One-shot Visual Imitation via Attributed Waypoints and Demonstration Augmentation

Matthew Chang[1] and Saurabh Gupta[1]

***Abstract*— In this paper, we analyze the behavior of existing techniques and design new solutions for the problem of one-shot visual imitation. In this setting, an agent must solve a novel instance of a novel task given just a single visual demonstration. Our analysis reveals that current methods fall short because of three errors: the DAgger problem arising from purely offline training, last centimeter errors in interacting with objects, and mis-fitting to the task context rather than to the actual task. This motivates the design of our modular approach where we a) separate out task inference (what to do) from task execution (how to do it), and b) develop data augmentation and generation techniques to mitigate mis-fitting. The former allows us to leverage hand-crafted motor primitives for task execution which side-steps the DAgger problem and last centimeter errors, while the latter gets the model to focus on the task rather than the task context. Our model gets 100% and 48% success rates on two recent benchmarks, improving upon the current state-of-the-art by absolute 90% and 20% respectively.** [1]

I. Introduction

Consider a single video, demonstrating the task depicted in Figure 1 (left). Given just this input, as humans we can reliably execute the demonstrated task in the novel situation shown in Figure 1 (right). This is in spite of the differences in the task instance (location of relevant objects are different from where they were in the demonstration), embodiment (*e.g.* robot hand in demonstration *vs.* human hand), and the large ambiguity in what precisely the task was (was it to move the hand through those locations or was it to move the object). In this paper, we seek to imbue robotic agents with a similar capability: given a *single visual demonstration* of a *novel task*, the robot should execute the demonstrated task on a *novel instance* of the task. We refer to this problem as *one-shot visual imitation*.

While humans are adept at this form of one-shot visual imitation, machine performance in this setting lacks considerably. For instance, the recent method from Dasari *et al.* [1] obtains a 10% success rate on a harder version of their pick-and-place task-set, and 28% on a one shot visual imitation benchmark constructed using Meta-world [2]. In this paper, we investigate what causes recent methods to underperform and develop algorithms to bridge this performance gap.

We start by analyzing the behavior of current methods. Current works on this problem [1], [3], [4] cast it as a conditional policy learning problem (*i.e.* predict the *next action* conditioned on the demonstration and the execution so far) using meta-learning [3] or expressive neural network models [1], [4]. Models are trained on *offline* datasets of video demonstrations paired with expert executions. This immediately reveals two issues that hinder the performance of these past works. Purely offline and non-interactive training causes the learned policies to suffer from a form of distribution shift known as the *DAgger problem*, (going off-distribution due to compounding errors while imitating long-horizon behaviors) and near misses while executing fine motor control, or *last centimeter errors* (prior work has shown that learning generalizable policies for fine-motor control requires specialized architecture or thousands of online samples).

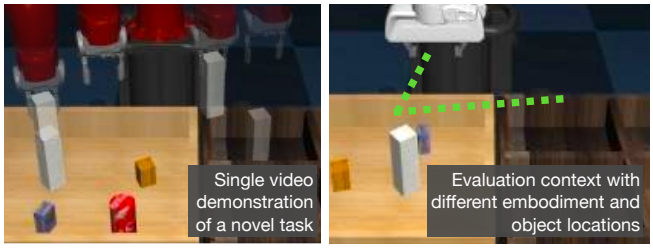

Fig. 1. The One-shot Visual Imitation Problem.

When we try to extend current methods to more diverse collection of tasks, a third, more subtle *mis-fitting* issue comes to light. As tasks are often contextual (*i.e.* one only interacts with a given object in a limited number of ways), current models tend to make predictions based on the objects in the scene rather than the motion depicted in the demonstration. This causes them to generalize poorly to novel tasks.

These insights motivate the design of our method. To circumvent the first two problems, we employ a hierarchical and modular approach that separates out the task execution (*how* to do it) from task inference (*what* to do). This separation enables us to use robust and high-performing, hand-crafted motor primitives for task execution, while the use of learning for task inference allows the system to interpret the intent depicted in the provided demonstration, and synthesize a solution for the novel instance at test-time. More concretely, given the video demonstration and just a single image of the current scene, our learned model predicts a sequence of *attributed waypoints* that outline a trajectory to achieve the task. These *attributed waypoints* represent the 3D motion of the arm, along with additional attributes of the robot's state (such as "an object is in the gripper") at those waypoints. The predicted attributed waypoints are achieved using motor primitives (based on kinematic planning or classical grasping primitives using depth images from hand-in-eye cameras).

