy of $PICO$ on the dial dataset is lower than the Husky+UR5 dataset. Additional analysis revealed that many of the mislabeling occurred at the beginning of a new key press where context about where the Jaco is moving next is weakest. The dataset is also more challenging than than the Husky dataset because the number of unique behavior primitives has increased from 3 to 10.

5. CONCLUSION

In this paper, we describe $PICO$, an approach to learn behavior primitives from unlabeled demonstrations and a partial set of behavior primitives. We compared our results to several related approaches across two environments and achieve both better label accuracy and reconstruction accuracy as measured by action prediction mean squared error. While we have demonstrated success in these tasks, there are limitations to our approach. The number additional primitives to add to the library must be decided prior to training. Future work may explore relaxing these assumptions.

References

www.clearpathrobotics.com/husky-unmanned-ground-vehicle-robot/. Accessed: 2019-09-10.

www.universal-robots.com. Accessed: 2019-09-10.

Akinola, I., Chen, B., Koss, J., Patankar, A., Varley, J., and Allen, P. Task level hierarchical system for bci-enabled shared autonomy. In *2017 IEEE-RAS 17th International Conference on Humanoid Robotics (Humanoids)*, pp. 219–225, Nov 2017. doi: 10.1109/HUMANOIDS.2017.8246878.

Andreas, J., Klein, D., and Levine, S. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 166–175. JMLR. org, 2017.

Andrychowicz, M., Denil, M., Gomez, S., Hoffman, M. W., Pfau, D., Schaul, T., Shillingford, B., and de Freitas, N. Learning to learn by gradient descent by gradient descent, 2016.

Argall, B., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and Autonomous Systems*, 57(5):469 – 483, 2009. ISSN 0921-8890. doi: https://doi.org/10.1016/j.robot.2008.10.024. URL http://www.sciencedirect.com/science/article/pii/S0921889008001772.

Bengio, Y., Bengio, S., and Cloutier, J. Learning a synaptic learning rule. *IJCNN-91-Seattle International Joint Conference on Neural Networks*, 01 2002. doi: 10.1109/IJCNN.1991.155621.

Dietterich, T. G. Hierarchical reinforcement learning with the maxq value function decomposition. *Journal of Artificial Intelligence Research*, 13:227303, Nov 2000. ISSN 1076-9757. doi: 10.1613/jair.639. URL http://dx.doi.org/10.1613/jair.639.

Duan, Y., Andrychowicz, M., Stadie, B., Ho, J., Schneider, J., Sutskever, I., Abbeel, P., and Zaremba, W. One-shot imitation learning. In *Advances in neural information processing systems*, pp. 1087–1098, 2017.

Finn, C., Abbeel, P., and Levine, S. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1126–1135. JMLR. org, 2017.

Graves, A., Fernández, S., Gomez, F., and Schmidhuber, J. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pp. 369–376. ACM, 2006.

Hussein, A., Gaber, M., Elyan, E., and Jayne, C. Imitation learning: A survey of learning methods. *ACM Comput. Surv.*, 50(2):21:1–21:35, April 2017. ISSN 0360-0300. doi: 10.1145/3054912. URL http://doi.acm.org/10.1145/3054912.

Jacobs, R. A., Jordan, M. I., Nowlan, S. J., and Hinton, G. E. Adaptive mixtures of local experts. *Neural Computation*, 3:79–87, 1991.

Kipf, T., Li, Y., Dai, H., Zambaldi, V., Sanchez-Gonzalez, A., Grefenstette, E., Kohli, P., and Battaglia, P. CompILE: Compositional imitation learning and execution. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 3418–3428, Long Beach, California, USA, 09–15 Jun 2019. PMLR. URL http://proceedings.mlr.press/v97/kipf19a.html.

Konidaris, G., Kuindersma, S., Grupen, R., and Barto, A. Robot learning from demonstration by constructing skill trees. *The International Journal of Robotics Research*, 31 (3):360–375, 2012.

Kulkarni, T., Narasimhan, K., Saeedi, A., and Tenenbaum, J. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pp. 3675–3683, 2016.

Laskey, M., Lee, J., Fox, R., Dragan, A., and Goldberg, K. Dart: Noise injection for robust imitation learning, 2017.

Mu, T., Goel, K., and Brunskill, E. Plots: Procedure learning from observations using subtask structure. In *Proceedings of the 18th International Conference on Autonomous Agents and MultiAgent Systems*, pp. 1007–1015. International Foundation for Autonomous Agents and Multiagent Systems, 2019.

