Learning Robot Skills with Temporal Variational Inference

Tanmay Shankar [1] **Abhinav Gupta** [1]

Abstract

In this paper, we address the discovery of robotic options from demonstrations in an unsupervised manner. Specifically, we present a framework to jointly learn low-level control policies and higher-level policies of how to use them from demonstrations of a robot performing various tasks. By representing options as continuous latent variables, we frame the problem of learning these options as latent variable inference. We then present a temporal formulation of variational inference based on a temporal factorization of trajectory likelihoods, that allows us to infer options in an unsupervised manner. We demonstrate the ability of our framework to learn such options across three robotic demonstration datasets.

1. Introduction

The robotics community has long since sought to acquire general purpose and reusable robotic skills, to enable robots to execute a wide range of tasks. The idea of such skills is attractive; by abstracting away the details of low-level control and reasoning over high-level skills instead, a robot can address more complex and longer term tasks. Further, the ability to *compose* skills leads to a combinatorial increase in the robot's capabilities (Leslie Pack Kaelbling, 2017), ideally spanning the abilities required for the robot to complete the desired tasks. For example, a robot equipped with reaching, grasping, and pouring skills could compose them to make a cup of tea as well as pour cereal into a bowl.

Indeed, the promise of skills has been explored in several contexts, be it options in reinforcement learning (RL) (Sutton et al., 1999), operators in the planning (Fikes & Nilsson, 1971), primitives and behaviours in robotics (Muelling et al., 2010), or abstractions (Kim et al., 2019). Nomenclature aside, these ideas share the notion of eliciting a certain pattern or template of action in response to a particular situation, differing in their exact implementations and how these skills are obtained. For example, while previous works (Muelling et al., 2010; Mülling et al., 2013; Konidaris & Barto, 2009) manually defined these skills, more recent approaches have sought to *learn* these skills, either from interaction (Kulkarni et al., 2016) or from demonstrations (Xu et al., 2018; Huang et al., 2019; Fox et al., 2017; Krishnan et al., 2017; Kipf et al., 2019).

Learning skills from demonstrations is appealing, since it allows non-robotics-expert humans to demonstrate solving the target tasks, bypassing the tedium of manually specifying these skills or carefully engineering solutions to the tasks (Argall et al., 2009). But even using demonstrations, approaches such as Xu et al. (2018); Huang et al. (2019) require heavy supervision such as segmentation of demonstration data. Instead, learning skills in an unsupervised, data-driven manner not only avoids having to annotate demonstrations; it also enables the use of large scale and diverse demonstration data in robotics (Sharma et al., 2018; Mandlekar et al., 2018). This in turn enables learning a correspondingly diverse set of skills, as well as how to use them to achieve a variety of tasks.

Aside from the issue of learning and representing individual skills, is the notion of *composing* them. The efficacy of these skills is rather limited when used in isolation; by selecting and sequencing the appropriate skills however, a robot can achieve a variety of complex tasks, as illustrated by the tea and cereal example. A rich body of literature addresses composing skills, including sequencing skills (Neumann et al., 2014; Peters et al., 2013), hierarchical approaches (Lioutikov et al., 2017; Xu et al., 2018; Huang et al., 2019; Konidaris et al., 2012; Konidaris & Barto, 2009; Shankar et al., 2020), options (Sutton et al., 1999), etc. Some works among these (Fox et al., 2017; Krishnan et al., 2017; Kipf et al., 2019) address jointly learning skills and how to compose them. Jointly learning both levels of this hierarchy not only addresses how to use these skills, but also allows for adapting the skills based on how useful they are to the task at hand.

Unfortunately, these works have their own limitations. While Neumann et al. (2014); Niekum et al. (2012); Konidaris & Barto (2009) do learn to sequence skills, they

[*]Equal contribution [1]Facebook AI Research, Pittsburgh, PA, USA. Correspondence to: Tanmay Shankar <tanmayshankar@fb.com>.

Proceedings of the 37^{th} International Conference on Machine Learning, Vienna, Austria, PMLR 119, 2020.

assume restrictive primitive representations - such as DMPs (Ijspeert et al., 2013). While Shankar et al. (2020) learn continuous representations of primitives, it requires an additional post hoc policy learning step. Fox et al. (2017); Krishnan et al. (2017) do afford directly usable policies, but critically are restricted to a fixed number of discrete options.

In this paper, we propose a framework to jointly learn options and how to compose and use them from demonstrations in an unsupervised manner. At the heart of our framework is a *temporal variational inference* (TVI) based on a corresponding factorization of trajectory likelihoods. We adopt a latent variable representation of options; this allows us to treat the problem of inferring options as inferring latent variables, which we may then accomplish via our temporal variational inference. Specifically, optimizing the objective afforded by our temporal variational inference with respect to the distributions involved naturally gives us low and high-level policies of options.

We evaluate our approach's ability to learn options across three datasets, and demonstrate that our approach can learn a meaningful space of options that correspond with traditional skills in manipulation, visualized at https://sites.google.com/view/learning-causal-skills/home. We quantify the effectiveness of our policies in solving downstream tasks, evaluated on a suite of tasks.

2. Related Work

Learning from Demonstrations: Learning from Demonstrations (LfD) addresses solving tasks by learning from demonstrations of the task being solved by an expert. This may be accomplished by simply cloning the original demonstration (Esmaili et al., 1995), or fitting the demonstration to a trajectory representation (Kober & Peters, 2009; Peters et al., 2013) or a policy (Schaal, 1997). More recent efforts have sought to segment demonstrations into smaller behaviors, and fitting a model to the resulting segments (Niekum et al., 2012; Krishnan et al., 2018; Murali et al., 2016; Meier et al., 2011). Argall et al. (2009) presents a thorough review of these techniques. Our work falls into the broad paradigm of LfD, but seeks to learn hierarchical policies from demonstrations.

Sequencing Primitives: The concept of learning movement primitives using predefined representations of primitives (Kober & Peters, 2009; Peters et al., 2013) has been a popular technique to capturing robotic behaviors. A natural next step is to sequence such primitives to perform longer horizon downstream tasks, (Neumann et al., 2014; Niekum et al., 2012; Muelling et al., 2010). Konidaris & Barto (2009); Konidaris et al. (2012) address merging these skills into skill-trees. Lioutikov et al. (2017; 2020) address sequencing primitives using probabilistic segmentation and attribute grammars respectively. Our work differs from a majority of these works in that we jointly learn both the representation of primitives and how these primitives must be sequenced.

Hierarchical Policy Learning: Fox et al. (2017); Krishnan et al. (2017); Kipf et al. (2019); Sharma et al. (2019) also address the problem of learning options from demonstrations without supervision. However these works are restricted to a discrete set of options, which must be pre-specified. Further, Fox et al. (2017); Krishnan et al. (2017) employ a forward-backward algorithm for inference that requires an often intractable marginalization over latents. The CompILE framework (Kipf et al., 2019) makes use of continuous latent variables to parameterize options, but requires carefully designed attention mechanisms, and is evaluated in relatively low-dimensional domains. Smith et al. (2018) and Bacon et al. (2017) derive policy gradients to address hierarchical policy learning in the RL setting.