While this seemingly simple model works well for pick-and-place tasks (achieving 100% success rate on the task-set

[1]University of Illinois Urbana-Champaign, Illinois, USA. Emails: {mc48,saurabhg}@illinois.edu.

[1]Project website with additional details: https://matthewchang.github.io/awda_site/

from Dasari *et al.* [1]), it still underperforms on the diverse tasks in Meta-world [2], due to the mis-fitting issue described above. To mitigate this, we propose novel demonstration augmentation schemes that generate training samples to break the correlation between tasks and their contexts.

We evaluate our proposed method on 4 benchmarks, representing a wide range of diverse tasks in simulation, and evaluation on real-world data [5]. In comparisons against 3 past methods (DAML [6], T-OSVI [1], and MOSAIC [4]) our model achieves strong results, surpassing all baselines. Notably, our method makes large improvements on two benchmarks, reaching success rates of 100% and 48%. These represent absolute improvements of 90% and 20% respectively, over the current state-of-the-art.

II. Related Work

Learning from demonstrations in robotics has taken many different forms over the years. One such setting is behavior cloning (BC), in which one is given many paired sensor and action trajectories for a single task, and the goal is to obtain a policy for this task [7]. Purely training on offline datasets of expert behavior is known to suffer from compounding errors at execution time, motivating improvements like DAgger [8]. Recent approaches to this problem have considered BC using only one trajectory with action labels [9], [10], and using meta-learning to adapt to novel tasks at test time [3].

One-shot visual imitation takes this problem a step further, where only a video (*i.e.* with no associated actions) of the expert's execution is available [3], [6], [11]. Researchers have explored many different variants, with demonstrations differing in embodiment [6], [10], viewpoint [12], or using natural language [5], [13]. Researchers have also pursued many different solutions: conditioning on task embeddings [14], [15], using meta-learning [6], [10], predicting sub-goals [11], [12], using expressive transformer architectures [1], [4], and contrastive training of visual features [4]. Huang *et al.* [16] and Sharma *et al.* [12] follow a hierarchical design similar to ours for one-shot visual imitation. However, our formulation can deal with tasks involving arbitrary objects and motions, unlike the method from [16] which only operates within a pre-defined set of discrete symbols and motions. [12] synthesizes images to use as sub-goals, while our use of generalized waypoints sidesteps the need for image generation, which can be challenging for novel objects in complex environments.

We distinguish from another related line of work on inverse reinforcement learning (IRL) [17]–[22]. IRL assumes interactive access to the underlying environment to learn policies, which our setting does not.

Hierarchical policies have been found to useful in many settings, indoor navigation [23]–[25], self-driving [26], drone control [27], [28], and manipulation [29]–[34] among many others. In reinforcement learning settings, motion primitives have been incorporated as additional actions to speed up learning [31], [32], [35]. Instead, our work develops techniques to use motor primitives for one-shot visual imitation. While we found hand-designed primitives effective in our work, our method is agnostic to the form of motor primitives, and could benefit from the many recent works on discovering motor primitives from diverse trajectories [36]–[40].

Data augmentation techniques have been found to be effective in improving the generalization of learned models [41]. They have also been effective at improving generalization in robot learning, *e.g.* when learning policies via RL [42], or for pre-training representations [4], [43]. Our work also employs data augmentation techniques, inspired by mixup [44]–[46] for improving generalization. However, the specific form of overfitting in our one-shot visual imitation problem motivates the need for *asymmetrical* mixing of samples to decorrelate tasks from task contexts.

III. Diagnosing Errors Made by Current One-shot Visual Imitation Methods

The problem we are interested in is that of one-shot visual imitation. At test time, our agent will be given a video demonstration of a task not seen during training, and must perform the depicted task with no additional experience in the environment. Note that, the environment configuration may be different from that depicted in the example video, but the overall semantic task will be the same.

