Robotics: Science and Systems 2020
Corvalis, Oregon, USA, July 12-16, 2020

Concept2Robot: Learning Manipulation Concepts from Instructions and Human Demonstrations

Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg

***Abstract*—We aim to endow a robot with the ability to learn manipulation concepts that link natural language instructions to motor skills. Our goal is to learn a single multi-task policy that takes as input a natural language instruction and an image of the initial scene and outputs a robot motion trajectory to achieve the specified task. This policy has to generalize over different instructions and environments. Our insight is that we can approach this problem through *Learning from Demonstration* by leveraging large-scale video datasets of humans performing manipulation actions. Thereby, we avoid more time-consuming processes such as teleoperation or kinesthetic teaching. We also avoid having to manually design task-specific rewards. We propose a two-stage learning process where we first learn single-task policies through reinforcement learning. The reward is provided by scoring how well the robot visually appears to perform the task. This score is given by a video-based action classifier trained on a large-scale human activity dataset. In the second stage, we train a multi-task policy through imitation learning to imitate all the single-task policies. In extensive simulation experiments, we show that the multi-task policy learns to perform a large percentage of the 78 different manipulation tasks on which it was trained. The tasks are of greater variety and complexity than previously considered robot manipulation tasks. We show that the policy generalizes over variations of the environment. We also show examples of successful generalization over novel but similar instructions.**

I. Introduction

Humans have gradually developed language, mastered complex motor skills, created and utilized sophisticated tools, and built scientific theories to understand and explain their environment. Concepts are fundamental to these abilities as they allow humans to mentally represent, summarize, and abstract diverse knowledge and skills [42]. By means of abstraction, concepts that are learned from a limited number of examples can be extended to a potentially infinite set of new and unseen entities. Current robots lack this generalization ability which hampers progress towards deploying them to real domestic, industrial or logistic environments.

Most related work in cognitive science and robotics have focused on lexical concepts that correspond to words in natural language [42]. A large body of work in robotics has studied how robots can acquire concepts that relate to objects, their attributes (like color, shape, and material), and their spatial relations [17, 43, 51, 1]. In this paper, we endow a robot with the ability to acquire *manipulation concepts* [25] that can be seen as mental representations of verbs in a sentence.

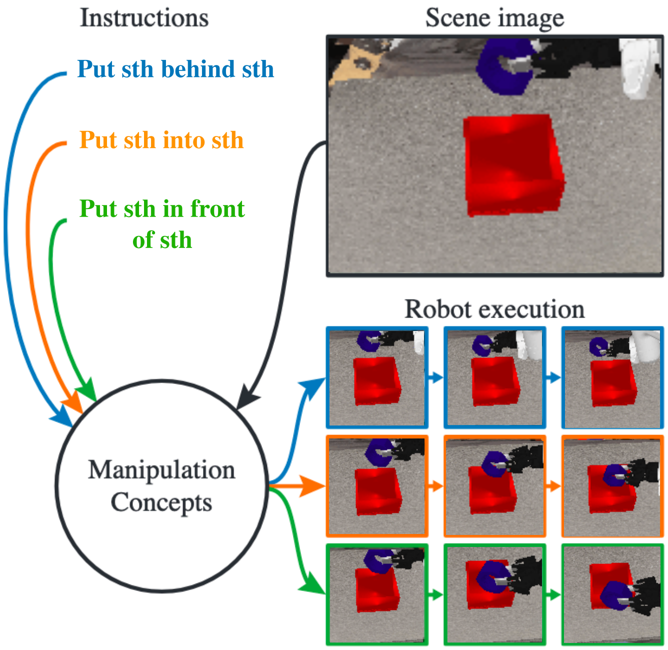

Fig. 1: We propose a method that learns manipulation concepts from videos of human demonstrations labelled with natural language instructions. After training, our network takes as input a task instruction and image of the start scene, and then outputs a robot motion trajectory that will succeed at the task.

The way concepts are represented and acquired is debated within Cognitive science. In the classical view, concepts reflect contingencies in the environment that can be perceived through different sensors [28]. In this paper, we follow this classical view and propose a robot learning framework to acquire manipulation concepts from human video demonstrations. Here, manipulation concepts reflect contingencies in the demonstration videos that instantiate specific natural language instructions. By using videos, we can leverage large-scale datasets such as *20BN-something-something* (*Sth Sth*) [13] instead of providing demonstrations through the more time-consuming processes of kinesthetic teaching or teleoperation.

Specifically, we train a model that takes as input an image of the environment and a natural language instruction. The output of the model is a robot motion trajectory that is executed open-loop. During training, we compute a reward for the executed trajectory by scoring the resulting robot video using the video classifier trained on *Sth Sth*. In experiments, we show how our framework is capable of learning 78 different single-task policies from human video demonstrations that generalize over

L. Shao, T. Migimatsu, K. Yang and J.Bohg are with the Stanford AI Lab, Stanford University, Stanford, California 94305. Email: {lins2,takatoki, kaiyuany, bohg}@stanford.edu. Q. Zhang is with the Zhiyuan College, Shanghai Jiao Tong University, 200240 Shanghai, China. Email: zhangqiang2016@sjtu.edu.cn. Toyota Research Institute ("TRI") provided funds to assist the authors with their research but this article solely reflects the opinions and conclusions of its authors and not TRI or any other Toyota entity.

variations of the environment—in our case, varying object poses. Given these single-task policies, we can then learn a multi-task model. For this part, we use an imitation learning approach where the multi-task policy imitates the trajectories of the single-task policies. This multi-task model can then not only generalize over variations of the environment but also accept instructions that are similar to those it was trained on.

Due to the complex nature of learning manipulation concepts from human visual demonstrations and natural language instructions, our learning framework is composed of many subcomponents that each plays a critical role in the working of the entire system. The focus of our paper is not the specific algorithms used for each subcomponent, but rather the combination of these subcomponents into a whole learning framework. Our primary contributions are as follows: 1) a novel learning framework to acquire a unified model of diverse manipulation concepts from human visual demonstrations and language instructions, 2) a suite of 78 diverse simulated manipulation tasks for multi-task learning, 3) extensive evaluation and demonstrations on our approach in simulation on the 78 manipulation tasks.

II. Related Work

A. Vision and Language Grounding

In this paper, we are interested in associating natural language, especially verbs, with robot motion skills for various manipulation scenarios. Bridging vision with language has been extensively studied for captioning images or videos with natural language [8, 9, 49, 50, 52]. Using natural language and vision to guide mobile robot behaviors also has a rich literature, ranging from pioneering work on connecting language instructions such as "pointing left" and "pointing right" with the behavior experiences of a mobile robot [43] to more recent work on vision-language navigation [51, 1].