Learning Trajectory Representations: Shankar et al. (2020) learning representations of primitives, but need to adopt an additional phase of training to produce usable policies. Co-Reyes et al. (2018) also approach hierarchical RL from a trajectory representation perspective. Our work shares this notion of implicitly learning a representation of primitives, but differs in that we do not require an additional high-policy learning step to perform hierarchical control.

Learning Temporal Abstractions: Kim et al. (2019) and Gregor et al. (2019) both address learning temporal abstractions in the form of 'jumpy' transitions. Kim et al. (2019) seeks to learn a generic partitioning of the input sequence into temporal abstractions, while Gregor et al. (2019) learns temporal abstractions with beliefs over state to capture uncertainity about the world. While both these works adopt variational bounds of a similar form to ours, our bound objective is derived in terms of usable option *policies* rather than abstract or belief states.

Compositional Policies: Both Xu et al. (2018) and Huang et al. (2019) address unseen manipulation tasks by learning compositional policies, but require heavy supervision to do so. Andreas et al. (2017) and Shiarlis et al. (2018) compose modular policies in the RL and LfD settings respectively, using policy sketches to select which policies to execute. While our work approaches doesn't explicitly address compositionality, we too seek to benefit from the benefits of such compositionality.

3. Approach

3.1. Preliminaries

Throughout our paper, we adopt an undiscounted Markov Decision Process without rewards (denoted as MDP\R). An

Algorithm 1 Trajectory Generation Process with Options

Input: Low-level Policy π, High-level Policy η, Initial State Distribution $d_1(s)$,
Output: Trajectory τ

$s_1 \sim d_1, b_1 \leftarrow 1$ ▷ Initialize state.
for $t \in [1, 2, .., T]$ **do**
  if $b_t = 1$ **then**:
    $z_t \sim \eta(z|s_{1:t}, a_{1:t-1}, z_{1:t-1})$ ▷ Select option.
  else:
    $z_t \leftarrow z_{t-1}$ ▷ Continue previous option.
  $a_t \sim \pi(a|s_{1:t}, a_{1:t-1}, z_{1:t})$ ▷ Select action.
  $s_{t+1} \sim p(s_{t+1}|s_t, a_t)$ ▷ Execute action.
  $b_{t+1} \sim \beta(s_{t+1})$ ▷ Decide whether to terminate.
$\tau \leftarrow \{s_t, a_t\}_{t=1}^T, \zeta \leftarrow \{z_t, b_t\}_{t=1}^T$

MDP\R is a tuple $\mathcal{M} : \langle S, A, P \rangle$, that consists of states s in state space S, actions a in action space A, and a transition function between successive states $P(s_{t+1}|s_t, a_t)$.

Options: An option $\omega \in \Omega$ (Sutton et al., 1999) formally consist of three components - an initiation set $\mathcal{I}$, a policy $\pi : S \rightarrow A$, and a termination function $\beta : S \rightarrow [0, 1]$. When an option is invoked in a state in $\mathcal{I}$, policy π is executed until the termination function dictates the option should be terminated (i.e., $\beta(s) = 1$). As in Fox et al. (2017); Smith et al. (2018), we assume options may be initiated in the entire state space.

Options as Latent Variables: We assume that the identity of an option being executed is specified by a latent variable z, that may be either continuous or discrete. We also consider that at every timestep, a high-level policy $\eta : S \rightarrow \Omega \times [0, 1]$ selects the identity z_t of the option to be invoked, as well as a binary variable b_t of whether to terminate the option or not. This option informs the low-level policy π's choice of action, constructing a trajectory as per the generative process in algorithm 1. We denote the sequence of options executed during a trajectory as a sequence of these latent variables, $\zeta = \{z_t, b_t\}_{t=1}^T$.

3.2. Decomposition of trajectory likelihood

Consider a trajectory $\tau = \{s_t, a_t\}_{t=1}^T$, and the sequence option sequence that generated it $\zeta = \{z_t, b_t\}_{t=1}^T$. The joint likelihood $p(\tau, \zeta)$ of the trajectory and these options under the generative process described in algorithm 1 may be expressed as follows:

$$p(\tau, \zeta) = p(s_1) \prod_{t=1}^{T} \eta(\zeta_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t-1}) \\ \pi(a_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t}) p(s_{t+1}|s_t, a_t) \quad (1)$$

The distributions $\eta(\zeta_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t-1})$ and $\pi(a_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t})$ implicitly capture the causal restriction that future variables (ex. $s_{t+1:T}$) do not influence earlier variables (ex. $a_{1:t}$). Further, both π and η may be queried at any arbitrary time t with the information available till that time. Ziebart et al. (2013) formalized the notion of *causally conditioned distributions* to represent such distributions, a notion that plays a part in the formulation of our approach. We refer the reader to Ziebart et al. (2013); Kramer (1998) for a more thorough treatment of this concept.

3.3. Temporal Variational Inference

To reiterate, our goal is to learn options and how to use them from demonstrations; this formally corresponds to learning low and high level policies π and η, from a dataset of N demonstrations, $\mathcal{D} = \{\tau_i\}_{i=1}^N$. This is equivalent to inferring latent variables ζ from a trajectory, since policies π and η essentially reason about the choice of ζ and its effect on the choice of actions. The representation of options as latent variables ζ we adopt hence allows us to view the problem of inferring options from a perspective of inference of latent variables.

This allows us to employ unsupervised latent variable inference techniques; we employ a variant of variational inference (VI) (Kingma & Welling, 2013) to infer latent options ζ. Our choice of VI over the forward-backward style algorithm employed in Fox et al. (2017); Krishnan et al. (2017) is because VI is amenable to both continuous and discrete latent variables z. Further, VI bypasses the often intractable marginalization over latents in favor of sampling based approach.

In standard VI, a variational distribution $q(z|x)$ is used to infer latents z from observed data x, approximating the unknown conditional $p(z|x)$. One then optimizes the likelihood of observations given the predicted latents under a learned decoder $p(x|z)$. In our case, we seek to infer the *sequence* of options $\zeta = \{z_t, b_t\}_{t=1}^T$ from a trajectory $\tau = \{s_t, a_t\}_{t=1}^T$; we estimate the conditional $p(\zeta|\tau)$ with a variational approximation $q(\zeta|\tau)$.

To retrieve usable policies that can be queried at inference or "test" time, we require policies that can reason about the current choices of option ζ_t and action a_t given the observations available *so far*, i.e. $s_{1:t}$, $a_{1:t-1}$, and $\zeta_{1:t-1}$. This precludes optimizing the conditional $p(\tau|\zeta)$ as would be done in standard VI. Instead, we optimize the joint likelihood $p(\tau, \zeta)$ of trajectories and latents with respect to the causally conditioned π and η. Not only does this afford us directly usable policies as desired, but this objective naturally arises from the variational bound constructed below.

