TACO: Learning Task Decomposition via Temporal Alignment for Control

Kyriacos Shiarlis [1]   **Markus Wulfmeier** [2]   **Sasha Salter** [2]   **Shimon Whiteson** [3]   **Ingmar Posner** [2]

Abstract

Many advanced *Learning from Demonstration* (LfD) methods consider the decomposition of complex, real-world tasks into simpler sub-tasks. By reusing the corresponding sub-policies within and between tasks, they provide training data for each policy from different high-level tasks and compose them to perform novel ones. However, most existing approaches to modular LfD focus either on learning a single high-level task or depend on domain knowledge and temporal segmentation. By contrast, we propose a weakly supervised, domain-agnostic approach based on *task sketches*, which include only the sequence of sub-tasks performed in each demonstration. Our approach simultaneously aligns the sketches with the observed demonstrations and learns the required sub-policies, which improves generalisation in comparison to separate optimisation procedures. We evaluate the approach on multiple domains, including a simulated 3D robot arm control task using purely image-based observations. The approach performs commensurately with fully supervised approaches, while requiring significantly less annotation effort, and significantly outperforms methods which separate segmentation and imitation.

1. Introduction

Learning from demonstration (LfD) represents a popular paradigm to teaching complex behaviours to robots and virtual agents through demonstrations, without the need for explicit programming or other description of a task, such as a cost function (Argall et al., 2009). The benefits of LfD over manual task definitions are numerous. First, some behaviours are difficult to program or manually encode, but can be easily demonstrated. Second, while programming behaviours requires expert knowledge of the application platform, LfD requires only the ability to control the agent.

However, complex real world tasks pose a severe challenge for simple LfD approaches such as *behavioural cloning* (BC). While a complex task can often be broken down into simpler sub-tasks, the algorithm itself lacks the means to discover the decomposition and must instead learn a monolithic policy for the whole task. The resulting policies are not only more complex but also less reusable. For example, a cube stacking task can be broken down into sub-tasks for approaching a cube, grasping it, moving to a location, and placing it onto an existing stack. In particular, sub-policies for approaching and grasping the cube can be reused in other manipulation routines such as rotating or throwing the cube.

Modular LfD addresses this challenge by modelling the task as a composition of sub-tasks for which reusable sub-policies (modules) are learned. These sub-policies are often easier to learn and can be composed in different ways to execute new tasks, enabling zero-shot imitation.

One approach to modular LfD is to provide the learner with additional information about the demonstrations. This comes in many forms, e.g., manual segmentation of demonstrations (Hovland et al., 1996), interactive feedback during learning (Niekum et al., 2015), or prior knowledge about the task. Such domain knowledge may come in the form of motion primitives, hard-coded parametric motion models (Ekvall & Kragic, 2006; Schaal, 2006) and problem-specific state modelling (Abdo et al., 2013; Konidaris et al., 2012). An additional benefit is that since these methods ground demonstrations on human-defined sub-tasks, the learned policies are interpretable. They are, however, labour intensive, perform sub-optimally if the underlying assumptions are incorrect, and require domain knowledge.

In this work, we consider a general, weakly supervised, modular LfD setting where demonstrations are augmented only with a *task sketch*. This sketch describes the sequence of *sub-tasks* that occur within the demonstration, without additional information on their alignment (see Figure 1).

Drawing inspiration from speech recognition (Graves et al., 2006) as well as modular reinforcement learning (Andreas et al., 2017), we introduce *temporal alignment for control* (TACO)[1], an efficient, domain agnostic algorithm that learns modular and grounded policies from high level task descriptions, while relying purely on weak supervision.

---

[1]Informatics Institute, University of Amsterdam, Netherlands
[2]Department of Engineering Science, University of Oxford, United Kingdom [3]Department of Computer Science, University of Oxford, United Kingdom. Correspondence to: Kyriacos Shiarlis <k.c.shiarlis@uva.nl>.

---

[1]The TACO algorithm and experiments will soon be available at https://sites.google.com/view/taco-ml.

Instead of considering the alignment of a demonstration with the task sketch and the learning of associated sub-policies as two separate processes, in TACO the imitation learning stage affects the alignment and vice-versa by maximising the joint likelihood of the observed sketch and the observed action sequence given the states. TACO learns one sub-policy for each sub-task present in the data and extends each sub-policy's action space to enable self-termination. At test time, the agent is presented with new, potentially unseen, and often longer sketches, which it executes by composing the required sub-policies. In addition to simplifying the imitation of complex tasks, the approach enables zero-shot imitation given only a sketch.

We evaluate the performance of TACO on four domains of varying complexity. First, we consider two toy domains, a 2D navigation task and the Craft domain proposed by Andreas et al. (2017). Finally, we consider the scenario of controlling a simulated robot arm to use a number pad and extend the task to use only image-based observations. We demonstrate that, in all domains, policies trained using TACO are capable of matching the performance of policies trained using a fully supervised method, where the segmentation of the demonstration is provided, at a small fraction of the labelling cost. At the same time, the approach significantly outperforms our baselines which separate the optimisation processes for segmentation and imitation.

2. Related Work

Learning from demonstration encompasses a wide range of techniques that focus on learning to solve tasks based on (human) expert demonstrations (Argall et al., 2009). The fields of modular and hierarchical LfD aim to extract reusable policy primitives from complex demonstrations to increase data efficiency and transfer knowledge between tasks.

