TACO: Learning Task Decomposition via Temporal Alignment for Control - Appendix

A. Technical Details

A.1. Implementation Details and Architecture

The policies for all tasks are modelled as MLPs unless images are used. In this case, convolutional layers, shared between all sub-policies, are used to extract state features. In continuous domains, action probabilities are modelled as $a \sim \mathcal{N}(\mu, \sigma = 1)$ where μ is the output of the MLP. In discrete domains, action probabilities are modelled using categorical distributions. We found that having separate networks for the domain actions and the a_{STOP} action, leads to better performance, especially for the Dial domain. See Table 1 for implementation details of the architectures used in each experiment where *Core* represents the convolutional architecture that learns a feature representation of the input space before feeding it into the stop and action policies.

For larger domains we found that dropout on the a_{STOP} policy greatly improved results. It serves as regulariser as well as to ensure optimising over a broad range of paths as various alignment paths are sampled when units in the network are dropped. For further performance, we exponentialy decrease the dropout rate. All models are implemented in TensorFlow (Abadi et al., 2016).

Experiment	State Dim	Core	Stop Policy	Action Policy	Output Dim
Nav World	8	-	FC [100]	FC [100]	2
Craft	1076	-	FC [400,100]	FC [400,100]	7
Dial	39	-	FC [300,200,100]	FC [300,200,100]	9
Dial (Images)	[112,112,3]	conv[10,5,3] kernel[5,5,3] stride[2,2,1]	FC [400,300]	FC [400,300]	9

Table 1. Specification of the architectures used in each experiment. For experiment *Dial (Images)* conv[], kernel[] and stride[] represent the number of channels, dimension of square kernels and 2D strides per convolutional layer respectively

A.2. Data Collection

As mentioned in the paper, data collection took place using DART (Laskey et al., 2017) in order to compensate for the issue of covariate shift and compounding errors during policy deployment. For the Dial domain we add noise to each joint, proportional to the maximum allowed torque, which was manually tuned. During demonstrations we add a varying degree of noise to better cover the state space resulting in more robust policies that are better able to handle suboptimality in the test domain.

B. Detailed Results

This section covers more detailed results for different domains and algorithms. For the NavWorld scnario, we report results on non-0-shot settings. For the Dial domain, we additionally address the sub-task accuracy i.e how many sub-tasks were completed out of the total attempted. Finally, we detail alignment accuracy values for the methods considered in the paper. The metric describes the percentage of correctly aligned sub-policies on a set of hold-out tasks.

B.1. NavWorld Domain

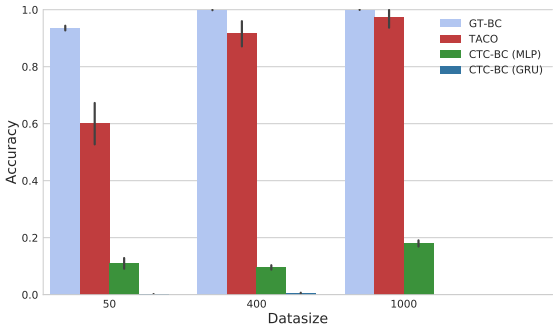

Figure 1. Task accuracy on NavWorld. $L = L_{test} = 3$.

As is seen in Figure 1, TACO not only vastly outperforms CTC-BC (MLP and GRU), both of which favour poorly in NavWorld, but as well asymptotically approaches the fully supervised GT-BC with increasing number of training trajectories.

B.2. Joint-State Dial Domain

The sub-task accuracy is displayed in Figure 2.

B.3. Sequence Alignment Accuracy

Alignment accuracy is measured as the % of agreement between the ground truth sub-task sequence of length T and the most likely sequence predicted by either of the three alignment architectures considered in the paper, TACO, CTC-BC (MLP), CTC-BC (GRU). The most likely sequence is given by taking the argmax of the forward variables at each timestep for either CTC or TACO. For GT-BC,

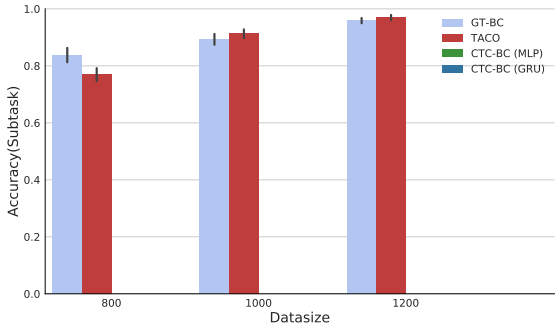

Figure 2. Sub-task accuracy for the Dial domain. We measure the number of sub-tasks succeeded over the total given, for 5 independent evaluations of 20 tasks of task length 5. Due to the complexity of the task and bad alignment performance, CTC based methods are unable to complete any sub-tasks.

we pass the learned policies through TACO alignment without learning. We compute the alignment accuracy for 100 hold-out test sequences, which come from the same distribution as the training data. The results across all the experiment domains are stated in Table 2.