Samples in one-shot visual imitation learning datasets consist of: a) the video demonstration $\mathbf{v}$; and b) a robotic trajectory, $\{(o_1, s_1, a_1), \ldots\}$ that conducts the same task in a potentially different situation. These video demonstrations are not the same trajectory as the robotic trajectory and may differ in embodiment, or solution method, but they must be solving the same high-level task. In fact, the pairing of video demonstrations to robotic trajectories is what defines the notion of a task for the model being trained.

This is commonly cast as a supervised learning problem [1], [3], [4], [6], where a model is learned on demonstration-trajectory pairs, to predict the action at a given timestep, conditioned on the demonstration, and previous frames, $\pi(a_t|\mathbf{v}, o_{1:t}, s_{1:t})$. However, this approach can lead to undesirable behaviors on novel tasks. In this section, we characterize the failure modes of T-OSVI, the transformer-based per-time-step action prediction model from Dasari *et al.* [1] as a representative recent method. Specifically, we highlight 3 different failure modes that arise in this standard approach: the DAgger problem arising from purely offline training, last centimeter errors in interacting with objects, and mis-fitting to the task context rather than the depicted task. A summary of the terms and notation used in this paper can be found in Table III.

Experimental Setting. We consider a harder version of the 4 object and 4 bin pick-and-place task family proposed in [1] (visualized in Figure 1), that has been modified to hold out tasks as opposed to task instances as originally done in [1]. That is, of the 16 possible tasks (picking one of the four objects and placing it in one of the four bins), we use 14 for training and hold out 2 for testing. In this modified setting, the success rate for T-OSVI [1] drops to 10% from the 88% reported in their paper. In this setting, we identify two consistent failure modes: a) failure to reliably reach the target object (about 88% trials) and often times (35% of

TABLE I
DEFINITIONS OF PHRASES AND TERMS.

Term	Definition
Task	High level objective for the agent, *i.e.* open the window, or move the white block to the first bin
Novel Task	A task which does not appear in the training dataset
Task Instance	A unique configuration of objects in the scene in which a task must be performed
Task Context	The objects and elements in the scene in which a task is performed. One task context may admit many different tasks, i.e. the same blocks could be pushed, pulled, stacked etc.
Demonstration ($\mathbf{v}$)	A video containing only RGB images of a task being successfully performed. This may differ from the test time environment in agent embodiment, and will feature a different task instance.
Robot Trajectory $\{(o_1, s_1, a_1), \ldots\}$	A sequence of RGB images $\{o_1, \ldots, o_T\}$ along with robot states $\{s_1, \ldots, s_T\}$ (*e.g.* joint angles, joint velocities) and commands $\{a_1, \ldots, a_T\}$ (*e.g.* commanded torques, or end-effector destinations)
Attributed Waypoint	A point in $3 + k$ space indicating a point in $\mathcal{R}^3$ and the presence of up to k attributes. See Section IV-A
AWDA	Attributed Waypoints and Demonstration Augmentation (our method)
ADM	Asymmetric Demonstration Mixup, see Section IV-B
T-OSVI	Transformers for one-shot visual imitation [1]
TS	Trajectory Synthesis: Generating additional training samples featuring free-space motion, see Section IV-B

trials) reaching a non-target object due to what we believe to be a version of the DAgger problem, and b) near misses in grasping the object (about 10% of all trials in which it reached any object).

DAgger problem. T-OSVI can be viewed as a conditional policy of the form $\pi(a_t|\mathbf{v}, o_{1:t}, s_{1:t})$, trained through behavior cloning on an offline dataset of expert executions. Behavior cloning on expert data is known to suffer from poor execution performance due to compounding errors [8]. While this may explain the low reaching success rate for the target object, it doesn't explain the relatively high rate with which the policy reaches and attempts to grasp a non-target object.

Our belief is that this is a task execution error. The policy is trained with memory of its execution over the last 6 frames, and often it relies more on these recent execution frames, than the demonstration. Consequently, we find that if the agent makes small errors early during execution, subsequent behavior is more in line with the current execution, at the cost of being inconsistent with the given demonstration.