We now formally present temporal variational inference, a variant of VI suitable for our sequential latent variables,

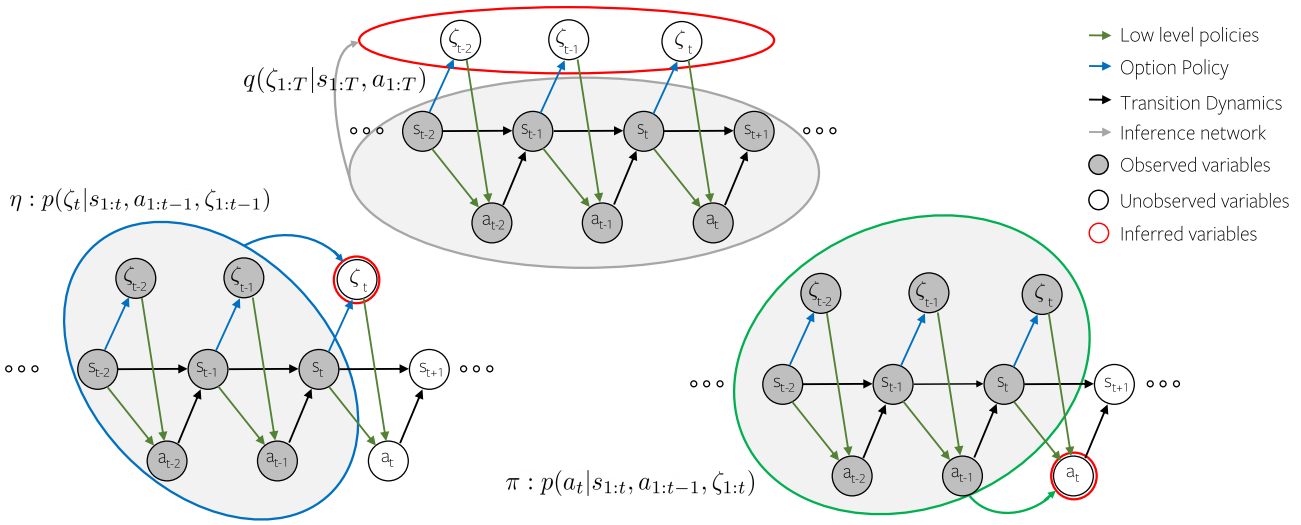


Figure 1. Depiction of the key distributions q, π, and η, and the probabilistic graphical model underlying our approach. Note the dependence between trajectories and options. We also depict the variables that each of the three networks reason about, and the information they make use of to do so.

that accounts for the causal restriction of these latent variables based on the decomposition of trajectory likelihood in section 3.2. We begin with the standard objective of maximizing the log-likelihood of trajectories across the dataset, $\mathcal{L} = \mathbb{E}_{\tau\sim\mathcal{D}}\big[\log p(\tau)\big]$. $\mathcal{L}$ is lower bounded by J, where:

$$\begin{aligned} J &= \mathbb{E}_{\tau\sim\mathcal{D}}\big[\log p(\tau)\big] - D_{KL}\big[q(\zeta|\tau)||p(\zeta|\tau)\big] \\ &= \mathbb{E}_{\tau\sim\mathcal{D}}\big[\log p(\tau)\big] - \mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\log \frac{q(\zeta|\tau)}{p(\zeta|\tau)}\Big] \\ &= \mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\log p(\tau) + \log p(\zeta|\tau) - \log q(\zeta|\tau)\Big] \\ &= \mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\log p(\tau,\zeta) - \log q(\zeta|\tau)\Big] \end{aligned}$$

Substituting the joint likelihood decomposition from eq. (1) above yields the following objective:

$$\begin{aligned} J = \mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\sum_t \{&\log \eta(\zeta_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t-1}) \\ &+ \log \pi(a_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t}) + \log p(s_{t+1}|s_t, a_t)\} \\ &+ \log p(s_1) - \log q(\zeta|\tau)\Big] \end{aligned} \tag{2}$$

Assuming distributions π, η, and q are parameterized by θ, ϕ, and ω respectively, we may optimize these using standard gradient based optimization of J:

$$\begin{aligned} \nabla J = \nabla_{\theta,\phi,\omega}\mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\sum_t \{&\log \pi(a_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t}) \\ &+ \log \eta(b_t, z_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t-1})\} - \log q(\zeta|\tau)\Big] \end{aligned} \tag{3}$$

Note that the dynamics $p(s_{t+1}|s_t, a_t)$ and initial state distribution $p(s_1)$ factor out of this gradient, as derived in the supplementary material.

Parsing the objective J: We provide a brief analysis of how objective J and its implied gradient update eq. (3) jointly optimizes π, η, and q to be consistent with each other. Three interacting terms optimize q. The first two terms encourage q to predict options ζ that result in high likelihood of actions a_t under π, and that are likely under the current estimate of the high level policy η. The final $-\log q(\zeta|\tau)$ term encourages maximum entropy of q, discouraging q from committing to an option unless it results in high likelihood under π and η. This entropy term also prevents q from trivially encoding all trajectories into a single option.

Low-level policy π is trained to increase the likelihood of selecting actions a_t given the current option being executed ζ_t. High-level policy η is trained to mimic the choices of options made by the variational network (i.e. options that result in high likelihood of demonstrated actions), only using the available information at time t to do so.

Reparameterization: While eq. (3) can be implemented via REINFORCE (Williams, 1992), we do so only for inferring discrete variables $\{b_t\}_{t=1}^T$. As in standard VI, we exploit the continuous nature of z's and employ the reparameterization trick (Kingma & Welling, 2013) to enable efficient learning of $\{z_t\}_{t=1}^T$. Rather than sample latents $\{z_t\}_{t=1}^T$ from a stochastic variational distribution $q(\zeta|\tau)$, $\{z_t\}_{t=1}^T$ is parameterized as a differentiable and deterministic function of the inputs τ and a noise vector ϵ, drawn from an appropriately scaled normal distribution. In practice, we parameterize q as an LSTM that takes τ as input, and predicts

Algorithm 2 Temporal Variational Inference for Learning Skills

Input: $\mathcal{D}$ ▷ Require a demonstration dataset
Output: π, η ▷ Output low and high-level policies
1: Initialize $\pi_\theta, \eta_\phi, q_\omega$ ▷ Initialize networks
2: Pretrain π as VAE ▷ Pretrain latent representation
3: **for** $i \in [1, 2, ..., N_{\text{iterations}}]$ **do**
4: $\tau_i \leftarrow \mathcal{D}$ ▷ Retrieve trajectory from dataset
5: $\zeta \sim q(\zeta|\tau_i)$ ▷ Sample latent sequence from variational network
6: $J \leftarrow \sum_t \log \pi(a_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t}) + \sum_t \log \eta(\zeta_t|s_{1:t}, a_{1:t-1}, \zeta_{1:t-1}) - \log q(\zeta|\tau)$ ▷ Evaluate likelihood objective under current policy estimates
7: Update $\pi_\theta, \eta_\phi, q_\omega$ via $\nabla_{\theta,\phi,\omega} J$

mean μ_t and variance σ_t of the distribution $q(\{z_t\}_{t=1}^T|\tau)$. We then retrieve $\{z_t\}_{t=1}^T$ as $\{z_t = \mu_t + \sigma_t \epsilon_t\}_{t=1}^T$.

In our case, gradients flow from our objective J through *both* the low and high-level policies π and η to the variational network q. This in contrast with standard VI, where gradients pass through the decoder $p(x|z)$. This reparameterization enables the efficient gradient based learning of q based on signal from both the low and high-level policies.