Recently, the use of natural language in robot learning has gained increasing interest [31]. Task descriptions formulated in natural language are used to condition policy learning. Jiang et al. [18] use language to structure compositional task learning in the context of hierarchical RL, where the high-level policy produces language instructions that direct the low-level policy. Natural language can induce the rewards in an inverse reinforcement learning context. Bahdanau et al. [4] proposed language-conditional reward functions trained on (instruction, goal_state) pairs from demonstrations. Transferring knowledge from unsupervised language learning on large web corpora allows learned policy to generalize to instructions outside of the training distribution.

Our approach is more related to work on using natural language and vision to learn manipulation skills. Shu et al. [39] use hierarchical reinforcement learning with a stochastic temporal grammar to decide when to use previously learned policies or acquire new skills based on their language descriptions. Jiang et al. [17] similarly use the compositional structure of language to learn hierarchical abstractions of manipulation skills. However, these works are limited to tasks such as fetching, arranging, and sorting objects. We propose a learning framework that allows a robot to autonomously associate language instructions with a diverse set of 78 manipulation skills (Fig. 3), including complex tasks such as "scoop something up with something" or "let something roll up a slanted surface".

B. Learning from Visual Demonstration

Learning from Demonstration (LfD) [3] enables a robot to learn a policy from expert demonstrations. LfD significantly reduces human effort in robot skill learning by avoiding the need to manually design task-specific reward functions. Because we propose to learn from human activity videos, here we review only approaches that are based on visual data. For a more comprehensive review of LfD, we refer to [3].

Recent advances in processing large-scale image and video datasets with deep learning have made it possible to leverage visual data for LfD. Fu et al. [10] propose a variational inverse reinforcement learning method to learn policies when a large number of desired goal state samples are available. Singh et al. [41] present a follow-up work to enable a robot to learn from a modest number of successful examples followed by actively solicited queries, where the robot shows the user a state and asks whether the task was successfully completed. Gupta et al. [14] propose a method for solving long-horizon tasks which contains an imitation learning stage to produce goal-conditioned hierarchical policies, and a reinforcement learning phase to fine-tune these policies for task performance. The goal is to reproduce an image of the final successful state. However, the final state may not be enough to characterize a manipulation skill where the shape of the motion trajectory is important, such as pouring water into a cup.

In this paper, we propose to evaluate the entire video of a robot action by leveraging a video classifier trained on a large-scale video dataset [13]. This is in contrast to using single images to classify goal states. One concern in learning from human activity videos is the domain gap between the videos that the classifier is trained on and the videos rendered from the robot simulation environment. However, our hypothesis is that a video-based activity classifier captures the visual dynamics of the different actions rather than the appearance of the environment. If the visual dynamics are shared between the simulated and real environments, then a video-based action classifier should be able to guide the robot to successfully learn various manipulation concepts. In experiments, we demonstrate that reward provided by a video classifier is indeed more informative than an image-based classifier.

C. Motion Trajectory Representation

How to best represent motion skills is an age-old question in robotics with a rich literature. Pastor et al. [34] propose *Dynamic Movement Primitives* (DMPs) to encode movement trajectories using the attractor dynamics of a nonlinear differential system. DMPs are commonly used for representing and learning basic movements in robotics, such as swinging tennis rackets [16, 29], playing drums [48], or writing [23]. Khansari-Zadeh and Billard [21] propose learning nonlinear dynamical systems with Gaussian mixture models that are robust to spatial and temporal perturbations. Paraschos et al. [32] and Meier et al. [30] present a probabilistic formulation of DMPs that maintains a distribution over trajectories to enable composition and blending of multiple primitives.

In contrast to these methods, we adopt a simple second-order spring-damper system with auxiliary forces at each timestep to modify the shape and velocity profile of the trajectory. The simplicity and efficiency of this attractor system works well for our application, where a complex neural network needs to learn and output these trajectory parameters from high-dimensional language and image inputs. In experiments, we show that the auxiliary forces are an important ingredient to this motion representation in addition to the goal to shape the trajectories of more complex manipulation tasks.

D. Multi-Task Learning

If we want robots to be broadly useful in realistic environments, we need algorithms that can learn a wide variety of skills reliably and efficiently. To this end, Yu et al. [53] released a multi-task learning benchmark containing simulation environments for 50 different manipulation tasks. While this benchmark uses one-hot vectors to represent each task, we use natural language instructions as input to our multi-task policy network. In this way, we can achieve generalization to various even unseen instructions and manipulation scenes.

In terms of learning procedures, the majority of previous multi-task learning (MTL) work mainly falls into two families: (i) modular policy design [2], where a multi-task policy maintains separate subpolicies for different tasks, and (ii) knowledge transfer through distillation [36, 33, 45], where a single unified policy is learned from multiple experts. Devin et al. [6] decompose network policies into task-specific and robot-specific modules to enable robots to learn multiple skills from other robots. Competing objectives for individual tasks can make MTL difficult, but Sener and Koltun [37] formulate MTL as a multi-objective optimization with the goal of finding a Pareto optimal solution. Teh et al. [44] propose using a shared "distilled" policy that captures common behavior across tasks. Each single-task policy is trained to solve its own task while constrained to stay close to the shared policy, while the shared policy is trained to be the centroid of all task policies.

Our multi-task learning method falls under the policy distillation category. Single-task policies are trained to solve individual tasks, and after their performance stabilizes, we train the multi-task policy to imitate the behaviors of all the single-task policies. In experiments, we show how our approach outperforms related policy distillation approaches.

III. Technical Approach

A. Overview

Our model takes as input a natural language instruction describing the task along with an RGB image of the initial scene, and outputs the parameters of a motion trajectory to accomplish the task in the given environment. These inputs are first fed into a semantic context network to combine the information from natural language with the visual perception of the robot in order to produce a joint task embedding. This serves as a description of the desired task. The task embedding is then input to a policy network, which synthesizes the parameters of a motion trajectory. The trajectory is finally executed by the robot with Operational Space Control [22]. An overview of the system is given in Fig. 2.

We learn manipulation concepts from human demonstrations by scoring how closely a robot resembles a human executing the same task. For scoring, we use a video-based action classifier that is trained on human activity videos in *Sth Sth*. With this classifier as a proxy reward, we first learn single-task policies through reinforcement learning. Then, we train a multi-task policy over all the tasks by performing imitation learning on the single-task policies. The final result is a multi-task policy that can take a new natural language instruction and environment image and execute the desired task using its knowledge base of 78 previously learned tasks.

In this section, we describe the subcomponents of our framework. While we chose to implement each subcomponent with certain algorithms, the focus of this paper is not the specific algorithms. Instead, the focus is the integration of these subcomponents to allow learning manipulation concepts from rewards that are provided by a video classifier trained to distingiush human manipulation actions. We leave the optimization of these subcomponents to future work. To ensure reproducability, we provide a detailed description of the implementation in an appendix in addition to the source code, both available on https://sites.google.com/view/concept2robot.

B. Semantic Context Network

1) Instruction Encoding: To represent a natural language instruction, we tokenize the sentence and feed it into BERT [7] to extract an instruction feature with dimension 1024. This feature is fed into a *Multilayer Perceptron* (MLP) to reduce the dimensionality to 128. The weights of the MLP are optimized during training, while the BERT model is frozen.

