One-Shot Learning of Multi-Step Tasks from Observation via Activity Localization in Auxiliary Video

Wonjoon Goo and Scott Niekum
Department of Computer Science
University of Texas at Austin, Austin, TX 78712
{wonjoon, sniekum}@cs.utexas.edu

***Abstract*— Due to burdensome data requirements, learning from demonstration often falls short of its promise to allow users to quickly and naturally program robots. Demonstrations are inherently ambiguous and incomplete, making correct generalization to unseen situations difficult without a large number of demonstrations in varying conditions. By contrast, humans are often able to learn complex tasks from a single demonstration (typically observations without action labels) by leveraging context learned over a lifetime. Inspired by this capability, our goal is to enable robots to perform one-shot learning of multi-step tasks from observation by leveraging auxiliary video data as context. Our primary contribution is a novel system that achieves this goal by: (1) using a single user-segmented demonstration to define the primitive actions that comprise a task, (2) localizing additional examples of these actions in unsegmented auxiliary videos via a metalearning-based approach, (3) using these additional examples to learn a reward function for each action, and (4) performing reinforcement learning on top of the inferred reward functions to learn action policies that can be combined to accomplish the task. We empirically demonstrate that a robot can learn multi-step tasks more effectively when provided auxiliary video, and that performance greatly improves when localizing individual actions, compared to learning from unsegmented videos.**

I. Introduction

Learning from demonstration (LfD) [1] has emerged as a powerful way to quickly and naturally program robots to perform a wide variety of tasks. Unfortunately, demonstrations are inherently ambiguous and incomplete. Correct generalization to unseen situations is therefore difficult without a large number of demonstrations in varying conditions. This data requirement places a significant burden on end-users, often limiting the use of LfD to simple tasks.

By contrast, humans are often able to learn complex tasks from a single demonstration by leveraging context learned over a lifetime—for example, information about how objects work, episodic memories of similar situations, or an intuitive understanding of the intentions of the demonstrator. Similarly, robots increasingly have access to auxiliary sources of video data—for example, from prior experiences, curated datasets such as the Epic-Kitchen dataset [2], or less structured Youtube videos. In this work, we propose to leverage auxiliary video data as contextual information to help robots intelligently disambiguate and generalize a single demonstration of a multi-step task. Essentially, a single user-provided, segmented demonstration describes *what* activities to perform (as well as one example of how to perform them, grounded in the actual environment that the robot will act in), while the auxiliary video data provides additional examples of *how* each activity ought to be performed, allow the robot to learn to generalize without overfitting.

Although prior works have explored the use of video data in an LfD setting, in all instances that we are aware of these methods have only addressed single step tasks [3], [4] or have assumed well-aligned data with little variance [5]. However, many common robotics tasks, such as cooking and assembly, require multiple steps that may have different goals and involve different objects or features. Thus, as we show experimentally (and as other works have argued [6], [7]), learning a separate policy for each step of a task can lead to improved generalization.

Our primary contribution is a novel framework that can localize additional examples of each user-demonstrated action in unsegmented auxiliary videos, which are then used to aid learning. Specifically, we cast the problem of action localization as a single-shot activity recognition problem, in which we only have one example of each activity (from user-provided demonstration segments) and attempt to classify small sets of frames in each of the auxiliary videos as one (or none) of those activities.

We then use the segmented video clips in tandem with the original demonstration segments to learn to perform each step of the task separately. However, one significant difficulty in utilizing video data in an LfD setting is that there are typically no available action labels for the observations. This difficulty has motivated work in the learning from observation (LfO) setting. Observations without action labels are generally not sufficient for direct imitation learning, but have been used instead to help build dense reward signals [5], [3], learn object affordances [8], or resolve ambiguities of written instructions [9].

While there are many possible ways to use the segmented video data from our algorithms in an LfO setting, we focus on an inverse reinforcement learning setting. Specifically, for each subtask, the segmented video clips of that activity are used to perform reward function inference, followed by reinforcement learning on the inferred reward functions. The learned policies can then be sequentially executed to accomplish the task in novel situations. Figure 1 illustrates this full learning pipeline.