Features of objective: The temporal variational inference we present has several desirable traits that we describe below.
(1) First and foremost, J provides us with causally conditioned low and high-level policies π and η. These policies are directly usable at inference time towards solving downstream tasks, since they are only trained with information that is also available at inference time.
(2) With TVI, we can adopt a continuous parameterization of options, $z \in \mathbb{R}^n$. This eliminates the need to pre-specify the number of options required; instead we may learn as many options as are required to capture the behaviors observed in the data, in a data driven manner. The continuous space of options also allows us to reason about how similar the various learned options are to one another (allowing substitution of options for one another).
(3) The joint training of π, η, and q implied by J not only allows training the high-level policy η to based on the available options, but also allows the adaptation of the low-level options π based on how useful they are to reconstructing a demonstration.
(4) The objective J is also amenable to gradient based optimization, allowing us to learn options efficiently.

3.4. Learning Skills with Temporal VI

Equipped with this understanding of our temporal variational inference (TVI), we may retrieve low and high-level policies π and η by gradient based optimization of the objective J presented. We first make note of some practical considerations required to learn skills with TVI, then present a complete algorithm to learn skills using TVI.

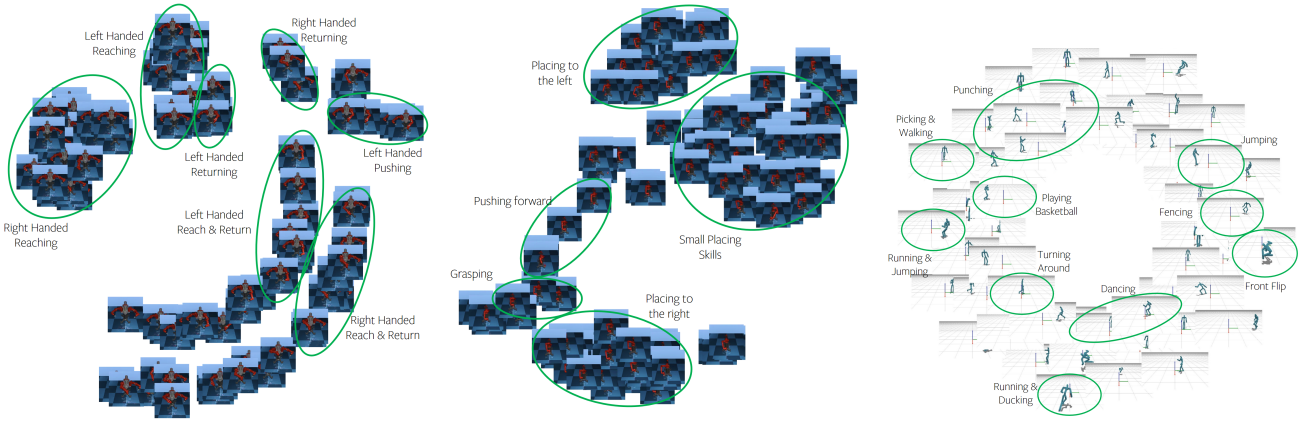
Policy Parameterization: We parameterize each of the policies π and η as LSTMs (Hochreiter & Schmidhuber, 1997), with 8 layers and 128 hidden units per layer. The recurrent nature of the LSTM naturally captures the causal nature of π and η. In contrast, q is parameterized as a bi-directional LSTM, since q reasons about the sequence of latents ζ given the entire trajectory τ. q, π and η all take in a concatenation of trajectory states and actions as input. π and η also take in the sequence of latents until the current timestep as input. Since η reasons about the choice of latents to solve the task occurring in the demonstration, η also takes in additional information about the task, such as task ID and object-state information. π predicts the mean μ_a and variance σ_a of a Gaussian distribution from which actions $a \in \mathcal{A}$ are drawn. η and q both predict mean μ_z and variance σ_z of a Gaussian distribution from which latent variables z are drawn. η and q also predict the probability of terminating a particular option $p(b)$, from which binary termination variables b are drawn. While q predicts the entire sequence of ζ's, latents are retrieved from η during a rollout via the generative process in algorithm 1.

Pretraining the Low-level Policy: Optimizing our joint objective J with randomly initialized low-level policies results in our training procedure diverging. During initial phases of training, the random likelihoods of actions and options under the random initial policies provide uninformative gradients. To counteract this, we initialize the low-level policy to capture some meaningful skills, by pretraining it to reconstruct demonstration segments in a VAE setting. Specifically, we draw trajectory segments from the dataset and encode them as a single latent z (i.e. a single option). We train the low-level policy (i.e. as a decoder) to maximize the likelihood of the actions observed in this trajectory segment given the latent z predicted by the encoder.

Algorithm: We present the full algorithm for learning skills via temporal variational inference in algorithm 2. After the pre-training step described above, there are three steps that occur every iteration in training. (1) For every trajectory, a corresponding sequence of latent variables ζ is sampled from the varitional network q. (2) This likelihood of this estimated sequence of latents ζ is evaluated under the current policy estimates, giving us objective J. (3) The gradients of J are then used to update the three networks π, η, and q.

4. Experiments

We would like to understand how well our approach can discover options from a set of demonstrations in an un-



(a) Latent space of skills for MIME dataset. Note the clustering of skills into left and right handed reaching, returning and sliding skills, along with additional hybrid skills.

(b) Latent space of skills for Roboturk dataset. Note the clustering of skills into single armed pushing, grasping, and placing in different locations and to different extents.

(c) Latent space of skills for Mocap dataset. While less structured than (a) and (b), the space consists of diverse skills ranging such as running, front flips and punching.

Figure 2. Visualization of the learned latent space of skills for the (a) MIME dataset, (b) Roboturk dataset, and (c) Mocap dataset. Note the emergence of clusters of skills in each case.

supervised manner, and quantify how useful the learnt policies are for solving a set of target tasks. To this end, we evaluate our approach across the three datasets described below, as well as a suite of simulated robotic tasks. We present visualizations of the results of our model at https://sites.google.com/view/learning-causal-skills. We first describe the datasets used below.

MIME Dataset: The MIME Dataset (Sharma et al., 2018) consists of 8000+ kinesthetic demonstrations across 20 tasks (such as pushing, bottle-opening, stacking, etc.) collected on a Baxter robot. We use the 16 dimensional joint-angles of the Baxter (7 joints for each of the 2 arms, and a gripper for each arm) as the input and prediction space for our model.

Roboturk Dataset: The Roboturk Dataset (Mandlekar et al., 2018) consists of 2000+ demonstrations collected on a suite of tasks (such as bin picking, nut-and-peg assembly, etc.) by teleoperating a simulated Sawyer robot. These teleoperation demonstrations are hence more noisy than the kinesthetic MIME dataset. We use the 8 dimensional joint angles of the Sawyer (7 arm joints, and a single gripper), as well as the robot-state and object-state vectors provided in Mandlekar et al. (2018) to train our model.

CMU Mocap Dataset: The CMU Mocap Dataset (CMU, 2002) consists of 1953 motions collected by tracking visual markers placed on humans, while performing a variety of different actions. These actions include punching, jumping, performing flips, running, etc. We use the local (i.e., relative to a root node on the agent itself) 3-D positions of each of the 22 joints recorded in the dataset, for a total of 66 dimensions.

Preprocessing and Train-Test Split: In both the MIME and Roboturk datasets, the gripper values are normalized to a range of $\{-1, 1\}$, while joint angles are unnormalized. The trajectories across all datasets are downsampled (in time) by a factor of 20. For each dataset, we set aside 500 randomly sampled trajectories that serve as our test set for our experiments in section 4.2. The remaining trajectories serve as the respective training sets.