2) Vision Encoding: To represent the image of the initial scene, we leverage ResNet18 [15]. The resulting feature is fed into an MLP to reduce the dimensionality to 256. The weights of both ResNet18 and the MLP are optimized during training.

We concatenate the instruction and image features to get the final task instance embedding of dimension 384. This vector is fed into a policy network to produce a motion trajectory.

C. Policy Network

The objective of the policy network is to output the parameters of an open-loop motion trajectory that, when executed by the robot, achieves the manipulation task in the current environment. In this paper, we choose to represent the motion trajectory by a second-order dynamical system of the form:

$$\ddot{y}(t) = k_p(g - y(t)) - k_d\,\dot{y}(t) + f(t) \qquad (1)$$

where y, $\dot{y}$, and $\ddot{y}$ are position, velocity, and acceleration of the end-effector, g is the goal pose, and f are the additional forces at each timestep that modify the shape and velocity profile of the trajectory. k_p and k_d are standard PD control gains. This

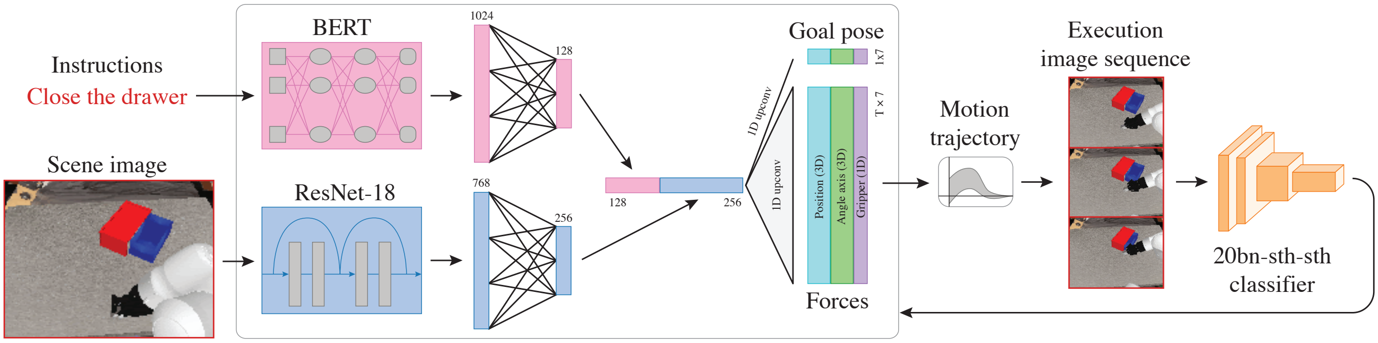

Fig. 2: System architecture. The input to the network is an initial image of the environment and a natural language command from the 20BN Something Something Dataset. ResNet18 [15] and BERT produce feature vectors for the image and the command sentence, respectively. These feature vectors are concatenated and processed with 1D upconvolution to produce 7-dof vectors for the trajectory goal pose and forces, where 3-dof represent position, 3-dof represent angle axis orientation, and the last dof represents the gripper state. The controller for this trajectory is executed in simulation, and a classifier trained on the 20BN Something Something Dataset classifies the task from the resulting image sequence. The classification score is treated as a proxy reward for completing the task, and is fed back to the network for training.

kind of dynamical system is analytically well understood and is widely utilized in the robotic community [21].

While the above representation is similar to a DMP, DMPs have a canonical system in which the phase variable exponentially decays to zero [34]. The forcing term is weighted by the phase variable, thus also decaying to zero and ensuring convergence. DMPs have often been used for reaching movements, where it makes sense to slow down towards the end. However, we are considering a variety of 78 tasks involving contact, such as “hit sth with sth” or “close sth with sth”, where diminishing forces is less desirable. For example, for “close sth with sth”, the robot needs to apply large forces until the end to overcome the sliding friction of the drawer. While it may be possible to capture such motions with DMPs by, for example, placing the goal behind the drawer,we found it difficult for the neural network to learn trajectories in this manner. Thus, we choose a free parameterization of the forcing terms so the network has the freedom to apply large forces at the end, without being hindered by exponential decay. While this means we can no longer ensure convergence, the video classifier encourages the network to learn stable trajectories.

The motion trajectory operates in a 7-dimensional space: 3D position and orientation of the end effector and 1D movement of the two-fingered gripper. For orientation, we adopt the angle axis representation, which has been shown to be effective for neural networks due to its lack of parameter constraints [38].

The final task embedding from the semantic context network with shape 1×384 is simultaneously fed into an MLP (equivalent to 1D upconvolution) to generate the goal g with shape 1×7 and into a separate 1D upconvolution network to generate the forces f with shape $T \times 7$, where T is the number of timesteps in the trajectory. The force network applies 1D upconvolution along the time dimension of the task embedding to expand its size from 1 to T, and reduces the feature channels in the second dimension from 384 to 7. We choose to use upconvolutions across time because there may be trajectory features that are independent of time. Upconvolutions allow us to share weights across the trajectory, reducing the number of parameters required and thus improving learning efficiency [27].

D. Video Classification Network

In our LfD framework, the unknown reward function for a given task $\mathcal{T}_i$ is substituted by a video classifier that is trained to classify actions in *Sth Sth* videos. We use 3D convolution for the classifier, which is reported to be effective for learning spatio-temporal relationships in videos [47].

Note that after training on the *Sth Sth* dataset, we fix the weights of the video classifier. When training the policy network, the output trajectory is executed open-loop by the robot and rendered into a video. This video is then scored by the video classifier. For a given task $\mathcal{T}_i$ and a video $\mathcal{V}$, the classification score of the corresponding category is used as a reward signal $R(\mathcal{V}, \mathcal{T}_i)$ to update the policy network.

E. Single-Task Learning

As a first step towards learning manipulation concepts, we train single-task policies π_i per task $\mathcal{T}_i$, for $i \in [1 \dots 78]$. Given the video classifier as a substitute for the reward, we frame this problem as a reinforcement learning problem per task $\mathcal{T}_i$. The actions are the motion trajectory parameters $a = [g, f] \in \mathcal{A}$, which represent the goal and forces for the entire trajectory (Eq. 1). The reward is the video classification score $R(\mathcal{V}, \mathcal{T}_i)$ as defined above. Note that since the output of the policy network is directly transformed into a continuous motion trajectory, our single-task learning problem is a Contextual bandit problem [24], where the contextual information is the language instruction and the image of initial scene.