In robotics, sub-policies can be modelled as *motion primitives* (Schaal, 2006) which build the foundation for various works on modular LfD e.g. (Niekum et al., 2012; Manschitz et al., 2014; Pastor et al., 2009). In this context, most similar to the ideas underlying our approach is recent work based on *skill trees* (Konidaris et al., 2012) and semantically grounded finite representations (Niekum et al., 2015). However, these approaches consider separate segmentation of the trajectories and fitting of primitives and imposes stronger constraints on the type of the controllers. In contrast, our work addresses segmenting the demonstrations and learning the policies in one combined process.

Interleaving the two processes has been shown to provide better segmentations and policies in recent work (Lioutikov et al., 2017). The approach however considers learning via policy embeddings in task space building on probabilistic motion primitives (Paraschos et al., 2013), which restricts the approach with respect to the types of tasks and observations. The method presented in this paper is less constrained and can handle arbitrary differentiable function approximator for the control policies.

Recent work on hierarchical LfD transfers concepts from the *options* framework (Sutton et al., 1999), which models low-level policies as actions for a meta controller, to LfD (Fox et al., 2017; Krishnan et al., 2017; Henderson et al., 2017). Generally, options serve as tools for dividing a complex task into multiple sub-policies specialised for regions in the state space. TACO differs from option discovery frameworks (Fox et al., 2017; Krishnan et al., 2017) during both training and inference by replacing the functionality of the meta-controller with weak supervision in the form of a sequence of symbols. Weak supervision constrains the learned policies to follow the description in the task sketch. This prevents degenerate cases including high-frequency switching between policies common with options (Krishnan et al., 2017) as well as the potential collapse of the meta-controller to apply only a single option. Furthermore, by applying task sketches at test time, we enable the composition of sub-policies in unseen and longer sequences for zero-shot imitation.

Recent work on *modular reinforcement learning* (RL) (Andreas et al., 2017), introduces the notion of sketches as additional information representing the decomposition of tasks. Similar to our work, Andreas et al. (2017) assume that complex tasks can be broken down into sub-tasks. Our approach exploits a similar modular structure but utilises an imitation based objective that addresses the problem of aligning sequences of different lengths.

A common approach to sequence alignment in speech recognition is *connectionist temporal classification* (CTC) (Graves et al., 2006). Previous extensions of CTC have been proposed to increase its flexibility by reducing the assumptions underlying the framework (Graves, 2012) and exploiting structure in the input space (Huang et al., 2016). In this paper, we extend CTC by combining sequence alignment and behavioural cloning.

3. Preliminaries

In this section, we introduce the required concepts and methods for the derivation of TACO.

3.1. Behavioural Cloning

Behavioural cloning (BC) models LfD as a supervised learning problem, by optimising a policy π to maximise the likelihood of the training dataset $\mathcal{D} = \{\rho_1, \rho_2, ..., \rho_M\}$, where $\rho = ((s_1, a_1), (s_2, a_2), ..., (s_T, a_T))$ is a state-action demonstration trajectory of T pairs of states $s \in \mathcal{S}$ and actions $a \in \mathcal{A}$. Let $\pi_\theta(a|s)$ be the probability of taking action a in state s as modelled by a policy π_θ parameterised by θ. BC performs the following optimisation:

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

One drawback of BC is its susceptibility to *covariate shift*, which occurs when small errors during testing cause the

agent to drift away from states it encountered during training, yielding poor performance. One way to overcome this problem is using *disturbances for augmenting robot trajectories* (DART) (Laskey et al., 2017), which introduces noise in the data collection process, allowing the agent to learn actions that can recover from errors. In this paper, unless stated otherwise, we use a training approach based on DART.

The standard formulations of BC and DART, which learn only one policy per task, lack two important properties. The first is modularity: the demonstrated behaviour can have a hierarchical structure that decomposes into modules, or sub-policies. The second is reuse: the modules can be composed in various ways to perform different tasks.

3.2. Modular Learning from Demonstration

Modular LfD introduces modularity and reuse to the LfD problem. A schematic is shown in Figure 1. To render the policies reusable, it assumes that any task can be solved by multiple sub-policies, each of which operates in an augmented action space $\mathcal{A}^+$ that includes a STOP action (i.e., $\mathcal{A}^+ := \mathcal{A} \cup a_{STOP}$) that does not have to be observed in the demonstration. It also assumes that more than one sub-policy may be present within a demonstration. In our formulation, extra information is provided in the form of a *task sketch* $\tau = (b_1, b_2, \ldots, b_L)$, with $L \leq T$, and $b_l \in \mathcal{B}$, where $\mathcal{B} = \{1, 2, \ldots, K\}$, is a dictionary of sub-tasks. The sketch indicates which sub-tasks are active in a trajectory.

Although the a_{STOP} action is never observed, it can be inferred from the data. If the demonstration contains a simple task then a_{STOP} is called only at the end of the demonstration. If $L = T$, then we know which policy from $\mathcal{B}$ is active at each time-step. i.e., a_{STOP} for each policy is called as soon as the active policy changes within the demonstration. If all extra information is available, we can perform behavioural cloning, with two differences. First, maximising the likelihood takes place assuming an action-augmented policy $\pi(a^+|s)$. Second, we learn $|\mathcal{B}| = K$ modular policies π_{θ_k} from K datasets $\mathcal{D}_k$ containing trajectories ρ_k as segmented based on τ:

$$\theta^*_{k=1,\ldots,K} = \underset{\theta_k}{\operatorname{argmax}} \, \mathbb{E}_{\rho_k}[\sum_{t=1}^{T_\rho} \log \pi_{\theta_k}(a_t^+|s_t)]. \quad (2)$$