4.1. Qualitative Evaluation of Learned Space

The first question we would like to answer is - "Is our approach able to learn a diverse space of options?". We answer this by presenting a qualitative analysis of the space of options learned by our model.

For a set of 100 demonstrations, we retrieve the sequence of latent z's and their corresponding trajectory segments executed over each demonstration from our model. We embed these latent z's in a 2-D space using T-SNE (van der Maaten & Hinton, 2008), and visualize the trajectory segments at their corresponding position in this 2-D embedded space. We visualize the learned embedding spaces for the MIME dataset in fig. 2a, for the Roboturk dataset in fig. 2b, and for the Mocap dataset in fig. 2c. Dynamic visualizations of these figures are available at https://sites.google.com/view/learning-causal-skills.

Note the emergence of *clusters* of skills on the basis of types of motions being executed across these datasets.

In the case of the MIME dataset fig. 2a, we note that the emergent skills are separated on the basis of the nature of motion being executed, as well as the *arm* of the robot being used. This is expected in a bimanual robot, as skills with the left and right hands are to be treated differently, as are skills using both hands. The skills that our approach

Table 1. Trajectory Reconstruction Error of our approach and baselines across various datasets. The baselines were adapted from (1) Kingma & Welling (2013), (2) Ijspeert et al. (2013), (3) Niekum et al. (2012), (4) Shankar et al. (2020).

METHOD	MIME DATA	ROBOTURK DATA	MOCAP DATA
FLAT VAE [1]	0.14	0.23	0.08
FLAT DMP [2]	0.36	0.54	3.45
H-DMP [3]	0.02	0.06	0.01
DISCO-MP [4]	0.02	0.03	0.03
OURS	0.02	0.04	0.04

captures correspond directly to traditional notions of skills in the manipulation community, such as reaching, returning, sliding / pushing, etc. Our approach further distinguishes between left handed reaching and right handed reaching, etc., a useful ability in addressing downstream tasks.

For the Roboturk dataset fig. 2b, the space is separated on the basis of the nature, shape and direction of motion being executed. For example, placing motions to the left and right of the robot appear separately in the space. Additional placing motions that move the arm to a lesser extent also appear separately. The space also captures finer motions such as closing the grippers down (typically in a position above the workspace).

For the Mocap dataset fig. 2c, the learned space is not as clearly structured as in the case of MIME and Roboturk datasets. We believe this is due to the much larger range of motions executed in the dataset, coupled with the high-dimensional nature of the humanoid agent. Despite this lack of structure, a diverse set of skills is still learned. For example, the space consists of running, walking, and jumping skills, which constitute a majority of the dataset. More interesting skills such as fencing, performing flips, and boxing skills were also present and captured well by our model.

In both the MIME and Roboturk datasets, the correspondence of many emergent skills with traditional notions of skills in the manipulation community is indicative that our approach can indeed discover diverse robotic skills from demonstrations without supervision.

4.2. Reconstruction of Demonstrations

We evaluate how well our approach can use the learned skills to reconstruct the demonstrated trajectories; i.e. whether it is able to capture the overall structure of the demonstrations in terms of the skills being executed. We do so quantitatively and qualitatively. Quantitatively, we measure the average state distance (measured by the mean squared error) between the reconstructed trajectory and the original demonstration, across the 500 randomly sampled unseen demonstrations in each dataset. We compare our approach's performance on this metric against a set of baselines:

- **Flat-VAE:** We train an LSTM VAE (Kingma & Welling, 2013) to reconstruct demonstrations. This represents a flat, non-hierarchical baseline with a learned representation. The architecture used is an 8 layer LSTM with 128 hidden units, like our policies.
- **Flat-DMP:** We fit a single Dynamic Movement Primitive (Ijspeert et al., 2013) to each trajectory in the dataset. This represents a non-hierarchical baseline with a predefined trajectory representation.
- **H-DMP:** We evaluate a hierarchical DMP based approach similar to Niekum et al. (2012), that first segments a trajectory by detecting changepoints in acceleration, and then fits individual DMPs to *each* segment of the trajectory. This represents a hierarchical baseline with a predefined trajectory representation.
- **Disco-MP:** We also evaluate the method of Shankar et al. (2020), which is directly optimized for trajectory reconstruction. This approach represents a hierarchical baseline with a learned trajectory representation.
- **Ours:** We obtain predicted trajectories from our model obtaining the latent z's that occur in a demonstration from our q network, and rolling out the low-level policy with these z's as input.

We present the average state distances obtained by our approach against these baselines in table 1. The flat VAE is able to achieve a reasonably low reconstruction error, indicating the benefits of a learnt trajectory representation over a predefined representation such as DMPs. Combinining a learnt trajectory representation with the ability to compose primitives naturally leads to a further decrease in the trajectory reconstruction error, as observed in the Disco-MP baseline and our approach. Note that our approach is able to achieve a similar reconstruction error to that of Disco-MP, which is explicitly optimized to minimize (aligned) state distances, as well as the H-DMP baseline, which heavily overfits to a single trajectory (thus promising low reconstruction error). This demonstrates that our approach indeed captures the overall structure of demonstrations and is able to represent them faithfully. These trends are consistently observed across all three datasets we evaluate on.

To qualitatively analyse how well our approach captures the structure of demonstration, we visualize the reconstructed trajectories predicted by our approach against the corresponding ground truth trajectory. These results are presented in our website: https://sites.google.com/view/learning-causal-skills/home. We observe that our approach is indeed able to capture the rough sequence of skills that occur in the demonstrated trajectories. In the case of the MIME and Roboturk datasets, our model predicts similar reaching, returning, sliding etc. primitives when the ground truth tra-

Table 2. Average Rewards of our approach and baselines on various RL environments, across 100 episodes. Baselines are adapted from (1) Lillicrap et al. (2015), (2) Esmaili et al. (1995), (3) Kulkarni et al. (2016).

Method	SawyerPick-PlaceBread	SawyerPick-PlaceCan	SawyerPick-PlaceCereal	SawyerPick-PlaceMilk	SawyerNut-AssemblyRound	SawyerNut-AssemblySquare
Flat RL [1]	0.11	0.34	0.18	0.14	0.44	0.55
Flat IL [2]	0.60	0.65	0.29	0.67	0.49	1.87
Hierarchical RL (No Init) [3]	0.13	0.11	0.04	0.15	0.36	1.16
Hierarchical RL (W/ Init) [3]	0.41	0.37	0.22	0.34	0.54	1.09
Ours	1.54	1.22	1.54	0.48	1.88	3.62

jectory executes a corresponding primitives. Further, the rough shape of the arms during the predicted skills correlate strongly with the shapes observed in the ground truth trajectories; this is consistent with the quantitative results presented above. Our approach also notably captures fine motions (such as opening and closing of the gripper) in trajectories well. In case of the Mocap dataset, the overall shape and trend of rolled out trajectories align very closely with the original demonstration, showing the use of our approach in learning skills across widely differently structured data and morphology of agents.