We empirically verify this by keeping the demonstration fixed but *guiding* the policy at test time towards the target object (positive guidance) or towards a distractor non-target object (negative guidance), by taking steps with an oracle policy. Even 1 step of guidance drastically improves the reaching rate from 12% to 58% $\pm$ 2% for the target object, and from 27% to 50% $\pm$ 2% for the distractor object. Note that it takes on average 12 steps to reach the object, so 1 step of guidance is not much. The increase in success rate for *both* target and distractor objects reveals the preference of the policy towards past execution frames over the demonstration.

Last centimeter errors in grasping. Next, we discuss the second substantial error mode of T-OSVI on this task. While 12% of executions reach the correct object, only 10.5% of executions successfully lift the object, meaning 10% of attempted grasps fail. This is because the gripper grasps near the object, but misses, or acquires an unstable grasp. This is not surprising as we are attempting to learn a grasping policy from as few as 1400 training samples. Past works have shown that without specialized architectures or sensing, many thousands of trials are necessary to learn grasping policies that generalize [47], [48]. Other recent works [4], [5] also noted these fine-grained errors in one-shot visual imitation.

Mis-fitting to Task Context. In the harder, more diverse set of 50 tasks from Meta-world [2], a new failure mode of one-shot visual imitation methods arises. We find that, if the novel evaluation task involves objects that are visually similar to those seen in training tasks, models trained with T-OSVI perform the motion from the training task, not what is seen in the demonstration (visualized in Figure 2). We believe that this is because the model is predicting actions

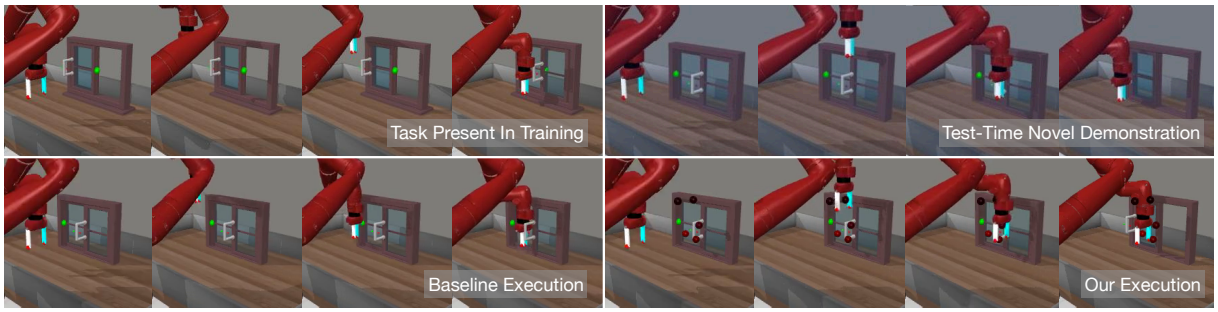

Fig. 2. **Example of mis-fitting in Meta-world (Section III)**: During training there is a task depicting the window being closed (top left). When presented with a novel task demonstration, opening the window (top right), action-prediction methods repeat the motion on the most similar training setting, trying to close the already closed window (bottom left). Our method successfully opens the window (bottom right). Predicted waypoints (red dots) correctly move to the right side of the handle and push left.

based on the task context (objects visible in the scene) as opposed to the task depicted in the demonstration.

Tasks are contextual, *i.e.* there are really only a few different things that one can do with a given object (*e.g.* opening a closed door). Besides, collecting training data for one-shot visual imitation is challenging, as it requires paired demonstrations and trajectories. Thus, training datasets are small and don't showcase diverse interactions with the manipulated objects. Models can easily satisfy the training objective by fitting to the task context and ignoring the motion depicted in the demonstration. This causes problems when we seek to imitate a novel task that bears visual similarity to a training task. As depicted in Figure 2, existing methods will attempt to perform the task seen in training instead of that depicted in the demonstration.

The analysis of these three failure modes motivates the design of our method, as detailed in the next section. We address task execution errors (*i.e.* the DAgger problem and the last centimeter problem) through the use of hand-crafted motor primitives, and present data augmentation strategies that break the correlation between tasks and task contexts, mitigating mis-fitting.