To solve the Contextual bandit problem, we combine *Deep Deterministic Policy Gradients* (DDPG) [26] with the *Cross Entropy Method* (CEM) [35]. The DDPG critic network is used to approximate the Q value function. The output of the DDPG actor network is used to initialize CEM. Then we run CEM to search the landscape of the critic network for a better action in a broad neighborhood of the initial action [20]. For the first iteration of CEM, we sample a batch of M points in the action space $\mathcal{A}$ and fit a Gaussian to the top N samples. In successive iterations, we sample M points from the Gaussian and again update the Gaussian with the top N samples. We run CEM for four iterations to select actions in both the training and test stages. In the training stages, we store the transition (s, a, r)

where s represents the state, a is the action selected after CEM, r is the reward from the video classifier. The critic loss is $\mathcal{L}_c = \|r - Q(s,a)\|^2$, and the actor loss is $\mathcal{L}_a = -Q(s, \text{Actor}(s))$, where $\text{Actor}(s)$ is the output of the actor network. Note that the weights of critic network are not updated when optimizing the loss of the actor network. In the contextual bandit problem, the critic network is trained to directly approximate the Q value, so overestimation is not an issue when using neural networks to learn the Q value [11]. Thus, the critic learns more stably than the actor, and CEM allows us to leverage the faster learning speed of the critic to improve the output of the actor. More analysis of this effect is provided in the Appendix.

F. Multi-Task Learning

After training single-task policies, we now have expert models for each task that we can use to train a multi-task model through imitation learning. We found that this staged approach works better than directly training a multi-task model from scratch and provide supporting evaluation in Sec. V.

For each task $\mathcal{T}_i$, we use the corresponding single-task policy π_i to produce N rollouts of state-action tuples $\left(s^i, a^i, \mathbf{y}^i\right)_{j=1}^{N}$ and save them into a large replay buffer. Here, s^i is the feature vector that represents the language instruction and initial image of the scene, a^i contains the motion trajectory parameters provided by π_i, and $\mathbf{y}^i$ is the actual end effector trajectory of length T recorded when the robot executes a^i.

After generating rollouts for all 78 tasks, we train the multi-task policy π_θ to generate an action $a = [g, f]$ that imitates the single-task policies by minimizing the following loss function:

$$\min_{\theta} \sum_{t=1}^{T} \left\| \hat{y}(a)_t - y_t^i \right\|^2 + \left\| g - \hat{y}(a)_T \right\|^2 + \left\| g - \hat{y}\left(a^i\right)_T \right\|^2 \quad (2)$$

where $\hat{y}(a)_T$ and $\hat{y}\left(a^i\right)_T$ are the final end effector poses of the trajectory predicted by the multi-task and single-task policy, respectively. The predicted trajectory is computed by forward simulating the dynamical system in Eq. 1. Note that $\hat{\mathbf{y}}$ usually differs from $\mathbf{y}$ due to physical contact or joint limits.

The first term in the loss function ensures that the predicted trajectory $\hat{\mathbf{y}}(a)$ from the multi-task policy π_θ is similar to the trajectory $\mathbf{y}^i$ of the single-task policy π^i. The second term ensures that the goal g of the multi-task policy is not too far from the predicted final state of the trajectory $\hat{y}(a)_T$. This could occur if the force terms f are too large towards the end of the trajectory and prevent the end-effector from reaching the goal. The third term ensures that g is not too far from the predicted final state of the single-task trajectory $\hat{y}\left(a^i\right)_T$.

Note that the gradients of $\hat{\mathbf{y}}$ with respect to g and f are non-trivial, since $\hat{\mathbf{y}}$ is computed by integrating the dynamical system in Eq. 1 twice across all timesteps. We compute the gradients based on the derivation in [12].

IV. Environment Setup

A. Human Demonstration Data

At the time of writing, the *Sth Sth* dataset contained 108,499 videos of 174 manipulation tasks, with durations of 2–6 seconds. Labels are textual descriptions based on templates, such as “dropping [something] into [something],” containing slots (“[something]”) that serve as placeholders for objects. We select 78 of the 174 tasks that are appropriate for our environment setting. The remaining 96 tasks either require dual-arm manipulation or are difficult to simulate in PyBullet [5], such as “tearing sth”. For the 78 chosen tasks, we adopt the same task IDs used in *Sth Sth*, as listed in Fig. 3.

B. Robot Environment

We use PyBullet [5] to simulate each environment associated with one of the 78 tasks. Our robot is a simulated 7-DoF Franka Panda robot arm with a two-fingered Robotiq 2F-85 gripper. A camera is statically mounted to capture the environment state as RGB images downsampled to 120×160. Every time the environment is reset after an RL episode, the manipulation objects in each environment are initialized with a random pose within manually defined bounds for each task.

C. Evaluation Metric

The reward that the robot receives from the video classification is a proxy objective that may not directly reflect how successful a policy is with respect to its task. Therefore, we manually define task-specific success metrics to evaluate (not train) the policies. For example, “hit sth with sth” is successful if there is a collision between the object grasped by the robot and the target object, and the grasped object remains in the robot’s gripper after the collision. We report the average success rate over 100 episodes.

V. Experiments

The core idea of this paper is to use a video classifier as a proxy reward for learning a multi-task policy and to specify tasks with natural language. We propose a new multi-task learning algorithm that outputs an entire robot motion trajectory. In the following experiments, we seek to justify these choices or demonstrate a performance gain over previous approaches. The hypotheses we want to test are:

H1. A motion trajectory representation that can capture complex trajectory shapes and velocity profiles is important for completing the tasks in our dataset.

H2. For learning complex tasks, it is important to take the video of an entire robot trajectory into account rather than only the before and after frames.

H3. Policies trained on the video classifier as a proxy reward for true task success perform comparably to policies trained on true task success.

H4. Our imitation learning method for multi-task learning outperforms other state-of-the-art methods.

H5. The multi-task policy performs comparably to single-task policies.

H6. Incorporating natural language through task embeddings allows the multi-task policy to generalize to novel instructions.