We first demonstrate the accuracy of our one-shot ac-

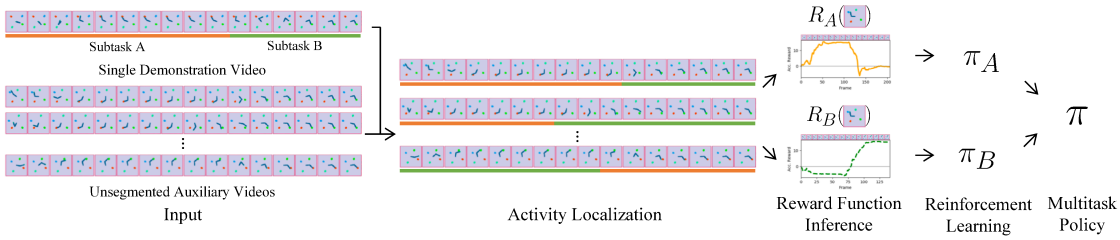


Fig. 1: The proposed one-shot learning from observation pipeline: (1) A single segmented demonstration defines subtasks; (2) additional clips of each subtask are then localized in unsegmented auxiliary videos; (3) these clips are used to infer a reward function for each subtask; (4) RL is used to learn subtask policies that can be combined to complete the full task.

tivity segmentation algorithm on both well-constrained, simulation-generated videos and unconstrained, naturalistic videos, namely the ActivityNet [10] dataset. Additionally, we perform reinforcement learning experiments in a simulated multi-step two-joint reaching task, in which the reward function is inferred from demonstration video data. We empirically demonstrate that the robot can learn more effectively when provided auxiliary video data, and that performance greatly improves when localizing individual actions, compared to learning from unsegmented videos.

II. Related Work

Learning from observation (LfO) is a recent area of research that aims to perform imitation learning on general sensory inputs, such as visual signals, without access to the true state or action labels [5], [3], [4]. Imitation learning has been performed in an LfO setting by training a context translation network by using a feature tracking loss in image space [5], by finding the most discriminative features directly related to subgoals [3], or by inferrring a reward function via a self-supervised embedding network [4]. However, all of these approaches assume pre-aligned videos of single-step tasks, restricting their applications.

Recently, several one-shot imitation learning methods have been proposed to obtain high sample efficiency in the LfO setting [11], [12]. While other LfO approaches typically require performing reinforcement learning on an inferred reward function, one-shot imitation learning generates a policy directly after observing a single demonstration. Recent examples have utilized attention-based models [12] and meta-learning methods [11]. However, these methods are restricted to single-step tasks within a distribution of task instances similar to those seen in the training data.

The action localization problem has been well-studied in the computer vision literature [13], [14]. The most closely related work is that of [15], in which they propose a similar few-shot action localization problem and solve it through a meta-learning framework. However, their approach requires a specialized network architecture—a full context embedding network—whereas our approach is fully general, allowing the flexibility of choosing any network architecture. Additionally, though we share the same goal of one-shot action localization, our ultimate goal is to apply learning from observation on top of action-localized multi-step videos.

The work of Hausman et al. [16] is also closely related to our problem setting. Similar to our goal of segmenting videos involving multiple subtasks, they suggest a method that can discover skills by inferring latent categorical codes through an InfoGAN-like [17] formulation. Although their method successfully imitates multiple policies for different skills with a single neural network, it requires state-action pairs, rather than purely observational data. By contrast, our LfO approach can be applied to videos without any action labels.

III. One-Shot Multi-Step Task Learning

We address the one-shot multi-step task learning problem in two parts, which we describe in the following subsections: (1) a one-shot activity localization algorithm, followed by (2) an LfO algorithm that is used separately for each subtask.

A. One-Shot Activity Localization

LfO algorithms typically require large amounts of well-aligned, preprocessed examples of a task (or steps of a task, in our case). Rather than relying on this, we assume that we have only a single, user-segmented demonstration of the task. We then propose to use a one-shot activity localization algorithm that can identify clips of activities in auxiliary videos that match the activities in each of the segments of the demonstration, providing additional examples of each. Thus, given a source of unsegmented auxiliary task-relevant videos, this approach can automatically gather activity-level video data for any new task.