	Domain			
Algorithm	Nav-World	Craft	Dial	Dial (Image)
TACO	**95.3**	95.6	**98.9**	**99.0**
CTC-BC (MLP)	89.0	41.4	31.4	84.6
CTC-BC (GRU)	80.0	57.1	28.6	48.8
GT-BC (aligned with TACO)	94.6	**99.4**	98.7	98.2

Table 2. 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.

C. CTC Probabilistic Sub-Policy Training

While the naive extension of CTC (Section 4.1) trains sub-policies based on the a single alignment by taking the argmax of the forward variables $\alpha_t(l)$ for every time-step, this paragraph addresses the possible probabilistic assignment of active sub-policies.

To obtain the probabilistic weighting based optimisation objective in Equation (1), we first define the probability distribution $p_t(l)$ at time-step t over all sub-policies l in a sketch based on the normalised CTC forward variables in Equation (2).

However, as the ground-truth targets in the trajectory ρ only exist for the regular actions, we first compute the stop action probability targets based on the CTC forward variables.

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

$$\text{where } p_t(l) = \frac{\alpha_t(l)}{\sum_{l_i=1}^{L}\alpha_t(l_i)}. \quad (2)$$

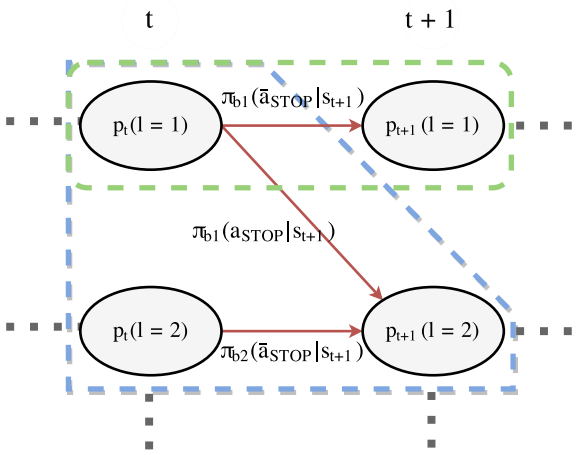

Figure 3. CTC-based computation of stop action targets. The green and blue areas respectively depict the relations for $l = 1$ in Equation 3 and $l > 1$ in Equation 4

To determine probabilistic targets for the stop actions, we associate the edges in Figure 3 between nodes of the same or subsequent sub-policies respectively with $\bar{a}_{stop}$ and a_{stop}. For $l = 1$, nodes depend only on a single relevant edge from the same sub-policy at the previous time-step, while for all nodes with $l > 1$ we take into account edges from the same sub-policy and the previous sub-policy at the previous time-step.

$$p_{t+1}(1) = p_t(1) \cdot \pi_1(\bar{a}_{stop}|s_{t+1})] \quad (3)$$

$$p_{t+1}(l+1) = p_t(l+1) \cdot \pi_{l+1}(\bar{a}_{stop}|s_{t+1}) + p_t(l) \cdot \pi_l(a_{stop}|s_{t+1}) \quad (4)$$

$$\pi_l(a_{stop}|s_t) = 1 - \pi_l(\bar{a}_{stop}|s_t) \quad (5)$$

Based on Equations (3), (4) and (5), we can derive the targets in Equation (6) for $t = 1, ..., T-1$. Starting with the targets for $l = 1$ we can compute the targets for $l+1$ based on the targets for l and the forward variables $\alpha_t(l)$.

$$\pi_l(\bar{a}_{stop}|s_{t+1}) \begin{cases} \frac{p_t(l)}{p_{t+1}(l)}, & \text{if } l = 1, \\ \frac{p_{t+1}(l+1)}{p_t(l+1)} - \frac{p_t(l)\cdot(1-\pi_l(\bar{a}_{stop}|s_{t+1}))}{p_t(l+1)} & \text{if } l > 1. \end{cases} \quad (6)$$

References

Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

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.