4.3. Downstream Tasks

We would also like to evaluate how useful our learned policies are for solving a set of target tasks. This is central to our motivation of jointly learning options *and* how to use them. We test our learned policies on a suite of 6 simulated robotic tasks from the Robosuite (Mandlekar et al., 2018) environment on the Sawyer robot. Since the MIME dataset lacks any object information, we disregard the Baxter tasks in Robosuite (Mandlekar et al., 2018), and instead use tasks from Robosuite for which the Roboturk dataset has demonstrations. In the pick-place tasks, a reward of 1 is given to objects successfully placed in the bin. In the nut-and-peg assembly tasks, a reward of 1 is given for successfully placing a nut. Both tasks also have additional rewards based on conditions of how the task was executed. We point the reader to (Mandlekar et al., 2018) for a full description of these tasks and their reward structure. We compare the performance of our method on these tasks against the following baselines:

- **Flat RL:** We train flat policies on each of the 6 tasks in the reinforcement learning setting, using DDPG (Lillicrap et al., 2015).
- **Flat IL:** We train flat policies to mimic the actions observed in the demonstrations of each of the 6 tasks, and subsequently finetune these policies in the RL setting.
- **Hierarchical RL:** We train hierarchical policies as in (Kulkarni et al., 2016) in the pure RL setting, with (W/ Init) and without (No Init) any prior initialization.
- **Ours:** We fine-tune the low and high-level policies obtained by our model in the RL setting.

All baseline policies are implemented as 8 layer LSTMs with 128 hidden units, for direct comparison with our policies. The RL based approaches are trained with DDPG with the same exploration processes and hyperparameters (such as initializations of the networks, learning rates used, etc.), as noted in the supplementary. We evaluate each of these baselines along with our approach by testing out the learned policies over 100 episodes in the 6 environments, and reporting the average rewards obtained in table 2.

We observe that the Flat RL and Hierarchical RL baselines are unable to solve these tasks, and achieve low rewards across all tasks. This is unsurprising given the difficulty of exploration problem underlying these tasks (Mandlekar et al., 2018). Pretraining policies with imitation learning somewhat allieviates this problem, as observed in the slightly higher rewards of the Flat IL baseline. This is likely because the policies are biased towards actions similar to those seen in the demonstrations. Training hierarchical policies initialized with our approach is able to achieve significantly higher rewards than both the IL and RL baselines consistently across most environments. By providing these policies with suitable notions of skills that extend over several timesteps, we are able to bypass reasoning over low-level actions for 100's of timesteps, thus guiding exploration in these tasks more efficiently.

5. Conclusion

In this paper, we presented a framework for jointly learning robotic skills and how to use them from demonstrations in an unsupervised manner. Our temporal variational inference allows us to construct an objective that directly affords us usable policies on optimization. We are able to learn semantically meaningful skills that correspond closely with the traditional notions of skills observed in manipulation. Further, our approach is able to capture the overall structure of demonstrations in terms of the learned skills. We hope that these factors contribute towards accelerating research in robot learning for manipulation.

Acknowledgements

The authors would like to thank Shubham Tulsiani for valuable discussions on the formulation of the approach, and Jungdam Won and Deepak Gopinath for help with data processing and visualization for the Mocap dataset.

References

Andreas, J., Klein, D., and Levine, S. Modular multitask reinforcement learning with policy sketches. In *ICML*, 2017.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 2009.

Bacon, P.-L., Harb, J., and Precup, D. The option-critic architecture. In *AAAI*, 2017.

CMU. Cmu graphics lab motion capture database. 2002. URL http://mocap.cs.cmu.edu.

Co-Reyes, J. D., Liu, Y., Gupta, A., Eysenbach, B., Abbeel, P., and Levine, S. Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings. *arXiv preprint arXiv:1806.02813*, 2018.

Esmaili, N., Sammut, C., and Shirazi, G. Behavioural cloning in control of a dynamic system. IEEE, 1995.

Fikes, R. E. and Nilsson, N. J. Strips: A new approach to the application of theorem proving to problem solving. *Artificial intelligence*, 1971.

Fox, R., Krishnan, S., Stoica, I., and Goldberg, K. Multi-level discovery of deep options. *arXiv preprint arXiv:1703.08294*, 2017.

Gregor, K., Papamakarios, G., Besse, F., Buesing, L., and Weber, T. Temporal difference variational auto-encoder. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=S1x4ghC9tQ.

Higgins, I., Matthey, L., Pal, A., Burgess, C., Glorot, X., Botvinick, M., Mohamed, S., and Lerchner, A. beta-vae: Learning basic visual concepts with a constrained variational framework.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, November 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

Huang, D.-A., Nair, S., Xu, D., Zhu, Y., Garg, A., Fei-Fei, L., Savarese, S., and Niebles, J. C. Neural task graphs: Generalizing to unseen tasks from a single video demonstration. In *CVPR*, 2019.

Ijspeert, A. J., Nakanishi, J., Hoffmann, H., Pastor, P., and Schaal, S. Dynamical movement primitives: learning attractor models for motor behaviors. *Neural computation*, 25(2):328–373, 2013.

Kim, T., Ahn, S., and Bengio, Y. Variational temporal abstraction. In Wallach, H., Larochelle, H., Beygelzimer, A., dAlché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 32*, pp. 11566–11575. Curran Associates, Inc., 2019. URL http://papers.nips.cc/paper/9332-variational-temporal-abstraction.pdf.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kingma, D. P. and Welling, M. Auto-encoding variational bayes, 2013. URL http://arxiv.org/abs/1312.6114. cite arxiv:1312.6114.

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 *ICML*, 2019.

Kober, J. and Peters, J. Learning motor primitives for robotics. In *ICRA*, 2009.

Konidaris, G. and Barto, A. Skill chaining: Skill discovery in continuous domains. In *the Multidisciplinary Symposium on Reinforcement Learning, Montreal, Canada*, 2009.

Konidaris, G., Kuindersma, S., Grupen, R., and Barto, A. Robot learning from demonstration by constructing skill trees. *IJRR*, 2012.

Kramer, G. Directed information for channels with feedback. 1998.

Krishnan, S., Fox, R., Stoica, I., and Goldberg, K. Ddco: Discovery of deep continuous options for robot learning from demonstrations. *arXiv preprint arXiv:1710.05421*, 2017.

Krishnan, S., Garg, A., Patil, S., Lea, C., Hager, G., Abbeel, P., and Goldberg, K. Transition state clustering: Unsupervised surgical trajectory segmentation for robot learning. In *RR*. 2018.

Kulkarni, T. D., Narasimhan, K. R., Saeedi, A., and Tenenbaum, J. B. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 3682–3690, Red Hook, NY, USA, 2016. Curran Associates Inc. ISBN 9781510838819.

Leslie Pack Kaelbling, T. L.-P. Learning composable models of parameterized skills. In *IEEE Conference on Robotics and Automation (ICRA)*, 2017. URL http://lis.csail.mit.edu/pubs/lpk/ICRA17.pdf.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Lioutikov, R., Neumann, G., Maeda, G., and Peters, J. Learning movement primitive libraries through probabilistic segmentation. *The International Journal of Robotics Research*, 36(8):879–894, 2017. doi: 10.1177/0278364917713116. URL https://doi.org/10.1177/0278364917713116.

Lioutikov, R., Maeda, G., Veiga, F., Kersting, K., and Peters, J. Learning attribute grammars for movement primitive sequencing. *The International Journal of Robotics Research*, 39(1):21–38, 2020. doi: 10.1177/0278364919868279. URL https://doi.org/10.1177/0278364919868279.