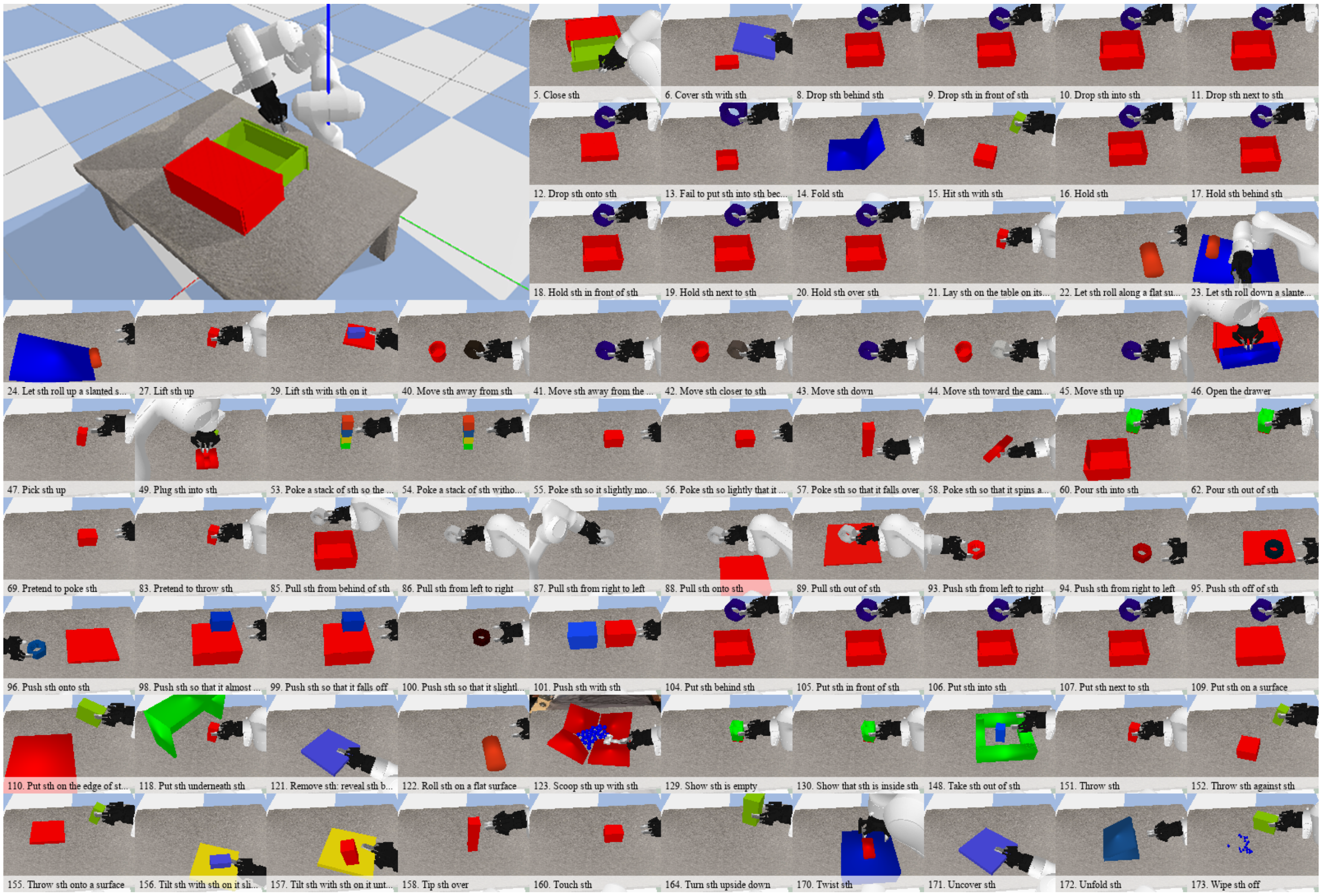
5. Close sth
6. Cover sth with sth
8. Drop sth behind sth
9. Drop sth in front of sth
10. Drop sth into sth
11. Drop sth next to sth
12. Drop sth onto sth
13. Fail to put sth into sth bec...
14. Fold sth
15. Hit sth with sth
16. Hold sth
17. Hold sth behind sth
18. Hold sth in front of sth
19. Hold sth next to sth
20. Hold sth over sth
21. Lay sth on the table on its...
22. Let sth roll along a flat su...
23. Let sth roll down a slante...
24. Let sth roll up a slanted s...
27. Lift sth up
29. Lift sth with sth on it
40. Move sth away from sth
41. Move sth away from the ...
42. Move sth closer to sth
43. Move sth down
44. Move sth toward the cam...
45. Move sth up
46. Open the drawer
47. Pick sth up
49. Plug sth into sth
53. Poke a stack of sth so the ...
54. Poke a stack of sth witho...
55. Poke sth so it slightly mo...
56. Poke sth so lightly that it ...
57. Poke sth so that it falls over
58. Poke sth so that it spins a...
60. Pour sth into sth
62. Pour sth out of sth
69. Pretend to poke sth
83. Pretend to throw sth
85. Pull sth from behind of sth
86. Pull sth from left to right
87. Pull sth from right to left
88. Pull sth onto sth
89. Pull sth out of sth
93. Push sth from left to right
94. Push sth from right to left
95. Push sth off of sth
96. Push sth onto sth
98. Push sth so that it almost ...
99. Push sth so that it falls off
100. Push sth so that it slightl...
101. Push sth with sth
104. Put sth behind sth
105. Put sth in front of sth
106. Put sth into sth
107. Put sth next to sth
109. Put sth on a surface
110. Put sth on the edge of st...
118. Put sth underneath sth
121. Remove sth: reveal sth b...
122. Roll sth on a flat surface
123. Scoop sth up with sth
129. Show sth is empty
130. Show that sth is inside sth
148. Take sth out of sth
151. Throw sth
152. Throw sth against sth
155. Throw sth onto a surface
156. Tilt sth with sth on it sli...
157. Tilt sth with sth on it unt...
158. Tip sth over
160. Touch sth
164. Turn sth upside down
170. Twist sth
171. Uncover sth
172. Unfold sth
173. Wipe sth off

Fig. 3: Initial scene images for each of the 78 tasks that form the input to the single and multi-task network. The top left image shows an overhead view of the simulation environment. More examples are shown in the supplementary video.

A. Single-Task Policies

To conduct an ablation study of our choices for the motion representation and reward signal, we first evaluate the performance of single-task policies.

1) Comparison of goal+forces to goal-only trajectories (H1): To evaluate whether a motion trajectory representation that can capture complex trajectories is important for learning tasks, we run an ablation experiment where we remove the auxiliary forces f at each timestep from our trajectory representation in Eq. 1. We train policies on the resulting straight-line trajectories with only goal poses g, which is equivalent to PD control to the goal. The results are presented in Fig. 4. The average task performance of the full trajectory representation (*Video w/ Goal+Forces*) is 61%, while the average of the goal-only representation (*Video w/ Goal*) is 54%. For most simple tasks, the goal-only representation is sufficient, but some tasks, like "show sth is empty" and "push sth so that it slightly moves" require more complex trajectory shapes or velocity profiles that can only be provided by the additional force term.

2) Comparison of video-based to image-based classification (H2): To demonstrate the importance of using videos to learn dynamic manipulation tasks, we run an ablation experiment where we replace the video-based action classifier with a classifier that only takes the initial and goal image of the trajectory as input. Because videos in *Sth Sth* do not necessarily end exactly when the task is finished, i.e. when the hand reaches the goal pose, we train the image classifier by randomly selecting a frame from the last 25% of the video to be the goal image. The image-based action classifier feeds the initial and goal images into a pretrained ResNet50 network [40]. The resulting features are concatenated and passed to an MLP with a softmax layer to output a probability distribution over the 174 manipulation tasks. The image classifier achieves a top-1 accuracy of 28% and top-5 accuracy of 43%, while the video classifier achieves 35% and 58%, respectively.

Fig. 4 (second column) compares the performance of the policies trained with the video- and image-based action classifier. In general, policies trained on the image-based classifier perform worse, particularly on tasks like "hit sth with sth" or "poke sth so that it slightly moves", which require intermediate frames in the trajectory, not just the goal image, to identify.