IV. Visual Imitation via Attributed Waypoints and Demonstration Augmentation

Our approach, AWDA, is a hierarchical and modular approach that separates out task inference and task execution. The task inference module takes the given demonstration video and a single image of the scene (depicting an instance of the task, different from that in the demonstration) and outputs the full execution plan, expressed as a sequence of *attributed waypoints*. Task execution happens simply by invoking the appropriate motor primitive to convey the robot end effector between each consecutive pair of predicted waypoints.

A. Task Inference and Execution via Attributed Waypoints and Motor Primitives

Attributed Waypoints. Our task inference and execution modules are interfaced via attributed waypoints. Typical waypoints used in robotics (*e.g.* for navigation [23]) only capture the 3D (or 6D) pose of the robot end-effector. This is quite restrictive for manipulation tasks, as purely kinematic guidance of the end-effector will not be able to interact with objects *e.g.* to pick them up or to exert forces on them. We overcome this limitation by assigning additional *attributes* to each waypoint, *e.g.* is an object in the end-effector, or is the end-effector experiencing force in a particular direction. We consider attributed waypoints to be $3+k$ dimensional, where k is the number of additional attributes associated with each 3D waypoint. Attributed waypoints are a powerful tool for expressing solutions to kinematic tasks. For instance, just 1 single attribute, of whether there is an object in the gripper or not, allows us to express all 50 tasks in the Meta-world task-set as a sequence of these 4D waypoints (3D for end-effector location and 1D for *"is there an object in the gripper"*). We will use this attribute as a running example for explanation, but other attributes could be added.

Motor Primitives. Given a pair of attributed waypoints, our method uses motor primitives to convey the robot between pairs of attributed waypoints. While conveying the end-effector between 3D waypoints in space is well understood (inverse kinematics and motion planning), moving between our proposed attributed waypoints is more involved, as it can involve a change in "attributes" along the way. Thankfully, changes in attributes correspond to well-studied basic skills in robotics literature. For instance, using the same 4D grasping example as above, going from waypoint $[\mathbf{p}; \text{false}]$ to $[\mathbf{q}; \text{true}]$ involves grasping an object near location $\mathbf{p}$ and taking it to location $\mathbf{q}$; while going from $[\mathbf{q}; \text{true}]$ to $[\mathbf{p}; \text{false}]$ corresponds to releasing the currently held object and then going to location $\mathbf{p}$. In general, for k attributes, this corresponds to 2^{k+1} motor primitives. Our method is agnostic to the exact implementation of motor primitives. For our experiments, we found that hand-crafted primitives were sufficient to solve the pick-and-place task-set from [1] and all 50 Meta-world tasks [2]. We implemented 4 hand-crafted primitives: a) free space motion without any object in hand, b) grasping an object, c) dropping an object, and d) free space motion with an object in hand; using eye-in-hand depth cameras.

Training the Model to Output Augmented Waypoints. Augmented waypoints and corresponding motor primitives let us express manipulation tasks as a sequence of waypoints. We next describe how we train the task inference module to predict these augmented waypoints from a given demon-