e

However, the fully supervised approach is labour intensive as each trajectory must be manually segmented into sub-policies. In this paper, we consider cases where $L << T$ and τ contains only the sequence of active sub-tasks in the order they occur, without duplicates. Inferring when a_{STOP} occurs is therefore more challenging as τ and ρ operate at different timescales and must first be aligned.
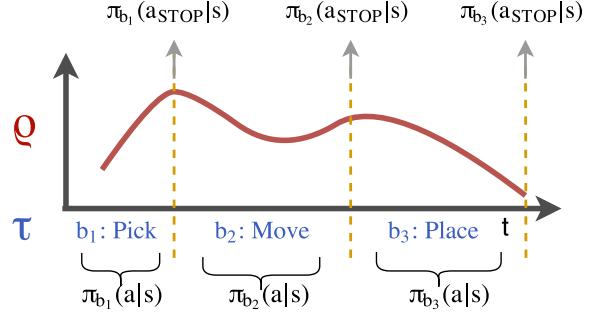

Figure 1. Problem setting: The trajectory ρ (red) is augmented by a task sketch τ (blue). The two sequences operate at different timescales. The whole trajectory is aligned (manually or automatically) and segmented into three parts. From this alignment three separate policies are learned. The unobserved a_{STOP} action for each policy is inferred to occur at the point where the policies switch from one to the other.

3.3. Sequence Alignment

Since $L << T$, we cannot independently maximise the likelihood of active sub-tasks in τ for every time step in ρ. The problem of aligning sequences of different lengths, a common challenge in speech and action recognition, is often addressed via *connectionist temporal classification* (CTC) (Graves et al., 2006). Here, we review a variant of CTC adapted to the notation introduced so far, which does not include gaps between the predictions, based on the assumption that sub-tasks occur in sequence without pause.

A *path* $\zeta = (\zeta_1, \zeta_2, ..., \zeta_T)$ is a sequence of sub-tasks of the same length as the input sequence ρ, describing the active sub-task ζ_t from the dictionary $\mathcal{B}$ at every time-step. The set of all possible paths $\mathcal{Z}_{T,\tau}$ for a task sketch τ is the set of paths of length T that are equal to τ after removing all adjacent duplicates. For example, after removing adjacent duplicates, the path $\zeta = (b_1, b_1, b_2, b_3, b_3, b_3)$ equals the sketch $\tau = (b_1, b_2, b_3)$. The CTC objective maximises the probability of the sequence τ given the input sequence ρ:

$$\psi^* = \underset{\psi}{\operatorname{argmax}} \, \mathbb{E}_{(\rho,\tau)}[p_\psi(\tau|\rho)] \quad (3)$$

$$= \underset{\psi}{\operatorname{argmax}} \, \mathbb{E}_{(\rho,\tau)}\Big[\sum_{\zeta \in \mathcal{Z}_{T,\tau}} p_\psi(\zeta|\rho)\Big] \quad (4)$$

$$= \underset{\psi}{\operatorname{argmax}} \, \mathbb{E}_{(\rho,\tau)}\Big[\sum_{\zeta \in \mathcal{Z}_{T,\tau}} \prod_{t=1}^{T} p_\psi(\zeta_t|\rho)\Big] \quad (5)$$

where $p_\psi(\zeta_t|\rho)$ is commonly represented by a neural network parameterised by ψ that outputs the probability of each sub-task in $\mathcal{B}$. While naively computing (3) is infeasible for longer sequences, dynamic programming provides a tractable solution. Let $\mathcal{Z}_{t,\tau_{1:l}}$ be the set that includes paths $\zeta_{1:t}$ of length t corresponding to task sketches $\tau_{1:l}$ of length l, and $\alpha_t(l) = \sum_{\zeta_{1:t} \in \mathcal{Z}_{t,\tau_{1:l}}} p(\zeta|\rho)$ be the probability of being in task b_l at time-step t in the graph in Figure 2a. The

probability of a task sketch given the input sequence $p(\tau|\rho)$ is equal to $\alpha_T(L)$.

We can recursively compute $\alpha_t(l)$ based on $\alpha_{t-1}(l)$, $\alpha_{t-1}(l-1)$, and the probability of the current sub-task $p(\zeta_t|\rho_t)$. As τ begins with a specific sub-task, the initial α's are deterministic and the probability of starting in the corresponding policy b_1 at time-step $t = 1$ is 1. Figure 2a depicts the recursive computation of the forward terms which is mathematically summarised as:

$$\alpha_t(l) = p(b_l|\rho_t)[\alpha_{t-1}(l-1) + \alpha_{t-1}(l))], \quad (6)$$

$$\alpha_1(l) = \begin{cases} 1, & \text{if } l = 1, \\ 0, & \text{otherwise.} \end{cases} \quad (7)$$

Based on the recursive computation of the CTC objective in (6) and any automatic differentiation framework, we can optimise our model. For the manual derivation of the gradients and CTC backwards variables, please see the work of Graves et al. (2006).

4. Methods

This section describes two ways to apply insights from CTC to address modular LfD. We first describe a naive adaptation which performs modular LfD with arbitrary differentiable architectures and discuss its drawbacks. We then introduce TACO, which simultaneously optimises the alignment between trajectory ρ and task sequence τ and learns the underlying policies.

4.1. Naively Adapted CTC

A naive modular LfD algorithm can first align the state-action trajectories with the prescribed sketches via CTC to derive the required datasets $\mathcal{D}_k$ and then learn K modular policies with behavioural cloning as in (2), an approach we denote as CTC-BC.