3) Comparison of video-based with ground truth reward (H3): In this work, we are proposing to learn manipulation concepts from a reward signal that is provided by a video-based action classifier trained on human video demonstrations. However, there is concern that this proxy reward is noisy and

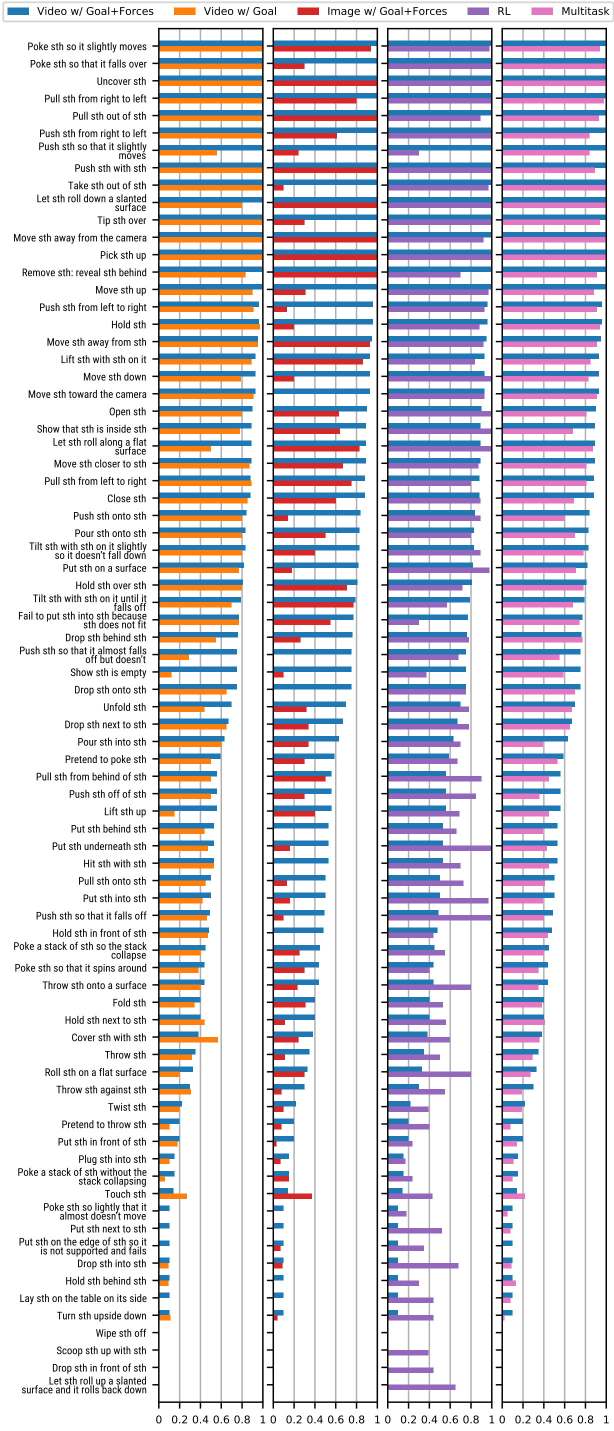

Fig. 4: Task Performance of Single and Multi-task policies. The blue bars in all plots represents the task performance of single-task policies when trained with reward from the video classifier and with the motion being represented by goal poses and forces at each time step. The bar plot on the left shows the effect of removing forces from the motion representation (orange). The second plot from the left shows the effect of replacing the video classifier with an image classifier (red). These two ablation studies show that with a few exceptions, being able to shape the trajectory with forcing terms improves task performance, and that taking the video of the entire robot motion into account instead of only before and after images also improves task performance. The third plot shows the effect of replacing the video classifier with the handcrafted RL reward that reflects true task success (purple). For most of the high-performing tasks, the video classifier performs comparably (in some cases better), while for the low-performing tasks, the video classifier faces a drop in performance, possibly due to video classification inaccuracies. The plot on the right compares single-task to multi-task performance, showing that the multi-task framework is able to maintain performance on most of the tasks.

may not provide a sufficient signal to reach a high success rate. To test this, we first gauge the difficulty of each task by training 78 single-task policies with a hand-designed, task-specific, binary reward signal that reflects real task success. To evaluate the effectiveness of using the video-based action classification score as a reward signal, we compare the performance of the resulting policies to those trained with our handcrafted reward functions. Ideally, there is no difference in task success. However, because the action classifier is not perfect, we expect some drop in performance. The results are presented in the third column of Fig. 4. Most of the high-performing, video-based policies match the performance of the "handcrafted RL" policies. Some of them even perform better, e.g. "push sth so that it slightly moves" and "fail to put sth into sth because sth does not fit". One possible explanation for this is that the ground truth rewards are binary, whereas the video-based classifier provides a continuous classification score that offers some feedback for attempts that almost achieve the task but fail. For some of the lower-performing, video-based policies, the gap to the performance of the handcrafted RL policies is quite large, indicating that the video-based action classifier only provides a noisy reward signal for these tasks. For other low-performing, video-based policies, the gap to the performance of the handcrafted RL policy is small, indicating that the task is hard to learn in general.

B. *Multi-Task Policies*

We aim to learn a unified model for all the 78 tasks such that this model can also generalize over novel instructions. In this section, we evaluate the performance of our learning method compared to baselines and show qualitative examples of how the model outputs reasonable actions for novel instructions.

1) Comparison of multi-task algorithms (H4): We first select eight tasks for a simple multi-task benchmark, shown in Table I. The tasks are grouped into complementary pairs, like "move sth away from sth"/"move sth closer to sth" and "pull sth from right to left"/"pull sth from left to right". We run our proposed multi task learning algorithm on the benchmark.

The first baseline, which we call *Simple*, trains a single multi-task policy on all eight environments with the corresponding video classification score as the reward. Because this model needs to learn a policy that maximizes the video classifier reward on all eight tasks, the objective function is complex, and policy gradient optimization easily gets stuck in local minima. For example, among the eight tasks, task 42 dominates, while the others remain low. Overall, the *Simple* baseline only achieves an average success rate of 17.4%.

We implement a second baseline based on the *Distral* algorithm [44]. Since our method generates deterministic actions, we replace the KL divergence—used to measure the difference between the single- and multi-task policy distributions—with the Euclidean distance between the actions generated by the single- and multi-task policy. *Distral* greatly outperforms *Simple*, but does not perform as well as our method. One possible explanation for this is that *Distral* requires that the multi-task policy is trained in conjunction with the single-task policies.

However, jointly optimizing the policies may be difficult in our task setting. First, the reward signal from the learned video classifier is noisy. Second, the magnitude of rewards for the optimal single-task policies may vary across tasks, since the video classifier may be more confident in scoring some tasks than others. The noise and bias in the reward function may cause *Distral* to get stuck in local optima.