In this paper, a video $\mathbf{v}$ is treated as a sequence of short snippets (typically of some small fixed length, e.g. 5 frames): $\mathbf{v} = [v_1, v_2, ..., v_t]$. A demonstration video is given in which K different activities are shown; the demonstration snippets have associated labels $\mathbf{l}^{demo} = [l_1, \ldots, l_t] \in [1, \ldots, K]^t$ corresponding to the activity that each one belongs to. For some new unlabeled target video $\mathbf{v}^{target}$, action localization can be defined as a classification problem in which a classifier assigns a label to each of the snippets in $\mathbf{v}^{target}$. Thus, unlike a standard classification problem with a fixed set of labels, the number of labels (and the corresponding activities) are determined by the demonstration. To address this classification problem in a one-shot manner, we use a framework based on model agnostic meta learning (MAML) [18] and its first order approximated version, Reptile [19], with a deep neural network.

MAML and Reptile aim to find good initial network parameters θ that can efficiently adapt to new problems with

a single (or small number of) stochastic gradient descent steps. Parameters found with these frameworks are not for any specific problem; instead, MAML and Reptile directly optimize for performance of the network when the parameters are fine-tuned via SGD step(s) on a small amount of training data for a particular problem. In our case, we wish to learn initial parameters that allow us to successfully train an activity classifier with only a single demonstration (a small number of snippets) of each activity.

Finding these initial parameters can be done through SGD as in other deep learning methods. MAML requires a set of training tasks $\tau \in \mathcal{T}$, and criteria C_τ to calculate the goodness of the parameters—how well the initial parameters perform after one SGD step on a new problem τ. The gradient descent step is formally defined as:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\tau \in \mathcal{T}} C_\tau(\ \cdot\ ; \theta_\tau), \tag{1}$$

where $\theta_\tau = \theta - \alpha \nabla_\theta C_\tau(\cdot; \theta)$, and α and β are learning rates for finetuning and MAML training respectively.

In our setting, a classification problem τ is defined by the set of activity labels for a given demonstration video. The loss function C_τ is a softmax cross entropy loss since the problem is a K-way classification problem. Therefore, Eq. 1 is formally represented as:

$$\theta \leftarrow \theta - \beta \nabla_\theta \sum_{\tau \in \mathcal{T}} C_\tau(\boldsymbol{v}_\tau^{target}, \boldsymbol{l}^{target}; \theta_\tau), \tag{2}$$

where $\theta_\tau = \theta - \alpha \nabla_\theta C_\tau(\boldsymbol{v}_\tau^{demo}, \boldsymbol{l}_\tau^{demo}; \theta)$, and both $(\boldsymbol{v}_\tau^{demo}, \boldsymbol{l}_\tau^{demo})$ and $(\boldsymbol{v}_\tau^{target}, \boldsymbol{l}_\tau^{target})$ are data samples belonging to the same task τ.

After we learn good initial parameters θ through Eq. 2, one-shot activity localization with a demonstration video can be performed using θ_τ, which are parameters that are fine-tuned with a demonstration video $\boldsymbol{v}_\tau^{demo}$ and labels $\boldsymbol{l}_\tau^{demo}$. A neural network of parameters θ_τ is specialized for a specific task τ, and it will emit a dense classification result for every snippet from a given task video $\boldsymbol{v}_\tau$. For the case in which a target video contains snippets of other activities not included in the task τ, we set a matching threshold, such that if predictions for all the classes are below this value, we will instead output "none of the above".

B. Learning from Observation

Now, for any given demonstration video, an activity localization network can be quickly trained in a one-shot fashion. Given an auxiliary set of task-relevant videos, this network can then be used to label each snippet of these videos as one (or none) of the demonstrated activities. This provides additional examples of each activity, which will support learning a separate reward function and policy for each activity. Since labeling happens at the snippet level, this process will not necessarily yield clean, contiguous activity segments. Future work may explore the advantages of denoising these labels and identifying large, contiguous action segments, but we demonstrate that our approach to reward inference works even in the presence of moderate label noise.

The proposed learning from observation procedure consists of three consecutive components: (1) reward function inference, utilizing noisy snippet level labels for each step, (2) behavioral cloning (BC), that exploits state-action pairs from the demonstration (if available) to expedite policy learning, and (3) reinforcement learning, working on top of the inferred reward function and the initial policy produced by 1 and 2, respectively.