However, this approach fundamentally differs from the regular application of CTC. At inference time, CTC is typically given a new trajectory ρ and computes the most likely sketch assignment τ. In contrast, we use CTC to align ρ with τ and subsequently train the sub-policies via BC while discarding the CTC based controller. This alignment for BC is derived via maximisation over $\alpha_t(l)$, as computed in (6), at each time-step, leading to the most likely path through the sequence. Following the determination of sketch-trajectory alignment, the sub-policies are optimised using (2).

While this approach only trains sub-policies via a single alignment based on the argmax of the forward variables α for every time-step in (6), we can account for the probabilistic assignment of active sub-policies by utilising a weighting based on the forward variables. However, as no aligned targets exist for the stop actions, they have to be derived based on the relations of the normalised α's between consecutive time-steps. A derivation of this α-weighted version of CTC-BC can be found in the Appendix. In our experiments,

we consider the direct, single alignment resulting from taking the argmax as well as the full optimisation of the joint probabilities via TACO.

A crucial drawback of CTC-BC is the independent computation of the optimisation for alignment and for imitation. The alignment affects the policy optimisation as it is performed in a later step but not vice-versa. The introduction of TACO in the next section addresses this shortcoming.

4.2. Temporal Alignment for Control (TACO)

Aligning the sequences ρ and τ via CTC treats the index for active sub-policies as pure symbols and fails to exploit the fact that we then need to learn the respective sub-policies via BC. Consequently, what CTC determines to be a good alignment might result in a badly conditioned optimisation problem for the sub-policies, converging to a local minimum and demonstrating degraded performance once deployed.

In this section, we propose *Temporal Alignment for Control* (TACO), in which the alignment is influenced by the performance of the sub-policies and addressed within a single optimisation procedure. Concretely, instead of maximising Equation (3) followed by Equation (2), we seek to maximise the joint log likelihood of the task sequence and the actions conditioned on the states:

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

where $p(\zeta|\mathbf{s}_\rho)$ is the product of the stop, a_{STOP}, and non-stop, $\bar{a}_{STOP}$, probabilities associated with any given path. The first term in (8) is similar to the corresponding term in (5) but now depends only on states. Every possible alignment ζ dictates which data within the sequence ρ is associated with which sub-policy π_θ, which is the second term in (8) and corresponds to the BC objective. Maximising thus performs simultaneous alignment of τ and ρ and learns the associated policies for each sub-task.

As with CTC, the sketch length is expected to be shorter than the trajectory length, $L \ll T$, which for longer trajectories renders the computation of all paths ζ in $\mathcal{Z}_{T,\tau}$ intractable. However, the summation over paths can be performed via dynamic programming with a forward-backward procedure similar to that of CTC. Using consistent notation, the likelihood of a being at sub-task l at time t can be formulated in terms of forward variables:

$$\alpha_t(l) := \sum_{\zeta_{1:t} \in \mathcal{Z}_{t,\tau_{1:l}}} p(\zeta|\mathbf{s}_\rho) \prod_{t'=1}^{t} \pi_{\theta_{\zeta_{t'}}}(a_{t'}|s_{t'}). \quad (9)$$

Here $\tau_{1:l}$ denotes the part of the sub-task sequence until l.

Even though not explicitly modelling the probability of a certain sub-task given the states and actions ($p(b|s,a)$), extending the policies with the stop action a_{STOP} enables switching from one sub-task in the sketch to the next. This
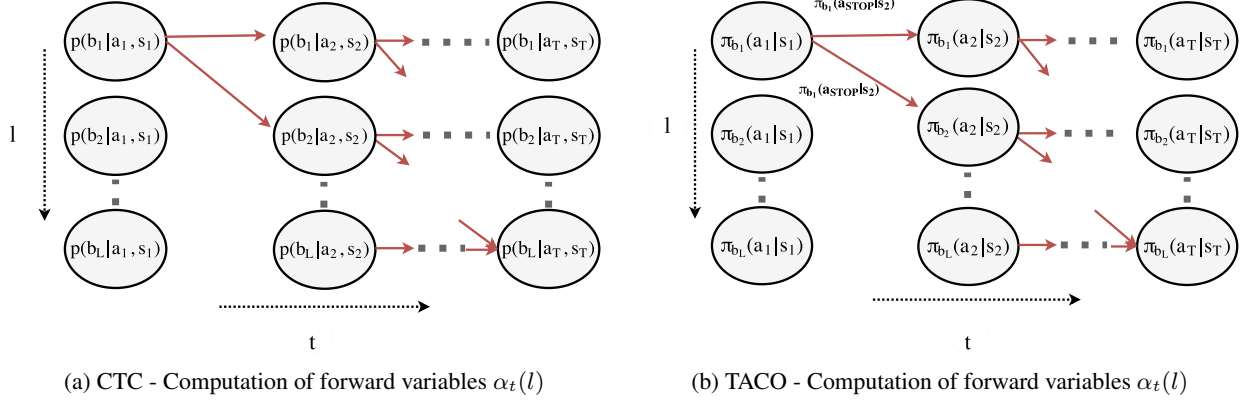
(a) CTC - Computation of forward variables $\alpha_t(l)$

(b) TACO - Computation of forward variables $\alpha_t(l)$

Figure 2. Visualisation of the forward recursion for all paths ζ corresponding to the sketch τ. The horizontal axis denotes time, while the vertical denotes the task sequence. While a node at l, t in CTC is only weighted by the meta-controller probability $p(b_l|s_t, a_t)$, nodes and edges are weighted in TACO respectively via the sub-policies $\pi_l(a_t|s_t)$ and $\pi_l(a_{STOP}|s_t)$

in turn allows computation of the probability of being in a certain sub-task b_l of the sketch, at time t. At time $t = 1$ we know that $\zeta_1 = \tau_1$, i.e., we always necessarily begin with the first sub-policy described in the sketch.