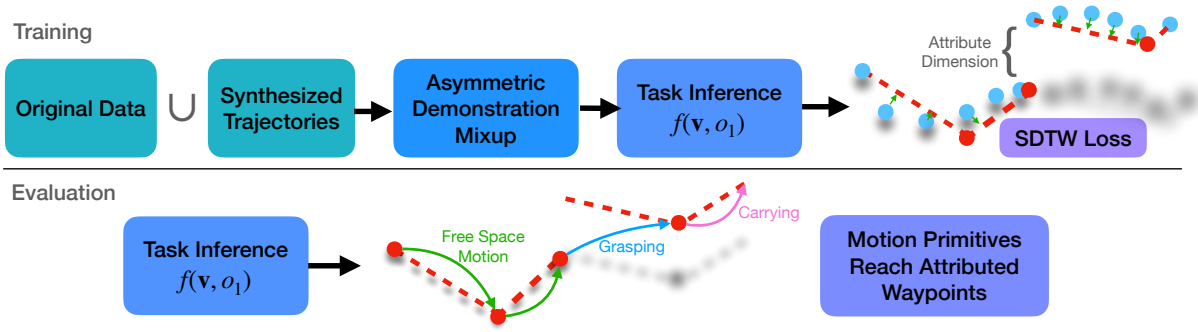

Fig. 3. **Bottom:** AWDA is a modular approach for one-shot visual imitation that separates task inference and task execution. Task inference function $f(\mathbf{v}, o_1)$ predicts a sequence of *attributed* waypoints (red points) that are achieved using hand-defined motion primitives (colored solid lines). **Top Right:** $f(\mathbf{v}, o_1)$ learns to predict attributed waypoints by aligning them with ground-truth attributed trajectories using SDTW (Section IV-A). **Top Left:** To prevent overfitting of $f(\mathbf{v}, o_1)$ to task contexts, we synthesize additional demonstrations and employ asymmetric demonstration miuxup (Section IV-B).

stration, and a single image of the novel task instance. Our task inference model f takes as input, a demonstration $\mathbf{v}$ and instance image o_1, and outputs n attributed waypoints $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$, each waypoint being $k+3$ dimensional. Supervision for these waypoint predictions is derived from the trajectories in the dataset as follows. We process the given robotic trajectory, $\{(o_1, s_1, a_1), \ldots\}$, into 3D end-effector locations using forward kinematics. We also assign to each time step, the appropriate attributes that the agent experiences in that frame. These attributes are not labeled by hand, but rather mined automatically from the robot state s_t and a_t provided in the robotic trajectories. For example, we label that an object has been grasped in a robotic trajectory when the commanded action is to close the gripper, but the gripper jaws do not close. This gives us end-effector's *attributed trajectory*: $\mathbf{t} = \{\mathbf{t}_1, \ldots, \mathbf{t}_T\}$, $t_i \in \mathbb{R}^{k+3}$. We derive supervision for waypoints $\{\mathbf{w}_1, \ldots, \mathbf{w}_n\}$ by constructing a trajectory $\hat{\mathbf{t}}$ by linearly interpolating $\mathbf{w}$'s and comparing it to the ground truth attributed trajectory $\mathbf{t}$. We use soft dynamic time-warping (SDTW) [49] to compute the loss between the predicted and ground truth trajectory. Minimizing this objective aligns the predicted trajectory with the ground truth trajectory.

Testing. Given a novel video $\mathbf{v}$, and task instance, as observed in image o_1, we use the task inference model $f(\mathbf{v}, o_1)$ to predict attributed waypoints. The appropriate motor primitives are used to convey the robot from one predicted attributed waypoint to the next, until all waypoints are exhausted.

B. Demonstration Augmentation for Improved Task Inference

We next look at tackling the *mis-fitting* issue highlighted in Section III. This mis-fitting happens because of the strong correlation between the task and its context in the training data. We design two augmentation strategies that break this correlation by generating training samples with the same context but different task motion.

Asymmetric Demonstration Mixup. Our first strategy creates new training samples by mixing existing samples in the dataset. This is reminiscent of mixup [44] but has modifications to break the aforementioned correlation. Naively mixing samples as done in original mixup wouldn't break the correlation to aid out-of-distribution generalization. Instead, we leverage the temporal nature of video demonstrations to *asymmetrically* blend samples (depicted in Figure 4). Given a sample $(\mathbf{v}, o_1, \mathbf{t})$, we use another sample $(\tilde{\mathbf{v}}, \tilde{o}_1, \tilde{\mathbf{t}})$ to produce a new decorrelated sample by: a) blending all frames in $\mathbf{v}$ with the first frame of the video $\tilde{\mathbf{v}}$ to generate new video $\mathbf{v}'$, b) blending o_1 with $\tilde{o}_1$ to generate o_1', and c) retaining $\mathbf{t}$ as is. Specifically,

$$\mathbf{v}'_t = \alpha \mathbf{v}_t + (1-\alpha)\tilde{\mathbf{v}_0}, \quad o_1' = \alpha o_1 + (1-\alpha)\tilde{o_1}, \quad \mathbf{t}' = \mathbf{t}$$

We use a blending ratio $\alpha \sim [0.3, 1.0]$, biased towards retaining all of o_1 and $\mathbf{v}$, since the trajectory is always $\mathbf{t}$. This asymmetric blending, where one of the demonstrations is frozen in time while the other is moving, breaks the correlation between objects present in the scene and the task being conducted on them. f can't just look at o_1', but it has to track how the hand moves through in $\mathbf{v}'$ to make correct predictions. Including unaltered samples in training lets us use demonstrations and observations as is at test time.