In order to perform reward function inference with the auxiliary snippets for each activity, we utilize a shuffle-and-learn-style loss [20]. Intuitively, we expect that accumulated rewards should (approximately) monotonically increase in a successful execution (or in a successful demonstration) of an activity. Based on this assumption, we can train a neural network g that can predict progress (i.e. outputs a single value representing the rate of activity completion), using it as a surrogate reward function.

By asking the network to predict the order of two frames from a video, we can directly model the monotonic progress of an activity. This loss can be formally written as:

$$Loss = L_{ce}(sigmoid(g(o_t, o_{t'})), \mathbb{1}(t < t')), \tag{3}$$

where g is a neural network, o_t and $o_{t'}$ are observations (video frames) of the same activity at each time step t and t', and L_{ce} is the cross entropy loss. This formulation is similar to both Shuffle-and-Learn [20] and TCN [4]; however, these methods use a triplet-based loss, whereas we have found empirically that using simple ordered pairs yields improved performance in our setting. Additionally, our approach does not require the additional hyperparameter that TCN uses to construct the positive and negative example for each triplet.

By measuring the progress of an activity through the trained function g, we can provide a surrogate reward signal to the agent; it gets a positive reward if it makes forward progress and vice versa. Many formulations of the function g are possible, such as $sigmoid(g(o_{t-1}, o_t)) - 0.5$, directly measuring progress between current time step and the previous time step. Yet, we empirically found that measuring progress between very adjacent frames is unstable. Instead, we measure progress with an initial frame o_0 as an anchor, then use a difference of raw g values as a reward function:

$$R_t = g(o_0, o_{t+1}) - g(o_0, o_t). \tag{4}$$

We train the function g for each activity by sampling pairs of video frames that both come from the same video and have the same predicted activity label. We empirically observed that trained function is fairly robust to noisy labels from the activity localization algorithm, as we will further discuss in the experiments. Furthermore, unlike the work of Sermanet et al. [4], the proposed reward function does not require time-aligned video demonstrations as input. This simple but effective algorithm for reward function inference is an additional contribution.

IV. Evaluation

We conducted two types of experiments to assess the performance of our algorithms. First, an action localization

experiment measured the performance of the the proposed algorithms in both well-constrained simulated videos and unconstrained real videos. Second, a policy learning experiment demonstrated the importance of the activity localization step for a multi-step task.

A. Activity Localization Experiment: Setup

We examined the proposed meta-learning based activity localization approach with both simulated and real-world videos. The simulated videos featured a two-joint robotic arm performing a reaching task. These videos were relatively consistent in content and presentation, making them, in principle, easier to analyze than unconstrained videos. By contrast, real-world videos present more challenges, including highly variable camera angles and environmental features. In our experiment, ActivityNet [10], which is commonly used for activity classification or detection, is adopted.

Reacher environment This dataset contains simulated videos of a two-joint robot arm trying to select and reach for targets based on color. In total, 4 potential targets of different colors are present, but the arm must reach for the correct 2 or 3 target colors, depending on the number of steps in the task. These videos were created using the Bullet physics simulator [21] with a pretrained policy that can successfully reach a target position. Note that the task involves selecting a set of multiple target colors in any order, so different videos featuring the robot solving the same underlying task (such as $\{$orange, green$\}$) can present the subtasks in a different order. Some sample videos are shown in Fig. 2.

We generated three datasets of videos with different configurations. For the first, most basic dataset, we generated videos with 4 colors, using 2 of them as targets (6 total combinations). For each combination, we generated 100 videos. Also, meta-test set videos (of unseen activities) were generated with 4 colors different from those in the training and validation sets. The second, more complex dataset was generated from a set of 6 possible colors, having a task of length 3. However, only 4 colors are shown in each scene—three target colors and one distractor color. Meta-test videos were also generated in the same manner, but with a set of 6 different colors. For each possible color combination, we generated 40 videos. The thrid dataset was designed to test learning transfer in significantly novel settings. Length 2 tasks with 36 colors were generated and used as a training and a validation set, with 3 videos for each possible task. Then the meta-trained network was tested on: videos with (1) another set of 36 different colors, (2) a bad reacher policy (under-damping around a target) with 4 different colors, and (3) a three-joint reacher arm with 4 different colors. The resolution of the videos was 64 by 64 pixels, and we used snippets of 16 raw video frames as an input for each network.