$$\alpha_1(l) = \begin{cases} \pi_{\theta b_1}(a_1|s_1), & \text{if } l = 1, \\ 0, & \text{otherwise.} \end{cases} \tag{10}$$

Subsequently, a certain sub-policy can only be reached by staying in the same sub-policy or stopping the previous sub-policy in the sketch using the a_{STOP} action:

$$\alpha_t(l) = \pi_{\theta b_l}(a_t|s_t)\big[\alpha_{t-1}(l-1)\pi_{\theta b_{l-1}}(a_{STOP}|s_t) \tag{11}$$
$$+\alpha_{t-1}(l)(1 - \pi_{\theta b_l}(a_{STOP}|s_t))\big].$$

Performing this recursion until T yields the forward variables. A visualisation of the recursion is shown in Figure 2b. The forward variables at the end of this recursion determine the likelihood in (8):

$$\alpha_T(L) = p(\tau, \mathbf{a}_\rho|\mathbf{s}_\rho). \tag{12}$$

Since the computation is fully differentiable, the backward variables and subsequently the gradient of the likelihood with respect to the parameters θ_i for each policy can be computed efficiently by any auto-diff framework. However, as the forward recursion can lead to underflow, we employ the forward variable normalisation technique from Graves et al. (2006).

Intuitively, the probability for the stop actions of a policy at each time-step determines the weighting of data points (state-action pairs) for all sub-policies. If a sub-policy assigns low probability a specific data-point, e.g., if at similar states it has been optimised to fit different actions, the optimisation increases the probability of the preceding and succeeding policy in the sketch for that data point, effectively influencing the probabilistic alignment.

A potential pitfall of simultaneously optimising for alignment and control is the early collapse of the alignment objective to a single path (Figure 2b). This stops further exposure of the sub-policies to potential state-action pairs they would be able to fit well. To achieve sufficient exploration of different possible alignments, we use dropout (Srivastava et al., 2014). At every forward pass, different alignment paths are sampled, exposing the sub-policies to a wider range of data-points they could potentially fit, greatly improving performance.

5. Experiments

We evaluate TACO across four different domains with different continuous and discrete states and actions including image-based control of a 3D robot arm. Our experiments aim to answer the following questions.

- How does TACO perform across a range of different tasks? Can it be successfully applied to a range of architectures and input-output representations?
- How does TACO perform with respect to zero-shot imitation on sequences not included in the training set and task sketches of different length?
- How does TACO perform in relation to baselines including CTC-BC (Section 4.1) and the fully supervised approach with all trajectories segmented into sub-tasks?
- How does the dataset size influence the relative performance of TACO in comparison with our baselines?

The latter questions are investigated by introducing a set of baselines based on Sections 3.2 and 4.1.

- GT-BC, which uses ground-truth, segmented demonstrations and performs direct maximum likelihood training to optimise the sub-policies (Equation 2).

- CTC-BC, which uses both bidirectional *gated recurrent units* (Cho et al., 2014) [CTC-BC(GRU)] or *multi-layer perceptrons* [CTC-BC(MLP)] to perform CTC. In both cases, we apply MLPs for the sub-policies.

While the options framework presents a common approach to hierarchical LfD (Fox et al., 2017; Krishnan et al., 2017; Henderson et al., 2017), it predominantly neglects the possibility of additional control information to switch between different high-level tasks, as is given by task sketches. Given the resulting limitation of independent modelling of different high level tasks, these framework cannot efficiently model multiple task sketches. This will result in highly degraded performance when evaluation is based on multiple different tasks, like the ones given in our evaluation (in both regular and in particular zero shot scenarios). For this reason, the evaluation focuses on approaches that utilise the task sketches.

The main evaluation metric is the task accuracy, i.e., the ratio of full tasks completed to the total attempted. In addition, the sub-task accuracy is ported to provide more insight. Finally, Table 1 provides results comparing the sequence alignment accuracy. That is, the percentage of timesteps that the demonstrated trajectory was given a correct sub-task label when compared to the ground truth alignment.

We focus on testing in a zero-shot setting: the task sequences prescribed at test time are not in the training data and are longer than the training sketches. Note that in the non-zero-shot setting, while the tasks to be completed have been seen, the world parameters such as feature positions are randomised. Finally, in all our evaluations we vary the size of the training set to investigate how performance varies with respect to available data.
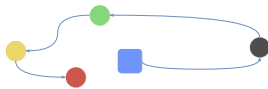
5.1. Nav-World Domain

Figure 3. The Nav-World. The agent (Blue) receives a route as a task sketch. In this case. $\tau = $ (Black, Green, Yellow, Red)