Rusu, A., Colmenarejo, S., Gulcehre, C., Desjardins, G., Kirkpatrick, J., Pascanu, R., Mnih, V., Kavukcuoglu, K., and Hadsell, R. Policy distillation, 2015.

Santoro, A., Bartunov, S., Botvinick, M., Wierstra, D., and Lillicrap, T. One-shot learning with memory-augmented neural networks, 2016.

Schaal, S. and Atkeson, C. Learning control in robotics. *IEEE Robotics & Automation Magazine*, 17(2):20–29, 2010.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q. V., Hinton, G. E., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *CoRR*, abs/1701.06538, 2017. URL http://arxiv.org/abs/1701.06538.

Shiarlis, K., Wulfmeier, M., Salter, S., Whiteson, S., and Posner, I. Taco: Learning task decomposition via temporal alignment for control. In *International Conference on Machine Learning*, July 2018.

Stolle, M. and Precup, D. Learning options in reinforcement learning. In *Abstraction, Reformulation, and Approximation*, pp. 212–223, Berlin, Heidelberg, 2002. Springer Berlin Heidelberg. ISBN 978-3-540-45622-3.

Xu, D., Nair, S., Zhu, Y., Gao, J., Garg, A., Fei-Fei, L., and Savarese, S. Neural task programming: Learning to generalize across hierarchical tasks. In *IEEE International Conference on Robotics and Automation*, pp. 1–8. IEEE, 2018.

Zhao, J., Li, W., Mao, X., Hu, H., Niu, L., and Chen, G. Behavior-based ssvep hierarchical architecture for telepresence control of humanoid robot to achieve full-body movement. *IEEE Transactions on Cognitive and Developmental Systems*, 9(2):197–209, June 2017. doi: 10.1109/TCDS.2016.2541162.

A. APPENDIX

A.1. Overview

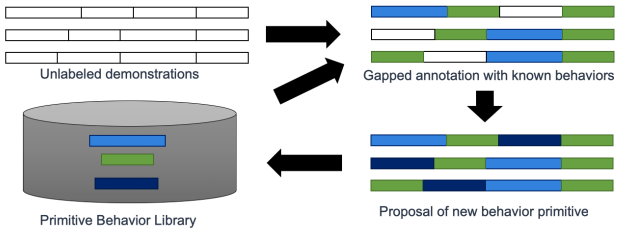


Figure 4. An overview of $PICO$. The approach takes as input unlabeled demonstrations and a library of primitive behaviors. The goal is to predict the primitive behavior label associated with each time point in all demonstrations. Additional behavior primitive models can be trained to fill gaps that are not well represented by existing behavior primitives.

A.2. Additional Related Work

In meta-learning a model is trained on a variety of learning tasks and the parameters of the method are fine-tuned for generalization. The idea of meta-learning is to combine a set of learner models to improve performance on a task more quickly than one without pretrained models. This is a common strategy for one-shot (Santoro et al., 2016) or few shot scenarios, where a model must be trained using one or a few examples. Some approaches for meta-learning come from the reinforcement learning (Finn et al., 2017), which typically differ in how they update individual learners. Some meta-learning methods update models using gradient information (Finn et al., 2017) and others learn how to update learners from data (Andrychowicz et al., 2016; Bengio et al., 2002).

Imitation learning alone does not provide a mechanism to generalize demonstrations to new tasks. One mechanism to address this challenge is task decomposition, which has the goal of identifying subtasks from demonstration. Subtasks can be made into sub-policies through imitation learning, including methods methods that combine subtask discovery with imitation learning (Shiarlis et al., 2018; Xu et al., 2018). By decomposing demonstrations into subtasks, it becomes possible to permute the sequence of sub-policies to achieve greater task diversity and generalizability. However, decomposing demonstrations into subtasks that are maximally useful for recombination is a challenge in task decomposition (Shiarlis et al., 2018).

Once sub-task policies are established, a hierarchical control policy can be learned that identifies the sequence of policies needed to achieve a specified goal. Given a sufficiently diverse set of demonstrations the reasoning layer can be learned from a set of demonstrations (Xu et al., 2018). Several approaches for learning hierarchical architectures for control policies from limited demonstrations have been proposed (Shiarlis et al., 2018; Xu et al., 2018; Duan et al., 2017).

Some approaches assume that the behavior primitive library is fully trained in advance (Xu et al., 2018). In the reinforcement learning domain, the options framework (Stolle & Precup, 2002; Andreas et al., 2017; Kulkarni et al., 2016) and hierarchical reinforcement learning (Dietterich, 2000) are common approaches for organising hierarchies of policies. The techniques in reinforcement learning are often predicated on being able to interact with an environment and collect a lot of data. In this work, we focus on learning hierarchical task decomposition strategies from a limited set of demonstrations.