Additional Demonstrations via Trajectory Synthesis. Additionally, we can break the correlation between tasks and task contexts by simply generating synthetic tasks involving free space motions for the robot, in various contexts. We do this by sampling a small number of points (1 to 3) uniformly at random within the agent's workspace and moving the end effector sequentially through these points using an inverse kinematics solver. Training samples are created by pairing each trajectory with itself, *i.e.* $\mathbf{v}$, o_1 and $\mathbf{t}$ all come from the same trajectory. To make correct predictions on these trajectories, the model must attend to the motion of the arm and ignore background elements, thus breaking the undesired correlations. Note that this is a simple data collection procedure that can be easily done in an unsupervised manner. We simply add in these additional samples for training, and find they boost performance, particularly for substantially out-of-distribution tasks.

We note that this procedure produces samples that are not driven by a semantically-meaningful, object-centered task. Thus, including these samples in training could also impact the model's ability to learn a meaningful prior over tasks. To mitigate this, we modify the final layer of the model to have two heads. One makes predictions for original samples in the dataset, while the other makes predictions for the synthesized

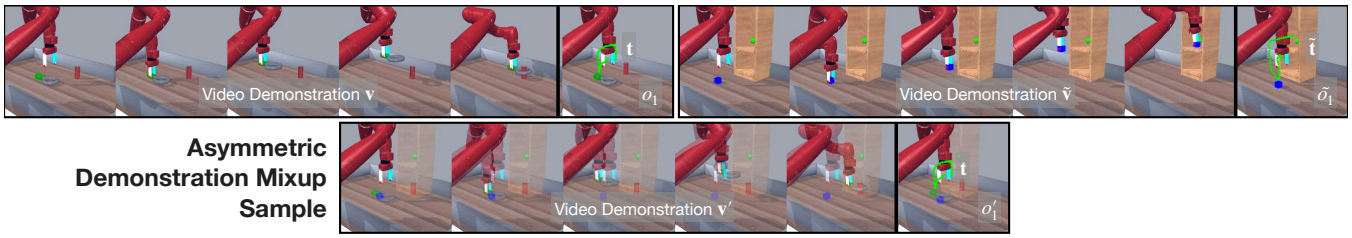

Fig. 4. **Asymmetric Demonstration Mixup (ADM) augmentation:** Two demonstration-trajectory pairs (row 1) are combined into one (row 2). The first frame of demonstration 2 is blended into every frame of demonstration 1. The end effector trajectory that serves as supervision for training on the combined sample is entirely from trajectory 1.

trajectories. This nudges the overall network to look at the motion of the hand while also letting the last layer learn the necessary priors from the task driven samples in the dataset.

V. Experiments

We design and conduct experiments to demonstrate the effectiveness of our proposed method with respect to prior work, and evaluate our various design choices.

Tasks, Environments, and Datasets. We conduct experiments on 4 datasets: a) **Pick-and-place** task-set from [1] (shown in Figure 1) but modified to hold 2 of 16 possible tasks as novel testing tasks; b) **Meta-world** task-set [2] (sample observations shown in Figure 2) where we hold out 4 of 50 tasks as novel testing tasks; c) **MOSAIC** task-set [4] containing 6 tasks, evaluating performance on each task with a model trained only using demonstrations from the other tasks; and d) **BC-Z** dataset [5] that has 17213 *real-world* trajectories (sample images in Figure 5) spanning 90 tasks of which we hold out 5 for testing.

Pick-and-place and MOSAIC use different embodiments for demonstration and execution (Sawyer and Panda respectively). Meta-world and BC-Z use the same embodiment. We conduct interactive evaluation of the learned policies on Pick-and-place, Meta-world and MOSAIC task-sets and report success rate. For BC-Z, we do offline evaluation and report the accuracy of predicted trajectories on a held-out validation set. We note that as all tasks are set up in the same environment for the Pick-and-place task set, so it doesn't suffer from correlations between tasks and task contexts, or the mis-fitting error described in Section III. However, different tasks in Meta-world and MOSAIC involve different objects making them suffer from the mis-fitting error.