The reason we learn the single-task policies independently of the multi-task policy is that the noisy and biased reward from the video classifier may cause the multi-task policy to be unstable and perform poorly in the early stages of learning. By learning the single-task policies first, we can ensure that the instability of the multi-task learning does not affect the single-task performance, especially since single-task learning is unaffected by the reward bias when trained alone.

SuccessRate%	Average	T40	T42	T86	T87	T93	T94	T103	T104
Simple	17.4	24	52	20	28	1	10	3	1
Distral	61.5	88	**78**	80	**100**	63	36	45	3
Ours	**76.3**	**94**	74	**82**	**100**	**100**	**96**	**60**	**4**

TABLE I: Comparison of multi-task baselines. The instructions for the eight tasks are “Move sth away from sth”, “Move sth closer to sth”, “Pull sth from left to right”, “Pull sth from right to left”, “Push sth from left to right”, “Push sth from right to left”, “Put sth behind sth”, and “Put sth in front of sth”. We report average success rate over four random seeds. The mean and standard deviation of the success rates of the corresponding single-task policies are: 91 ± 3.4, 78 ± 4.6, 88 ± 3.1, 96 ± 3.8, 96 ± 3.2, 94 ± 3.1, 51 ± 7.3, and 2 ± 2.2. This shows that the performance is stable across seeds.

2) Comparison of multi-task to single-task (H5): Learning a unified policy for multiple tasks is harder than learning a policy for a single task. Therefore, we expect the multi-task policy to perform worse than the single-task policies. This comparison is shown in the fourth column of Fig. 4. The multi-task policy achieves an average success rate of 54%, while the single-task policies collectively achieve 61%. As expected, this indicates a drop in performance, although not too severe. In future work, we will investigate other multi-task learning frameworks, such as from the meta-learning community.

3) Generalization to new instructions (H6): We demonstrate that our model shows reasonable generalization to new natural language instructions. We test the generalization at two levels. First, we replace words in a known task instruction with novel but similar words. For example, “move sth up” is replaced with “move sth higher” or “put sth higher”. As shown in Table II, these new instructions reach comparable task performance. The performance of generalization decreases the more novel words are in the instruction.

Second, we try to compose tasks. For example, combining “pull sth from left to right” and “move sth up” becomes “pull sth from left to right and move sth up”. Qualitatively, the policy network seems to linearly interpolate between the two tasks. This can be observed in the supplementary video. We evaluate the motion to be successful if it satisfies the success conditions in both tasks. However, the success rate drops by about half, which may not be surprising because we do not specifically train the multi-task network to be able to compose multiple tasks. This is a potential direction for future work.

Task Instruction	Success Rate
Known: Move sth up	88%
Unseen: Move sth higher	81%
Unseen: Put sth higher	79%
Unseen: Put an object higher	43%
Known: Move sth down	83%
Unseen: Move sth lower	83%
Unseen: Put sth lower	71%
Unseen: Put an object lower	34%
Known: Pull sth from left to right	93%
Unseen: Pull sth from left to right and move sth up	45%

TABLE II: Comparison of performance between unseen and related known instructions.

VI. Conclusion

In this paper, we approached the problem of learning a multi-task policy that links natural language instructions to motor skills. Specifically, we propose to learn from demonstrations that are provided in the form of videos of human manipulation actions. Our framework used a two-stage process. In the first stage, we trained single-task policies through a reinforcement learning algorithm where the reward is provided by a video-based activity classifier. This classifier scored how much the robot executing the current policy appears to perform the specified task. In a second stage, we then proceeded to train one multi-task policy by letting it imitate the single-task policies. In extensive simulated experiments, we showed how our policy can learn to perform a large number of the 78 complex manipulation tasks it was trained on. In ablation studies, we also motivated our choice of using a video- rather than an image-based classifier. Furthermore, we demonstrated how parameterizing the action through end effector goal pose *and* forces per time step enables the robot to learn a larger number of the 78 tasks than when only using the goal. Finally, we showed in qualitative examples how the multi-task policy generalizes to novel natural language instructions that are similar to the ones it was trained on.

Our approach opens exciting new future directions. For example, we would like to explore different algorithms for multi-task reinforcement learning as well as meta-learning to learn new tasks faster [53]. Furthermore, in this current approach, we have only learned to perform basic manipulation skills. However, for more complex manipulation tasks, we may want to compose or sequence skills which may require integrating task and motion planners similar to [19, 46]. Currently, we are only learning tasks on the coarsest level of natural language instructions that is provided by *Sth Sth* [13]. However, some actions are qualified by the kind of object they are applied to. For example, ”closing a bottle” requires a different motion from ”closing a drawer”. Expanding manipulation concept learning to these more complex instructions is also an interesting future direction. And finally, we would like to execute these learned manipulation concepts on a real robotic platform with real time closed-loop behavior.

References

[1] Peter Anderson, Qi Wu, Damien Teney, Jake Bruce, Mark Johnson, Niko Sünderhauf, Ian Reid, Stephen Gould, and

Anton van den Hengel. Vision-and-language navigation: Interpreting visually-grounded navigation instructions in real environments. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3674–3683, 2018.

[2] Jacob Andreas, Dan Klein, and Sergey Levine. Modular multitask reinforcement learning with policy sketches. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 166–175. JMLR. org, 2017.

[3] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57 (5):469–483, 2009.

[4] Dzmitry Bahdanau, Felix Hill, Jan Leike, Edward Hughes, Pushmeet Kohli, and Edward Grefenstette. Learning to follow language instructions with adversarial reward induction. *arXiv preprint arXiv:1806.01946*, 2018.

[5] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016–2019.

[6] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2169–2176. IEEE, 2017.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[9] Hao Fang, Saurabh Gupta, Forrest Iandola, Rupesh K Srivastava, Li Deng, Piotr Dollár, Jianfeng Gao, Xiaodong He, Margaret Mitchell, John C Platt, et al. From captions to visual concepts and back. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1473–1482, 2015.

[10] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. In *Advances in Neural Information Processing Systems*, pages 8538–8547, 2018.

[11] Scott Fujimoto, Herke Van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. *arXiv preprint arXiv:1802.09477*, 2018.

[12] Andrej Gams, Aleš Ude, Jun Morimoto, et al. Deep encoder-decoder networks for mapping raw images to dynamic movement primitives. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–6. IEEE, 2018.

[13] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fründ, Peter Yianilos, Moritz Mueller-Freitag, Florian Hoppe, Christian Thurau, Ingo Bax, and Roland Memisevic. The "something something" video database for learning and evaluating visual common sense. *CoRR*, abs/1706.04261, 2017. URL https://20bn.com/datasets/something-something.

[14] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning. *Conference on Robot Learning (CoRL)*, 2019.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[16] Auke Jan Ijspeert, Jun Nakanishi, and Stefan Schaal. Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*, volume 2, pages 1398–1403. IEEE, 2002.