We present the Nav-World domain, depicted in Figure 3, as a simple 2D navigation task. The agent (Blue) operates in a 8-dimensional continuous state space with four destination points (Green, Red, Yellow, Black). The state space represents the (x, y) distance from each of the destination points. The action space is 2-dimensional and represents a velocity (v_x, v_y). At training time, the agent is presented with state-action trajectories ρ from a controller that visits L destinations in a certain sequence given by the task sketch, e.g. $\tau = $ (Black, Green, Yellow, Red). At the end of learning, the agent outputs four sub-policies $\pi(a^+|s)$ for reaching each destination. At test time, it is given a sketch τ_{test} of length L_{test} containing a sequence of destinations. The task

is considered successful if the agent visits all destinations in the correct order. During demonstrations and testing, the agent's location and the destination points are sampled from a Gaussian distribution centred at predefined locations.
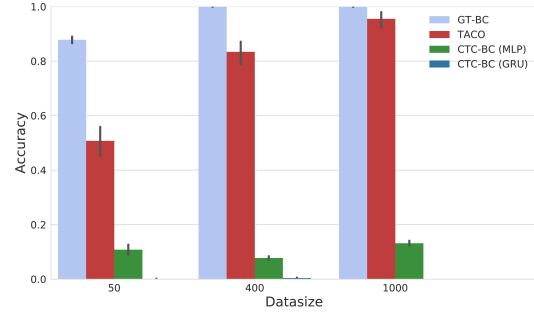

Figure 4. Nav-World results: Mean accuracy over 100 agents on 100 test tasks. Task length at test time is $L_{test} = 4$ and at training time is $L = 3$. TACO (red) approaches the performance of a fully supervised sequence (grey) given enough data.

In this domain, the dataset sizes are 50, 400, and 1000 demonstrations. The task length at training time is $L = 3$. We report the task success rate for unseen, longer tasks of length $L_{test} = 4$ (zero-shot setting). 100 agents are trained for each task and each algorithm, with the evaluation based on 100 testing tasks.

As displayed in the results in Figure 4, CTC-BC performs poorly in both settings, with the MLP architecture performing slightly better. However, CTC(MLP) provides accurate alignment, often reaching 90% overlap between the predicted sub-task sequence and ground truth (Table 1). Fitting on smaller datasets however strongly reduces the quality of the policies. As the states are similar before and after a policy switch but the actions are different (different policies), small mistakes in alignment cause multi-modality in the data distribution. This causes relatively low performance for BC with a mean squared error (MSE) objective. In contrast, TACO avoids this problem by probabilistically weighting all policies when fitting to each time-step. From Table 1 we can see that TACO's inductive bias for control allows it to achieve much better alignment accuracy than CTC-BC. These two factors bring about performance that approaches that of GT-BC for larger datasets, a consistent trend in all our experiments.

5.2. Craft Domain

Andreas et al. (2017) introduced the Craft Domain to demonstrate the value of weak supervision using policy sketches in the RL setting. In this domain, an agent is given hierarchical tasks and a sketch description of that task. The binary state space has 1076 dimensions and the action space enables discrete motions as well as actions to pick up and use objects. A typical example task is $\tau_{\text{make planks}} = $ (get(wood), use(workbench)). The tasks vary

Algorithm	Domain			
	Nav-World	Craft	Dial	Dial (Image)
TACO	**95.3**	95.6	**99.8**	**99.0**
CTC-BC (MLP)	89.0	41.4	90.1	84.6
CTC-BC (GRU)	80.0	57.1	93.4	48.8
GT-BC (aligned w. TACO)	94.6	**99.4**	89.0	98.2

Table 1. Alignment accuracy of each algorithm for all domains. TACO always outperforms CTC emphasising the importance of maximising the joint likelihood of task sequences and actions. The alignment for GT-BC was obtained by computing the argmax of the TACO forward variables on the policies learned using GT-BC.

from $L = 2$ to $L = 4$. The demonstrations are provided from a trained RL agent, which obtains near optimal performance. Performance is measured using the reward function defined in the original domain. We train agents for all baselines and deploy them on randomly sampled tasks from the same distribution seen during training.

Figure 5 shows the results, which are similar to those in Nav-World. CTC-BC fails to obtain substantial reward. However, unlike in Nav-World, CTC does not achieve good alignments with either architecture, as the abstract, binary state-action space makes it harder to detect distribution changes. Instead of concatenating state-action pairs, TACO learns a mapping from one to the other, making it easier to tell when the sub-policy changes. This in turn allows TACO to match or even surpass BC with sufficient data.
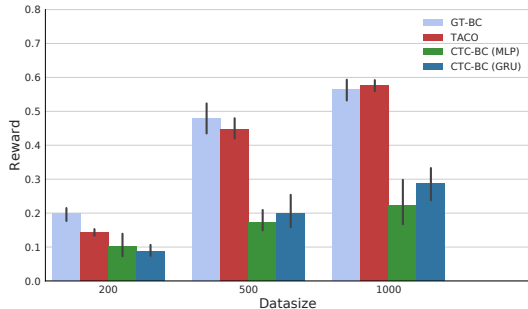

Figure 5. Craft results: Mean reward over 2000 tasks. The tasks are the same as during training but the environment instantiation is different. The RL agent used to derive the policies achieves a mean reward of 0.9. Taco (red) approaches the performance of a fully supervised method (grey) given enough data.

5.3. Dial Domain

Even though the Craft Domain is quite challenging, the lengths of the demonstrations and the resulting episodes are quite short, $T \leq 20$, making it easier to align ρ and τ. The final two experiments take place in a more realistic robotic manipulation domain. In the Dial Domain, a JACO 6 DoF manipulator simulated in MuJoCo (Todorov et al., 2012) interacts with a large dial-pad, as shown in Figure 6. The demonstrations contain state-action trajectories that describe the process by which a PIN is pressed, e.g., $\tau = (0, 5, 1, 6)$.