A.3. Reducing the impact of covariate shift

Following optimization, covariate drift can cause errors in the control process that can place the robot in a previously unobserved state. Control policies will have higher action prediction errors in parts of the state space that it has not observed, leading to poor action predictions and compounding errors with increased iterations of the policy. One approach that has been introduced to decrease the impact of covariate shift is to introduce noise into the demonstrations used for learning (Laskey et al., 2017). This approach increases the amount of state space covered by the policy and improves action predictions around the demonstrations, leading to better generalization and error tolerance.

A.4. Related Approaches for Comparison

In this section we describe the approaches most closely aligned with our work referred to as CTC (Graves et al., 2006) and TACO (Shiarlis et al., 2018).

A.5. Task Sketch for Sub-policy Discovery

Some related approaches (Andreas et al., 2017; Mu et al., 2019) perform demonstration decomposition by combining both demonstrations and task sketches. The literature refers to these approaches as *weakly-supervised* because the order of tasks is given and the exact transition points within a demonstration must be inferred.

Let $\mathcal{D}$ be our dataset containing trajectories $\rho = ((s_0, a_0), (s_1, a_1), \ldots, ((s_T, a_T))$ of length T containing state-action tuples (s, a) for state s and action a. A task sketch $\tau = (\tau_1, \tau_2, \ldots, \tau_L)$ is a sequence of sub-tasks labels where L is the length of the sketch. A path is a sequence of sub-task labels $\zeta = (\zeta_1, \zeta_2, \ldots, \zeta_T)$ where T is the length of a demonstration. We assume that $L << T$. We say that a path ζ matches a task sketch τ if $\tau = \zeta$ after removing all adjacent duplicate sub-tasks in ζ. For example, the path $(\pi_2, \pi_2, \pi_2, \pi_3, \pi_3, \pi_1, \pi_1, \pi_1, \pi_1)$ matches the task sketch (π_2, π_3, π_1).

Connectionist Temporal Classification Given a dataset $\mathcal{D}$ and task sketch τ, one approach to obtain a set of generalizable sub-tasks $\mathcal{B}$ is to separately learn alignment of trajectories to the task sketch then learn the control policies for sub-tasks with behavior cloning. Connectionist Temporal Classification (CTC) (Graves et al., 2006) addresses the problem of aligning sequences of dissimilar lengths. There are potentially multiple ways in which a path could be aligned to a task sketch. Let $\mathbb{Z}_{(T,\tau)}$ be the set of all paths of length T that match the task sketch τ. The CTC objective maximises the probability of the task sketch τ given the input trajectory ρ:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{(\rho,\tau)}[p_\theta(\tau|\rho)] \tag{6}$$

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, \mathbb{E}_{(\rho,\tau)}[\sum_{\zeta \in \mathbb{Z}_{(T,\tau)}} \prod_{t=1}^{T} p_\theta(\zeta_t, |\rho_t)] \tag{7}$$

$p_\theta(\zeta_t, |\rho)$ is commonly represented as a neural network with parameters θ that outputs the probability of each sub-task policy in $\mathcal{B}$. The objective is solved efficiently using dynamic programming. Inference using the neural network model is used to find a maximum likelihood path ζ for a trajectory ρ. The labels in ζ provide an association between state-action tuples (s_t, a_t) and subtask policies $\pi \in \mathcal{B}$. The state-action policies associated with a single sub-task are used to create a sub-task policy using behavior cloning.

Temporal Alignment for Control Given a demonstration ρ and a task sketch τ, Temporal Alignment for Control (TACO) (Shiarlis et al., 2018) will learn where each subtask begins and ends in the trajectory and simultaneously trains a library of sub-tasks policies $\mathcal{B}$. TACO maximizes the joint log likelihood of the task sequence τ and the actions from sub-task policies contained in $\mathcal{B}$ conditioned on the states. Let $\mathbf{a}_\rho$ and $\mathbf{s}_\rho$ be the set of actions and states respectively in trajectory ρ.

$$p(\tau, \mathbf{a}_\rho|\mathbf{s}_\rho) = \sum_{\zeta \in \mathbb{Z}_{(T,\tau)}} p(\zeta|\mathbf{s}_\rho) \prod_{t=1}^{T} \pi_{\zeta_t}(a_t|s_t) \tag{8}$$

where $p(\zeta|\mathbf{s}_\rho)$ is the product of action probabilities associated with any given path ζ. The path ζ determines which data within ρ corresponds to each sub-task policy π and $\prod_{t=1}^{T} \pi_{\zeta_t}(a_t|s_t)$ is the behavior cloning objective from Equation 1.