[17] Yiding Jiang, Shixiang Gu, Kelvin Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. *arXiv preprint arXiv:1906.07343*, 2019.

[18] Yiding Jiang, Shixiang Shane Gu, Kevin P Murphy, and Chelsea Finn. Language as an abstraction for hierarchical deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 9414–9426, 2019.

[19] Leslie Pack Kaelbling and Tomás Lozano-Pérez. Hierarchical task and motion planning in the now. In *Proceedings of the 1st AAAI Conference on Bridging the Gap Between Task and Motion Planning*, AAAIWS'10-01, page 33–42. AAAI Press, 2010.

[20] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018.

[21] S Mohammad Khansari-Zadeh and Aude Billard. Learning stable nonlinear dynamical systems with gaussian mixture models. *IEEE Transactions on Robotics*, 27(5): 943–957, 2011.

[22] Oussama Khatib. A unified approach for motion and force control of robot manipulators: The operational space formulation. *IEEE Journal on Robotics and Automation*, 3(1), 1987.

[23] Tomas Kulvicius, KeJun Ning, Minija Tamosiunaite, and Florentin Worgötter. Joining movement sequences: Modified dynamic movement primitives for robotics applications exemplified on handwriting. *IEEE Transactions on Robotics*, 28(1):145–157, 2011.

[24] John Langford and Tong Zhang. The epoch-greedy algorithm for contextual multi-armed bandits. In *Proceedings of the 20th International Conference on Neural Informa-*

tion Processing Systems, NIPS'07, page 817–824, Red Hook, NY, USA, 2007. Curran Associates Inc. ISBN 9781605603520.

[25] Miguel Lázaro-Gredilla, Dianhuan Lin, J Swaroop Guntupalli, and Dileep George. Beyond imitation: Zero-shot task transfer on robots by learning concepts as cognitive programs. *arXiv preprint arXiv:1812.02788*, 2018.

[26] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

[27] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015.

[28] Eric Margolis and Stephen Laurence, editors. *Concepts: Core readings.* The MIT Press, Cambridge, MA, US, 1999. ISBN 0-262-13353-9 (Hardcover); 0-262-63193-8 (Paperback).

[29] Takamitsu Matsubara, Sang-Ho Hyon, and Jun Morimoto. Learning stylistic dynamic movement primitives from multiple demonstrations. In *2010 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1277–1283. IEEE, 2010.

[30] F. Meier, E. Theodorou, F. Stulp, and S. Schaal. Movement segmentation using a primitive library. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3407–3412, 2011.

[31] Dipendra Misra, Andrew Bennett, Valts Blukis, Eyvind Niklasson, Max Shatkhin, and Yoav Artzi. Mapping instructions to actions in 3d environments with visual goal prediction. *arXiv preprint arXiv:1809.00786*, 2018.

[32] Alexandros Paraschos, Christian Daniel, Jan R Peters, and Gerhard Neumann. Probabilistic movement primitives. In *Advances in neural information processing systems*, pages 2616–2624, 2013.

[33] Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

[34] Peter Pastor, Heiko Hoffmann, Tamim Asfour, and Stefan Schaal. Learning and generalization of motor skills by learning from demonstration. In *2009 IEEE International Conference on Robotics and Automation*, pages 763–768. IEEE, 2009.

[35] Reuven Y. Rubinstein and Dirk P. Kroese. *The Cross Entropy Method: A Unified Approach To Combinatorial Optimization, Monte-Carlo Simulation (Information Science and Statistics)*. Springer-Verlag, Berlin, Heidelberg, 2004. ISBN 038721240X.

[36] Andrei A Rusu, Sergio Gomez Colmenarejo, Caglar Gulcehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. Policy distillation. *arXiv preprint arXiv:1511.06295*, 2015.

[37] Ozan Sener and Vladlen Koltun. Multi-task learning as multi-objective optimization. In *Advances in Neural Information Processing Systems*, pages 527–538, 2018.

[38] Lin Shao, Parth Shah, Vikranth Dwaracherla, and Jeannette Bohg. Motion-based object segmentation based on dense rgb-d scene flow. *IEEE Robotics and Automation Letters*, 3(4):3797–3804, 2018.

[39] Tianmin Shu, Caiming Xiong, and Richard Socher. Hierarchical and interpretable skill acquisition in multi-task reinforcement learning. *arXiv preprint arXiv:1712.07294*, 2017.

[40] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[41] Avi Singh, Larry Yang, Kristian Hartikainen, Chelsea Finn, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. *arXiv preprint arXiv:1904.07854*, 2019.

[42] Vladimir M Sloutsky. From Perceptual Categories to Concepts: What Develops? *Cognitive science*, 34(7): 1244–1286, sep 2010. ISSN 1551-6709. doi: 10.1111/j.1551-6709.2010.01129.x.

[43] Yuuya Sugita and Jun Tani. Learning semantic combinatoriality from the interaction between linguistic and behavioral processes. *Adaptive behavior*, 13(1):33–52, 2005.

[44] Yee Teh, Victor Bapst, Wojciech M Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. Distral: Robust multitask reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 4496–4506, 2017.

[45] Chen Tessler, Shahar Givony, Tom Zahavy, Daniel J Mankowitz, and Shie Mannor. A deep hierarchical approach to lifelong learning in minecraft. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

[46] Marc Toussaint, Kelsey Allen, Kevin Smith, and Joshua Tenenbaum. Differentiable physics and stable modes for tool-use and manipulation planning. In *Proceedings of Robotics: Science and Systems*, Pittsburgh, Pennsylvania, June 2018. doi: 10.15607/RSS.2018.XIV.044.

[47] Du Tran, Lubomir Bourdev, Rob Fergus, Lorenzo Torresani, and Manohar Paluri. Learning spatiotemporal features with 3d convolutional networks. In *Proceedings of the IEEE international conference on computer vision*, pages 4489–4497, 2015.

[48] Aleš Ude, Andrej Gams, Tamim Asfour, and Jun Morimoto. Task-specific generalization of discrete and periodic dynamic movement primitives. *IEEE Transactions on Robotics*, 26(5):800–815, 2010.

[49] Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3156–3164, 2015.

[50] Xin Wang, Wenhan Xiong, Hongmin Wang, and William Yang Wang. Look before you leap: Bridging model-free and model-based reinforcement learning for planned-ahead vision-and-language navigation. In *Proceedings of*

the European Conference on Computer Vision (ECCV), pages 37–53, 2018.

[51] Xin Wang, Qiuyuan Huang, Asli Celikyilmaz, Jianfeng Gao, Dinghan Shen, Yuan-Fang Wang, William Yang Wang, and Lei Zhang. Reinforced cross-modal matching and self-supervised imitation learning for vision-language navigation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 6629–6638, 2019.

[52] Licheng Yu, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg. Modeling context in referring expressions. In *European Conference on Computer Vision*, pages 69–85. Springer, 2016.

[53] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. *arXiv preprint arXiv:1910.10897*, 2019.