Mandlekar, A., Zhu, Y., Garg, A., Booher, J., Spero, M., Tung, A., Gao, J., Emmons, J., Gupta, A., Orbay, E., Savarese, S., and Fei-Fei, L. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. In *Conference on Robot Learning*, 2018.

Meier, F., Theodorou, E., Stulp, F., and Schaal, S. Movement segmentation using a primitive library. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

Muelling, K., Kober, J., and Peters, J. Learning table tennis with a mixture of motor primitives. In *2010 10th IEEE-RAS International Conference on Humanoid Robots*, pp. 411–416, Dec 2010. doi: 10.1109/ICHR.2010.5686298.

Murali, A., Garg, A., Krishnan, S., Pokorny, F. T., Abbeel, P., Darrell, T., and Goldberg, K. Tsc-dl: Unsupervised trajectory segmentation of multi-modal surgical demonstrations with deep learning. In *ICRA*, 2016.

Mülling, K., Kober, J., Kroemer, O., and Peters, J. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013. doi: 10.1177/0278364912472380. URL https://doi.org/10.1177/0278364912472380.

Neumann, G., Daniel, C., Paraschos, A., Kupcsik, A., and Peters, J. Learning modular policies for robotics. *Frontiers in computational neuroscience*, 8:62, 2014.

Niekum, S., Osentoski, S., Konidaris, G., and Barto, A. G. Learning and generalization of complex tasks from unstructured demonstrations. IEEE, 2012.

Peters, J., Kober, J., Mülling, K., Krämer, O., and Neumann, G. Towards robot skill learning: From simple skills to table tennis. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pp. 627–631. Springer, 2013.

Schaal, S. Learning from demonstration. In *Advances in Neural Information Processing Systems 9*, pp. 1040–1046, Cambridge, MA, 1997. MIT Press. URL http://www-clmc.usc.edu/publications/S/schaal-NIPS1997.pdf. clmc.

Shankar, T., Tulsiani, S., Pinto, L., and Gupta, A. Discovering motor programs by recomposing demonstrations. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgHY0NYwr.

Sharma, M., Sharma, A., Rhinehart, N., and Kitani, K. M. Directed-info GAIL: learning hierarchical policies from unsegmented demonstrations using directed information. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019. URL https://openreview.net/forum?id=BJeWUs05KQ.

Sharma, P., Mohan, L., Pinto, L., and Gupta, A. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *CoRL*, 2018.

Shiarlis, K., Wulfmeier, M., Salter, S., Whiteson, S., and Posner, I. Taco: Learning task decomposition via temporal alignment for control. *arXiv preprint arXiv:1803.01840*, 2018.

Smith, M., Hoof, H., and Pineau, J. An inference-based policy gradient method for learning options. In *ICML*, 2018.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 1999.

van der Maaten, L. and Hinton, G. Visualizing high-dimensional data using t-sne. 2008.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 1992.

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 *ICRA*, 2018.

Ziebart, B. D., Bagnell, J. A., and Dey, A. K. The principle of maximum causal entropy for estimating interacting processes. *IEEE Transactions on Information Theory*, 59(4):1966–1980, April 2013. ISSN 1557-9654. doi: 10.1109/TIT.2012.2234824.

6. Appendix

We provide additional details and insights into our approach below.

6.1. Derivation of Temporal Variational Inference:

While our main paper presents the specific temporal variational inference that we use to learn policies, the notion of temporal variational inference is more generally applicable to sequential data with hidden states (such as in HMMs).

We present a detailed derivation of this general temporal variational inference objective, and contrast it with standard variational inference below. We then provide a detailed derivation of the gradient of this objective in our case, explaining how dependencies on system dynamics may be factored out.

We begin by considering observed sequential data $x = \{x_t\}_{t=1}^{t}$ (in our case, this corresponds to state-action tuples, i.e. $\{x_t = (s_t, a_t)\}_{t=1}^{T}$), interacting with a sequence of unobserved latent variables $y = \{y_t\}_{t=1}^{T}$ (in our case, y_t is simply ζ_t). For these sequences of data, the true likelihood of observed data $\mathcal{L} = \mathbb{E}_{x\sim p(x)}\big[\log p(x)\big]$ is lower bounded by J, where:

$$J = \mathbb{E}_{x\sim p(x)}\big[\log p(x)\big] - D_{KL}\big[q(y|x)||p(y|x)\big]$$

where $q(y|x)$ is a variational approximation to the true conditional $p(y|x)$, and the lower bounded due to the non-negativity of KL divergence. Expanding the KL divergence term, and using the fact that $p(x, y) = p(y|x)p(x)$, we have:

$$J = \mathbb{E}_{x\sim p(x)}\big[\log p(x)\big] - \mathbb{E}_{x\sim p(x), y\sim q(y|x)}\Big[\log \frac{q(y|x)}{p(y|x)}\Big]$$
$$J = \mathbb{E}_{x\sim p(x), y\sim q(y|x)}\Big[\log p(x, y) - \log q(y|x)\Big]$$

Standard variational inference then decomposes the joint likelihood $p(x, y)$ into a "decoder" $p(x|y)$ and a prior $p(y)$:

$$J_{\text{StandardVI}} = \mathbb{E}_{x\sim p(x), y\sim q(y|x)}\Big[\log p(x|y) + \log p(y) - \log q(y|x)\Big]$$

Given the sequential nature of x and y, inferring the conditional $p(x|y)$ does not provide a useful insight into how the *sequence* of x and y will evolve, and is often difficult to learn. In contrast, our temporal variational inference makes use of the following decomposition of joint likelihood $p(x, y)$:

$$p(x, y) = \underbrace{\prod_{t=1}^{T} p(x_t|x_{1:t-1}, y_{1:t-1})}_{= p(x||y)} \underbrace{\prod_{t=1}^{T} p(y_t|x_{1:t}, y_{1:t-1})}_{= p(y||x)}$$

$p(x||y)$ and $p(y||x)$ are *causally conditioned distributions*, i.e. they only depend on information available until the current timestep. $p(x||y)$ and $p(y||x)$ are formally defined as $\prod_{t=1}^{T} p(x_t|x_{1:t-1}, y_{1:t-1})$ and $\prod_{t=1}^{T} p(y_t|x_{1:t}, y_{1:t-1})$ respectively (Ziebart et al., 2013). We can plug this decomposition into the objective J to give us the temporal variational inference objective:

$$\begin{aligned}
J_{\text{TemporalVI}} &= \mathbb{E}_{x\sim p(x), y\sim q(y|x)}\Big[\log p(x||y) + \log p(y||x) - \log q(y|x)\Big] \\
&= \mathbb{E}_{x\sim p(x), y\sim q(y|x)}\Big[\log \prod_{t=1}^{T} p(x_t|x_{1:t-1}, y_{1:t-1}) + \log \prod_{t=1}^{T} p(y_t|x_{1:t}, y_{1:t-1}) - \log q(y|x)\Big] \\
&= \mathbb{E}_{x\sim p(x), y\sim q(y|x)}\Big[\sum_{t=1}^{T} \log p(x_t|x_{1:t-1}, y_{1:t-1}) + \sum_{t=1}^{T} \log p(y_t|x_{1:t}, y_{1:t-1}) - \log q(y|x)\Big]
\end{aligned}$$