A task is considered successful if all the digits in the PIN are pressed in order. Demonstrations in this domain come from a PID controller that can move to predefined joint angles. We sub-sample the data from the simulator by 20, resulting in trajectories $T \approx 200$ for $L = 4$. We report task and alignment accuracy as before. We consider two variants of the Dial Domain, one with joint-angle based states and the other with images. The action space represents the torques for each joint of the JACO arm in both cases.
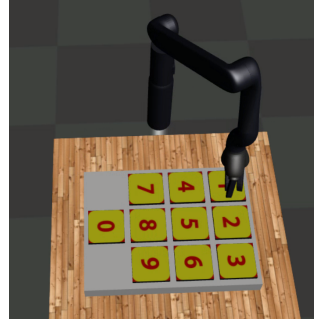

Figure 6. Dial Domain: A 6 DoF JACO arm must dial a PIN of arbitrary length.

5.3.1. JOINT SPACE DIAL DOMAIN

In the first variant, the state is manually constructed and 39-dimensional, containing the joint angles and the distance from each digit on the dial pad in three dimensions. If the locations of the numbers in the dial-pad remains the same, however, the problem can be solved only using joint angles. For this reason, during each demonstration we randomly swap the location of the numbers. We test the policies on the standard number formation displayed in Figure 6, which is never observed during training.

Figure 7 shows the results, which follow the same trend as in the other domains. CTC-BC fails to complete any tasks and is not capable of aligning the sequences. TACO's performance significantly increases with data size, achieving superior performance to GT-BC at sizes of 1000 and 1200 demonstrations. We believe is due to a regularising effect of TACO's optimisation procedure. To shed more light into this observation, we use a GT-BC policy and perform alignment with the TACO forward pass on 100 unseen trajectories (Table 1). The resulting alignment is lower than that of TACO, which suggests that GT-BC is more prone to overfitting.

To evaluate our methods in more complex zero-shot scenarios, we also measure the task accuracy over 100 tasks as L_{test} is increased, as shown in Figure 8. As expected, TACO's performance falls with increasing task length as the chance of failing at a single sub-task increases. The accuracy however decreases at a lower rate than that of the baseline.
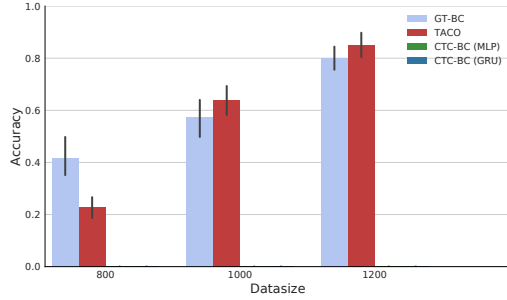
Figure 7. Joint Space Dial results: Task accuracy over 100 tasks of $L_{test} = 5$. During training $L = 4$. Evaluation is performed in an unseen configuration of the dial pad. Bars for CTC based methods do not appear as they did not finish any tasks.
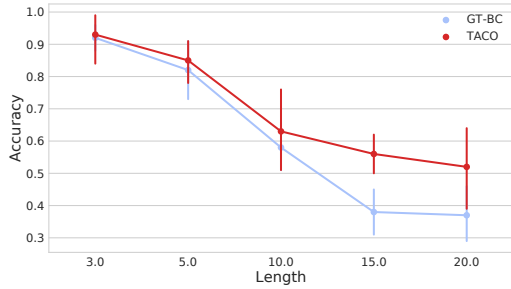

Figure 8. Task accuracy for increasing values of L_{test}. The accuracy is measured over 100 runs for TACO and GT-BC.

5.3.2. IMAGE SPACE DIAL DOMAIN

The image-based variant of the Dial Domain considers the same task as above, but with an image-based state representation. We use RGB images of size 112×112, which are passed through a simple convolutional architecture before splitting into individual policies. In this case, we do not randomise the digit positions but discard joint angles from the agent's state space, as those angles would allow an agent to derive an optimal policy without utilising the image. Details of the architectures used can be found in the Appendix.

Figure 9 details task and sub-task accuracy for this domain. We can see that TACO performs well in comparison with the baselines, despite the increased difficulty in the state representation.

6. Discussion

TACO requires only weak supervision for modular LfD while providing performance commensurate with fully supervised approaches on a range of tasks including zero-shot imitation scenarios.

To successfully complete a task, a policy must not only complete the sub-tasks but also terminate the active policy at the right moment. Both GT-BC and CTC-BC rely on a single alignment between sub-tasks and demonstrations, based on ground truth and an `argmax` alignment respectively. TACO's strength lies in optimising the sub-policies over a distribution instead of a point estimate of the alignment. Training policies over a distribution of alignments ex-
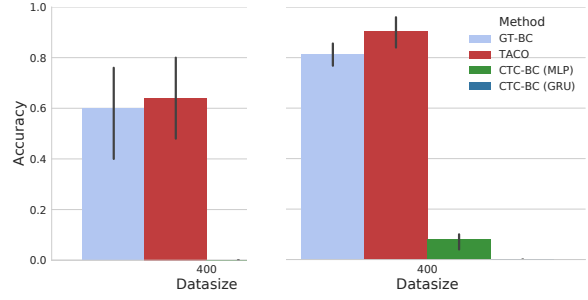
(a) Task Accuracy (b) Sub-Task Accuracy

Figure 9. Image-Space Dial: Accuracy over 25 test tasks. CTC-BC is unable to perform full task sequences and only solves a minor percentage of the sub-tasks, while both GT-BC and TACO are able to complete most sub-tasks.

poses the sub-policies to more data points, which induces a regularising effect. This view is supported by the alignment accuracy results of Table 1 in which we evaluate trained policies for alignment on unseen sequences (based on (10) and (11)). For larger datasets GT-BC achieves less accurate alignment than TACO (Table 1), which in turn suggests that it may be more prone to overfitting. Better alignment on test demonstrations is correlated with task accuracy. This suggests that the idea of integrating the optimisation objectives in TACO for sequence alignment as well as imitation learning could have applications in cases where the end objective is good alignment rather than control policies.