ActivityNet This dataset includes video clips of 100 human activities, such as playing golf or drinking coffee. While our algorithm is targeted to a single task video having multiple activities, the videos in this dataset only contain a single activity type. However, we can perform the same activity localization task by creating *pseudo* tasks by concatenating single-activity videos. This was done by randomly choosing 5 activities and using a random video segment from each activity. This would not be a good baseline for a segmentation algorithms, since there are stark differences between activity changes; however, given that our approach uses snippet-based activity classification, this does not give our algorithm an unfair advantage, and thus serves as a proof-of-concept that our approach can work on real-world datasets. However, this does suggest that our approach could be improved in future work by integrating our snippet-based method with more traditional segmentation that takes the broader temporal context into account.

We chose 80 activities for meta-training, and the remaining 20 activities were held out for meta-testing. We chose 5 random activities to make a pseudo multi-step task video, such that 5-way classification tasks were generated. 3D ConvNet [22] feature vectors of length 500 were extracted for every 16 frames for each video, and each of the feature vectors were used as an input snippet for our algorithm. Because the same set of activities were displayed across demonstration and target videos in both datasets, we set the prediction suppressing threshold low-enough so that no "none of the above" predictions we made.

Network architectures and baselines For the reacher environment, a three-layer 3D CNN was used across different configurations as a base architecture, and only the final classification layer was modified, depending on the number of steps in a target task. For example, for the base reacher environment with two steps, a fully connected layer outputting two logits was added on top of the base architecture. We used our MAML-based framework for training. Detailed hyperparameters used in the experiment can be found in the publicly available code[1].

A neural network sharing the same base network architecture was used as a baseline. It was trained with classification objective but with all the possible labels existing in the training set videos: the number of colors in its configuration. Then, we removed the last classification layer and used the trained network as a feature extractor. Activity localization was performed by assigning a label of the closest demo snippet to a target snippet in the feature space via Euclidean distance.

For the ActivityNet dataset, we used a recurrent neural network (RNN) with gated recurrent units (GRU) [23]. Specifically, two fully connected layers were applied to snippet-level feature vectors, and these outputs was passed through the RNN, followed by a final classification layer to make a dense prediction. In this setup, we used our Reptile-based framework for computational reasons. Since Reptile does not require second order derivatives, not only can the training be performed quickly, but also it does not blow up the size of computation graph when unrolling over many time steps.

A simple RNN classifier was adopted as a baseline. The RNN was trained with a classification loss sharing the same

[1]https://github.com/hiwonjoon/ICRA2019-Activity-Localize

Panel A: Reacher (2 subtasks, 4 colors each)

	Classifier	Meta Learning
Same Task	0.4820	**0.8479**
Unseen Task	0.4514	**0.6944**

Panel B: Reacher (3 subtasks, 6 colors)

	Classifier	Meta Learning
Same Task	0.7410	**0.7852**
Unseen Task	0.6512	**0.7189**

Panel C: Reacher (2 subtasks, more colors)

	Classifier	Meta Learning
Same Task (36 colors)	0.7418	**0.7867**
Unseen Task (36 colors)	0.6749	**0.7015**
Bad Policy Arm	0.5432	**0.6337**
Three-joint Arm	0.5218	**0.6161**

Panel D: ActivityNet Dataset

		RNN Classifier	RNN Reptile
mIoU	Validation set	0.2121	0.3585
	Meta-test Set	0.2245	0.2883
Accuracy	Validation Set	0.2266	0.4894
	Meta-test Set	0.2428	0.4077

TABLE I: Activity localization results on Reacher-domain and the real-world ActivityNet dataset; mIoU is displayed.

base architecture, and it was used as a feature extractor in the same fashion as the reacher environment. The activity localization was also performed in the same way as in the reacher experiments.

B. Activity Localization Experiment: Results

Our results are presented in Table I. Our meta-learning methods showed significantly better results than the baseline classifier for both the simulated and real world domains, even when using the validation sets for which the baseline was directly optimized. Though a slight drop in the performance can be observed for all meta-test set cases, it still demonstrates that the suggested method is able to adapt quickly without requiring a large video dataset for a novel task. Though the mean intersection-over-union (mIoU) performance on real videos dropped noticeably compared to the simpler simulated videos, there is large room for improvement since we employed a very simple network for this proof-of-concept experiment. Future work may also include the exploration of methods to suppress the high frequency noise in dense predictions, as well as techniques to more directly handle the large variance in real-world video via specialized network architectures like TCN [4].