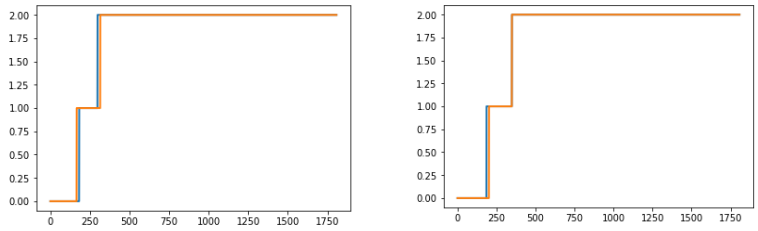
A.6. Domains

As an example of this scenario, consider a Universal Robots UR5 (UR) manipulator mounted on a Clearpath Husky platform (Hus) as shown in Fig. 1(a). The UR5 is used to demonstrate reaching, grabbing, and lifting a block on a table. Other tasks may require performing these actions in another order, so it may be useful to learn and maintain a collection of these primitive behaviors for later use. While the underlying behavior primitives are well defined for the reach-and-grasp

scenario, other example scenarios may not have as well defined or labeled primitives. In this work, we assume that the underlying label of the behaviors shown in the task demonstrations is unknown.

A.7. Reconstruction from existing primitives

Our initial experiment is an ablation study that separately evaluates the estimate of the primitive behavior probability distribution and the action predictions from learning behavior primitives. We train and freeze behavior primitive models for *reach*, *grasp*, and *lift* using the ground truth labeled data from trajectories. We evaluated $PICO$, TACO (Shiarlis et al., 2018), and CTC based on label classification accuracy. For Taco and CTC we additionally compared the methods using MLP and RNN based underlying network models. We evaluated all methods based on an 80/20 split of demonstrations into training and test sets. The average of five independent runs were obtained for each approach. In Table 1, we show the results of the comparison. As shown in the sample trajectory in Figure 5(b), the label prediction of the trained model closely aligns with the ground truth label from the example trajectory.



(a) Sample trajectory label accuracy (b) Missing primitive label accuracy

Figure 5. Example behavior primitive label accuracy for a single test demonstration. We compared the label predictions given by $PICO$ (red) to the ground truth (blue) (a) A sample reconstruction for a single trajectory with an existing behavior primitive library. Timepoints are on the x-axis. and behavior primitive label is on they y-axis. The labels 0,1, and 2 correspond to reach, grasp, and lift respectively. (b) Reconstruction of an example trajectory and discovery of a missing behavior primitive (grasp).

Figure 5(a), shows a comparisons between between the predicted label based on Equation 5 and the ground truth label. Over all trajectories in the test set, the average label classification accuracy was 96% compared to the ground truth label. The summary of results are shown in Table 1.

A.8. Visualizing the Learned Latent Space

To better understand the role of the embedding space for predicting the primitive probability distribution, we visualized the embedding of all states vectors from the test set in the recurrent hidden layer. We would expect that a useful latent embedding would naturally cluster states that correspond to different primitives into distinct locations in the embedding space. Figure 6 shows layout of the latent space in two dimensions. Each point corresponds to a state vector from the test

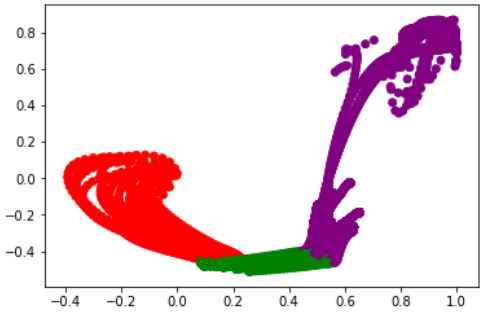


Figure 6. The organization of the learned latent space associated with the Husky-UR5 dataset for reach, grasp, and lift (red, green, and purple respectively).

dataset. The points are colored by the ground truth label.

A.9. Additional Discussion

For the dialpad dataset, label prediction accuracy is a challenging metric without a task sketch because the starting position of the jaco may not provide clues about which button will be pressed. As the jaco gets closer to a button, it becomes more clear which button will be pressed.

Also of note, we compare our results to TACO which is a weakly supervised approach. TACO is given the ordering of tasks. For task sequences of length 4, this means that a random baseline would be expected to achieve an accuracy of 25%. For an unlabeled approach like $PICO$, any of the 10 behavior primitives could be selected at each timepoint. This means that unlabeled demonstrations the expected accuracy of a random baseline would be 10%.