TACO has been effectively applied to the presented tasks and significantly reduces supervision efforts. However, the given sketches are highly structured and dissimilar to natural human communication. An interesting avenue for future work is the combination of the increased modularity of TACO with more flexible architectures that can handle natural language (Mei et al., 2016; Chaplot et al., 2017). In addition, future work aims at applications in more complex hierarchical tasks and on real robots.

While TACO reduces the annotation effort compared to temporally segmented trajectories, it relies on weak supervision via task sketches. Further work into relaxing the assumptions underlying the use of these sketches can aim at omitting the constraint on the order of the sub-tasks.

7. Conclusion

We presented TACO, a novel method to address modular learning from demonstration by incorporating weakly supervised information in the form of a task sketch, that provides a high-level description of sub-tasks in a demonstration trajectory. We evaluated TACO in four different domains consisting of continuous and discrete action and state spaces, including a domain with purely visual observations. With limited supervision, TACO performs commensurate to a fully supervised approach while significantly outperforming the straightforward adaptation of CTC for modular LfD in both control and alignment.

Acknowledgements

References

Abdo, N., Kretzschmar, H., Spinello, L., and Stachniss, C. Learning manipulation actions from a few demonstrations. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pp. 1268–1275. IEEE, 2013.

Andreas, J., Klein, D., and Levine, S. Modular multitask reinforcement learning with policy sketches. In *International Conference on Machine Learning*, pp. 166–175, 2017.

Argall, B. D., Chernova, S., Veloso, M., and Browning, B. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009.

Chaplot, D. S., Sathyendra, K. M., Pasumarthi, R. K., Rajagopal, D., and Salakhutdinov, R. Gated-attention architectures for task-oriented language grounding. *arXiv preprint arXiv:1706.07230*, 2017.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

Ekvall, S. and Kragic, D. Learning task models from multiple human demonstrations. In *Robot and Human Interactive Communication, 2006. ROMAN 2006. The 15th IEEE International Symposium on*, pp. 358–363. IEEE, 2006.

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

Graves, A. Sequence transduction with recurrent neural networks. *arXiv preprint arXiv:1211.3711*, 2012.

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.

Henderson, P., Chang, W.-D., Bacon, P.-L., Meger, D., Pineau, J., and Precup, D. OptionGAN: Learning Joint Reward-Policy Options using Generative Adversarial Inverse Reinforcement Learning. *ArXiv e-prints*, September 2017.

Hovland, G. E., Sikka, P., and McCarragher, B. J. Skill acquisition from human demonstration using a hidden markov model. In *Robotics and Automation, 1996. Proceedings., 1996 IEEE International Conference on*, volume 3, pp. 2706–2711. Ieee, 1996.

Huang, D.-A., Fei-Fei, L., and Niebles, J. C. Connectionist temporal modeling for weakly supervised action labeling. In *European Conference on Computer Vision*, pp. 137–153. Springer, 2016.

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. doi: 10.1177/0278364911428653. URL https://doi.org/10.1177/0278364911428653.

Krishnan, S., Fox, R., Stoica, I., and Goldberg, K. Ddco: Discovery of deep continuous options for robot learning from demonstrations. In *Conference on Robot Learning*, pp. 418–437, 2017.

Laskey, M., Lee, J., Fox, R., Dragan, A., and Goldberg, K. Dart: Noise injection for robust imitation learning. In *Conference on Robot Learning*, pp. 143–156, 2017.

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.

Manschitz, S., Kober, J., Gienger, M., and Peters, J. Learning to sequence movement primitives from demonstrations. In *Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on*, pp. 4414–4421. IEEE, 2014.

Mei, H., Bansal, M., and Walter, M. R. Listen, attend, and walk: Neural mapping of navigational instructions to action sequences. In *AAAI*, volume 1, pp. 2, 2016.

Niekum, S., Osentoski, S., Konidaris, G., and Barto, A. G. Learning and generalization of complex tasks from unstructured demonstrations. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pp. 5239–5246, Oct 2012. doi: 10.1109/IROS.2012.6386006.

Niekum, S., Osentoski, S., Konidaris, G., Chitta, S., Marthi, B., and Barto, A. G. Learning grounded finite-state representations from unstructured demonstrations. *The International Journal of Robotics Research*, 34(2):131–157, 2015.

Paraschos, A., Daniel, C., Peters, J. R., and Neumann, G. Probabilistic movement primitives. In *Advances in neural information processing systems*, pp. 2616–2624, 2013.

Pastor, P., Hoffmann, H., Asfour, T., and Schaal, S. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pp. 763–768, May 2009. doi: 10.1109/ROBOT.2009.5152385.

Schaal, S. Dynamic movement primitives-a framework for motor control in humans and humanoid robotics. In *Adaptive motion of animals and machines*, pp. 261–280. Springer, 2006.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Sutton, R. S., Precup, D., and Singh, S. Between mdps and semi-mdps: A framework for temporal abstraction in reinforcement learning. *Artificial intelligence*, 112(1-2): 181–211, 1999.

Todorov, E., Erez, T., and Tassa, Y. Mujoco: A physics engine for model-based control. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pp. 5026–5033. IEEE, 2012.