C. Policy Learning with Video Snippets: Setup

Next, we evaluated the the full multi-step learning pipeline in the two-joint reacher arm environment discussed in the previous subsection. This experiment is designed to show (1) the importance of activity localization for multi-step task learning and (2) the robustness of our reward function inference approach to noise in the localization step.

We generated 799 new videos for the {orange, green} task, and snippets of the videos were classified using the meta-trained model from the previous section, with one demonstration video as input. To investigate how the noise in the predicted snippet labels might affect reward function inference (compared to using perfectly segmented videos) and policy learning (compared to having ground truth rewards), we compared the performance of policies based on reward functions inferred from MAML with policies learned from (1) ground truth rewards (upper bound for RL performance), (2) rewards inferred from perfectly segmented videos (upper bound for the suggested reward function learning method), and (3) rewards inferred from unsegmented videos (baseline). Reward function inference was performed via the shuffle-and-learn-style LfO approach described in Section III-B.

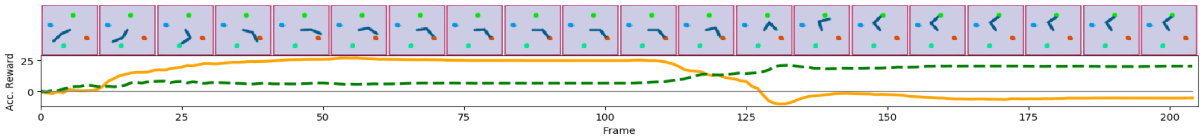
Reward function inference was conducted by training the proxy function g. We designed a two-stage neural network for g: a frame embedder and a progress predictor. The frame embedder converts a raw RGB image into a fixed-length embedding vector, and the progress predictor takes two such embeddings as input and emits a single logit value representing the relative order between the two input frames. A neural network having three 2D convolutional layers followed by two fully connected layers was used for the frame embedder, and a two-layer, fully connected neural network was used for the progress predictor. The embedding feature vector length was 64. Both of the networks were trained in a end-to-end fashion with the aforementioned loss functions. Detailed hyperparameters, such as kernel size, can be found in the publicly available code.

The policies were trained using proximal policy optimization (PPO) [24] using the implementation from OpenAI baselines [25]. A two-layer, fully connected neural network was used to represent the policy, with identical hyperparameters for each of the policies. The performance of a learned policy was measured by success rate among 300 trials; if the robot reached a target location and remained there for more than 32 frames, then a policy rollout was regarded as a success.

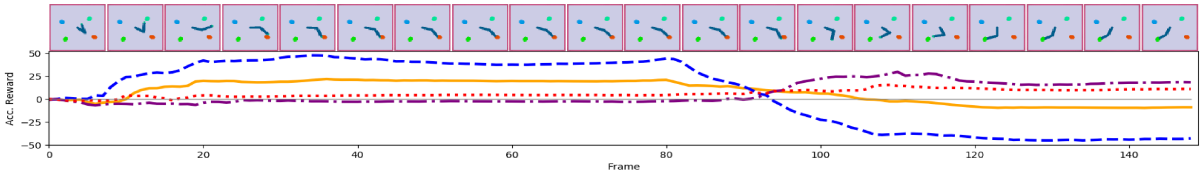
D. Policy Learning with Video Snippets: Results

Reward Function Inference Results In Figure 2, we show the accumulated reward from the inferred reward function of each subtask during a successful task execution (2a), as well as a comparison of accumulated reward for the orange subtask when the reward function is inferred using different segmentation methods (or no segmentation at all) (2b). First, we can clearly observe that each of the trained models effectively represents the degree of completion for each different subtask; the value goes up only as it reaches the correct target. The results also confirmed the necessity of video segmentation—without segmentation, the inferred reward function did not correctly represent any of the subtasks individually, nor the overall task.

RL Results The results of the reinforcement learning experiments are shown in Table II and Figure 3. We confirmed that inferred reward functions based on only raw video frames can be used to generate meaningful policies for multi-step tasks, but only when videos are properly divided into subtasks, as the reward signal learned without separation does not result in a successful policy. Furthermore, it was also possible to generate a successful policy with noisy



(a) Accumulated reward under the reward functions inferred by our algorithm for the orange subtask (orange line) and the green subtask (green dashed line) during an execution of the task in which the agent reaches first for the orange target and then for the green target.