6.2. Derivation of Gradient Update:

Now that we have a better understanding of the origin of the temporal variational inference objective, we present a derivation of the gradient to this objective in our case. In our option learning setting, the TVIobjective is:

$$J_{\text{TemporalVI}} = \mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\sum_{t=1}^{T}\big\{\log\eta(\zeta_t|s_{1:t},a_{1:t-1},\zeta_{1:t-1}) + \log\pi(a_t|s_{1:t},a_{1:t-1},\zeta_{1:t}) + \log p(s_{t+1}|s_t,a_t)\big\} + \log p(s_1) - \log q(\zeta|\tau)\Big]$$

Assuming distributions π, η, and q are parameterized by θ, ϕ, and ω respectively, the gradient of this objective is:

$$\nabla_{\theta,\phi,\omega} J_{\text{TemporalVI}} = \nabla_{\theta,\phi,\omega}\,\mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\sum_{t=1}^{T}\big\{\log\eta(\zeta_t|s_{1:t},a_{1:t-1},\zeta_{1:t-1}) + \log\pi(a_t|s_{1:t},a_{1:t-1},\zeta_{1:t}) + \log p(s_{t+1}|s_t,a_t)\big\} + \log p(s_1) - \log q(\zeta|\tau)\Big]$$

Note that the system dynamics $p(s_{t+1}|s_t, a_t)$ and the initial state distribution $p(s_1)$ are both independent of the parameterization of networks θ, ϕ, and ω. The gradient of the objective with respect to θ, ϕ, and ω can be separated into additive terms that depend on the dynamics and initial state distributions, and terms that depend on networks π, η, and q. The expectation of the dynamics and initial state distribution terms are constant, and their gradient hence vanishes:

$$\nabla_{\theta,\phi,\omega} J_{\text{TemporalVI}} = \nabla_{\theta,\phi,\omega}\ \underbrace{\mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\sum_{t=1}^{T}\log p(s_{t+1}|s_t,a_t) + \log p(s_1)\Big]}_{\text{Constant}} = 0$$

The gradient update of temporal variational inference hence doesn't depend on the dynamics and the initial state distribution, leading to the following gradient update, as presented in the main paper in equation 3:

$$\nabla_{\theta,\phi,\omega} J = \nabla_{\theta,\phi,\omega}\mathbb{E}_{\tau\sim\mathcal{D},\zeta\sim q(\zeta|\tau)}\Big[\sum_t \log\pi(a_t|s_{1:t},a_{1:t-1},\zeta_{1:t}) + \sum_t \log\eta(b_t,z_t|s_{1:t},a_{1:t-1},\zeta_{1:t-1}) - \log q(\zeta|\tau)\Big]$$

6.3. Implementation Details:

We make note of several implementation details below, such as network architectures and hyperparameter settings.

6.3.1. Network Architecture:

We describe the specific architectures of each of the networks q, π, and η in our approach. The base architecture for each of these three networks is an 8 layer LSTM with 128 hidden units in each layer. We found that an 8 layer LSTM was sufficiently expressive for represent the distributions q, π, and η.

The space of predictions for q and η are the binary termination variables b_t at every timestep t, and the continuous parameterization of options z. q and η are thus implemented with two heads on top of the final LSTM layer.

The first head predicts termination probabilities (from which termination variables b_t are sampled), and consists of a linear layer of output size 2 followed by a softmax layer to predict probabilities.

The second head of the network predicts the latent z parameterization at each timestep. It consists of two separate linear layers above the final LSTM layer, that predict the mean and variance of a Gaussian distribution respectively. The mean predictions do not use an activation layer. The variance predictions employ a SoftPlus activation function to predict positive variances. The dimensionality of latent z's is 64 across all three datasets and across all networks.

The prediction space for π is the continuous low-level actions (i.e. joint velocities). Similar to the z prediction, this is implemented by with 2 separate linear layers to predict the mean and variances of a Gaussian distribution, from which actions are drawn. As above, variances use a SoftPlus activation, while mean predictions are done without an activation. The dimensions of the action space are 16 for the MIME dataset, 8 for the Roboturk dataset, and 66 for the Mocap dataset.

6.3.2. HYPERPARAMETERS:

We provide a list of the hyperparameters and their values, and other training details used in our training procedure.

1. Optimizer: We use the Adam optimizer (Kingma & Ba, 2014) to train all networks in our model.
2. Learning Rate: We use a learning rate of 10^{-4} for our optimizer, as is standard.
3. Epsilon: The exploration parameter ϵ is used in our training procedure to both scale perturbation of latent z's sampled from our model, as well as to explore different latent b's in an epsilon-greedy fashion. We use an initial ϵ value of 0.3, and linearly decay this value to 0.05 over 30 epochs of training, and found this works well. We considered a range of $0.1 - 0.3$ for the initial value of epsilon, and a range of $0.05 - 0.1$ for the final epsilon.
4. Ornstein Uhlenbeck Noise Parameters: Our DDPG implementation for the RL training uses the Ornstein Uhlenbeck noise process, with parameters identical to those used in the DDPG paper (Lillicrap et al., 2015).
5. Loss Weights: In practice, the various terms in our objective are reweighted prior to gradient computation, to facilitate learning the desired behaviors and to prevent particular terms from dominating others.
   (a) Option likelihood weight: During initial phases of training, we reweight the option likelihood term $\sum_t \log \eta(\zeta_t | s_{1:t}, a_{1:t-1}, \zeta_{1:t-1})$ in our gradient update by a factor of 0.01, to prevent the variational network from getting inconsistent gradients from the randomly initialized η. Once the variational policy has been trained sufficiently, we set the weight of this option likelihood to 1.
   (b) KL Divergence weight: We reweight the KL divergence term, as done in the β-VAE paper (Higgins et al.), by a factor of 0.01.

6.3.3. RL DETAILS:

For our reinforcement learning experiments, we use variants of the following Robosuite (Mandlekar et al., 2018) environments to evaluate our approach:

- SawyerPickPlace - An environment where a sawyer robot grasps and places objects in specific positions in respective bins. We use 4 variants of this task, SawyerPickPlaceBread, SawyerPickPlaceCereal, SawyerPickPlaceCan, SawyerPickPlaceMilk, where the objective is to place the corresponding object into the correct bin.
- SawyerNutAssembly - An environment where a sawyer robot must pick a nut up and place it around an appropriately shaped peg. We use 2 variants of this task, SawyerNutAssemblySquare and SawyerNutAssemblyRound, where the shape of the nut and peg are varied.

The 3 baseline algorithms specified in the main paper and our approach all share the same policy architectures (i.e., an 8 layer LSTM with 128 hidden units) for both low-level policies (in all baselines and our approach) and high-level policies (our approach and the hierarchical RL baseline).

For these pick-place and nut-assembly tasks, the information available to the policies are the sequence of joint states of the robot, previous joint velocities executed, the `robot-state` provided by Robosuite (consists of sin and cos of the joint angles, gripper positions, etc.), and the `object-state` provided by Robosuite (consisting of absolute positions of the target objects, object positions relative to the robot end-effector, etc.). The output space for the policies is always the joint velocities (including the gripper).

6.3.4. DATASET DETAILS:

Regarding the CMU Mocap dataset, the data used in this project was obtained from mocap.cs.cmu.edu, the database was created with funding from NSF EIA-0196217.