(b) The agent reaches for the orange target, followed by the green target. The plot shows accumulated rewards under the reward function for the orange subtask when learning was performed with ground-truth video segmentations (blue dashed line), MAML segmentations (orange solid line), a single demonstration video (purple dash-dot line), and whole videos without any segmentation (dotted red line). While the reward function from MAML-segmented videos displays a similar pattern as that of ground-truth (increasing during movement toward orange, decreasing during movement toward green), the reward function inferred from only a single demonstration video and unsegmented videos does not, since those functions are either inferred from too little data or with both relevant and irrelevant frames.

Fig. 2: Reward function inference results with validation set videos on two-joint reacher environment.

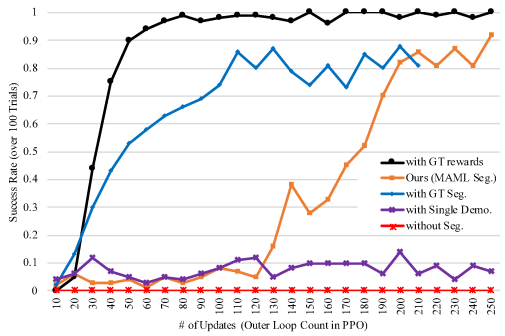


Fig. 3: The policy success rate vs. number of PPO iterations. Reward functions from 5 different sources for the 2-joint reacher task (orange target) are shown.

activity predictions from the proposed meta-learning method, rather than requiring perfect segmentation of a video. Interestingly, even with the imperfect activity localization that our method generated, it (asymptotically) resulted in a policy that performed as well as the perfectly segmented case, providing evidence that this approach may work with noisy localization on real-world video as well. Furthermore, it can be seen that performance is poor when only learning from a single demonstration; without auxiliary data, there is not enough data to enable generalization, motivating our approach.

	Target Orange	Target Green
with GT rewards	0.9600 (288/300)	0.9833 (295/300)
Ours (MAML)	0.8533 (256/300)	0.6800 (204/300)
with GT Seg.	0.7867 (236/300)	0.8300 (249/300)
with Single Demo.	0.04 (12/300)	0.07 (21/300)
without Seg.	0.0 (0/300)	0.0 (0/300)

TABLE II: Reinforcement learning results with inferred reward functions and PPO algorithm.

V. Conclusion

We addressed the challenging problem of learning multi-step tasks in a one-shot LfO setting by introducing novel algorithms that (1) perform one-shot activity localization in auxiliary videos to provide additional examples of each activity, and (2) infer reward functions for each step of the task individually, using the action-localized videos. We first examined the activity localization algorithm on both simulated and real-world datasets, showing that our approach can successfully classify snippets of a video into activities defined by a single segmented demonstration video. Second, our full proposed learning pipeline was tested on a multi-step reaching task. Our proposed method successfully completed these tasks, while the baseline LfO methods—which did not take the multi-step nature of the task into account or did not use auxiliary data—were shown to fail.

We anticipate that this work will serve as a step toward more general LfO algorithms that will be able to fully leverage unstructured web-scale video data for complex, multi-step tasks in the future. To achieve this, the most important piece of future work is scaling up this framework to work with real-world video data. This will present several challenging issues such as, including coping with visual differences between environments that auxiliary videos are captured in and handling multiple viewpoints or unsteady egocentric video. Another promising direction of future work is exploiting additional data modalities, such as text-based descriptions of tasks or audio signals from videos that can help to detect, segment, and infer reward functions from actions in a video.

Acknowledgment

This work has taken place in the Personal Autonomous Robotics Lab (PeARL) at The University of Texas at Austin. PeARL research is supported in part by the NSF (IIS-1724157, IIS-1638107, IIS-1617639, IIS-1749204) and ONR (N00014-18-2243).

REFERENCES

[1] B. D. Argall, S. Chernova, M. Veloso, and B. Browning, "A survey of robot learning from demonstration," Robotics and autonomous systems, vol. 57, no. 5, pp. 469–483, 2009.

[2] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, and M. Wray, "Scaling egocentric vision: The epic-kitchens dataset," arXiv preprint arXiv:1804.02748, 2018.

[3] P. Sermanet, K. Xu, and S. Levine, "Unsupervised perceptual reward for imitation learning," in Proceedings of Robotics: Science and Systems, 7 2017.

[4] P. Sermanet, C. Lynch, J. Hsu, and S. Levine, "Time-contrastive networks: Self-supervised learning from multi-view observation," arXiv preprint arXiv:1704.06888, 2017.

[5] Y. Liu, A. Gupta, P. Abbeel, and S. Levine, "Imitation from observation: Learning to imitate behaviors from raw video via context translation," arXiv preprint arXiv:1707.03374, 2017.

[6] S. Niekum, S. Osentoski, G. Konidaris, and A. G. Barto, "Learning and generalization of complex tasks from unstructured demonstrations," in Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on. IEEE, 2012, pp. 5239–5246.

[7] G. Konidaris, S. Kuindersma, R. Grupen, and A. Barto, "Robot learning from demonstration by constructing skill trees," The International Journal of Robotics Research, vol. 31, no. 3, pp. 360–375, 2012.

[8] M. Lopes, F. S. Melo, and L. Montesano, "Affordance-based imitation learning in robots," in Intelligent Robots and Systems, 2007. IROS 2007. IEEE/RSJ International Conference on. IEEE, 2007, pp. 1015–1021.

[9] D. K. Misra, J. Sung, K. Lee, and A. Saxena, "Tell me dave: Contextsensitive grounding of natural language to mobile manipulation instructions," in in RSS. Citeseer, 2014.

[10] B. G. Fabian Caba Heilbron, Victor Escorcia and J. C. Niebles, "Activitynet: A large-scale video benchmark for human activity understanding," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 961–970.

[11] C. Finn, T. Yu, T. Zhang, P. Abbeel, and S. Levine, "One-shot visual imitation learning via meta-learning," in Conference on Robot Learning, 2017.

[12] Y. Duan, M. Andrychowicz, B. Stadie, J. Ho, J. Schneider, I. Sutskever, P. Abbeel, and W. Zaremba, "One-shot imitation learning," in Advances in Neural Information Processing Systems (NIPS), 2017.

[13] R. Poppe, "A survey on vision-based human action recognition," Image and vision computing, vol. 28, no. 6, pp. 976–990, 2010.

[14] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, and L. Fei-Fei, "Large-scale video classification with convolutional neural networks," in Proceedings of the IEEE conference on Computer Vision and Pattern Recognition, 2014, pp. 1725–1732.

[15] H. Yang, X. He, and F. Porikli, "One-shot action localization by learning sequence matching network," in The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2018.

[16] K. Hausman, Y. Chebotar, S. Schaal, G. Sukhatme, and J. Lim, "Multi-modal imitation learning from unstructured demonstrations using generative adversarial nets," in Advances in Neural Information Processing Systems, 2017.

[17] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "Infogan: Interpretable representation learning by information maximizing generative adversarial nets," in Advances in Neural Information Processing Systems, 2016, pp. 2172–2180.

[18] C. Finn, P. Abbeel, and S. Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in Proceedings of the 34th International Conference on Machine Learning, 2017, pp. 1126–1135.

[19] A. Nichol and J. Schulman, "Reptile: a scalable metalearning algorithm," arXiv preprint arXiv:1803.02999, 2018.

[20] I. Misra, C. L. Zitnick, and M. Hebert, "Shuffle and Learn: Unsupervised Learning using Temporal Order Verification," in ECCV, 2016.

[21] E. e. a. Coumans, "Bullet physics sdk," https://github.com/bulletphysics/bullet3.

[22] D. Tran, L. Bourdev, R. Fergus, L. Torresani, and M. Paluri, "Learning spatiotemporal features with 3d convolutional networks," in Computer Vision (ICCV), 2015 IEEE International Conference on. IEEE, 2015, pp. 4489–4497.

[23] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP). Association for Computational Linguistics, 2014, pp. 1724–1734.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," arXiv preprint arXiv:1707.06347, 2017.

[25] P. Dhariwal, C. Hesse, M. Plappert, A. Radford, J. Schulman, S. Sidor, and Y. Wu, "Openai baselines," https://github.com/openai/baselines, 2017.