Implementation Details. We follow [1] to construct data for training. We collect 100 successful trajectories with each robot using hand-defined expert controllers and arbitrarily pair them up to construct 10K training samples per task. We train our models for 500K iterations. Following [4], we report the success rate on held-out tasks, averaged over 5 snapshots from the end of training. Our attributed waypoints use the *"is object in hand"* attribute. This leads to 4 motor primitives: free space motion without an object, moving an object, grasping a nearby object, and releasing the object. We implement the first two using inverse kinematics and motion planning; the third is implemented by analyzing depth images to identify objects and centering the gripper to grasp the nearest object; for the fourth, we just open the gripper.

The neural network design uses the same feature extraction process as T-OSVI [1]. We extract image features using ResNet-18, which remain spatial and have a sinusoidal positional encoding added, before being processed by a transformer module. Finally, the temporally processed features are projected down into waypoints by two separate heads, one to predict waypoints for task-driven trajectories, and one for trajectories synthesized as described in Section IV-B.

Results. We report results on the Pick-and-place and Meta-world task sets in Table II. We break down results on Meta-world into two splits based on the similarity of the novel task to tasks seen in training: Meta-world [easy] (*Button-Press-V2*, *Pick-Place-Wall-V2*, which differ from training tasks *Buttom-Press-Wall-V2* and *Pick-Place-V2* only due to presence/absence of distractor), and Meta-world [hard] (*Window-Open-V2*, *Door-Unlock-V2*, require categorically different solutions than training tasks). The results on the MOSAIC benchmarks are presented in Table III. Each column reports the performance on the indicated held-out task. The models for MOSAIC experiments are trained on all MOSAIC tasks except the held-out task. The column *All* reports the mean performance across all held-out task experiments. We summarize our key takeaways below.

• **AWDA outperforms prior work by a large margin.** Our full system (denoted *Full-1*) completely solves the Pick-and-place task (improving upon the 10% obtained by T-OSVI, 1% by DAML), and obtains 48% for Meta-world [all] *vs.* 28% for T-OSVI, 6% for DAML, while quadrupling the performance on the hard tasks 30% *vs.* 7% for T-OSVI. Performance gains are maintained even if we omit synthesized trajectory data altogether (denoted *no AD*), or when using data from other datasets instead (denoted *Full-2*). On the MOSAIC tasks (Table III), we match or outperform the current state-of-the-art for this benchmark [4] on all tasks except for one, yielding superior overall performance (7% *vs.* 4%).

• **Attributed waypoints with motor primitives eliminate all errors on Pick-and-place.** Our models without asymmetric demo mixup (*no ADM*), or without additional data (*no AD*), or without both (*only waypoints*) obtain close to perfect performance on the Pick-and-place task. This demonstrates the effectiveness of our proposed modular policy architecture. It also boosts performance on Meta-world [all] by an absolute 29% from 19% with *no waypoints vs.* 48% with (*Full-1*).

• **Additional data via trajectory synthesis or from other datasets helps improve generalization**. Using additional data via trajectory synthesis (*Full-1*) or from other datasets

TABLE II

SUCCESS RATES ON HELD-OUT TASKS ON THE PICK-AND-PLACE AND META-WORLD BENCHMARKS

	DAML [6]	**T-OSVI** [1]	**Full-1**	**Full-2**	**single head**	**no AD**	**no ADM**	**no way points**	**only waypoints**
Asymm. Demo Mixup (ADM)?			✓	✓	✓	✓	✗	✓	✗
Additional Data (AD) Source?			TS	BC-Z	TS	✗	TS	TS	✗
Waypoints?			✓	✓	✓	✓	✓	✗	✓
Pick-and-place	0.01	0.10	**1.00**	**1.00**	0.99	1.00	0.98	0.01	0.98
Meta-world [easy]	0.04	0.50	0.66	**0.73**	0.33	0.74	0.03	0.33	0.14
Meta-world [hard]	0.08	0.07	**0.30**	0.17	0.29	0.11	0.19	0.06	0.02
Meta-world [all]	0.06	0.28	**0.48**	0.45	0.31	0.42	0.11	0.19	0.08

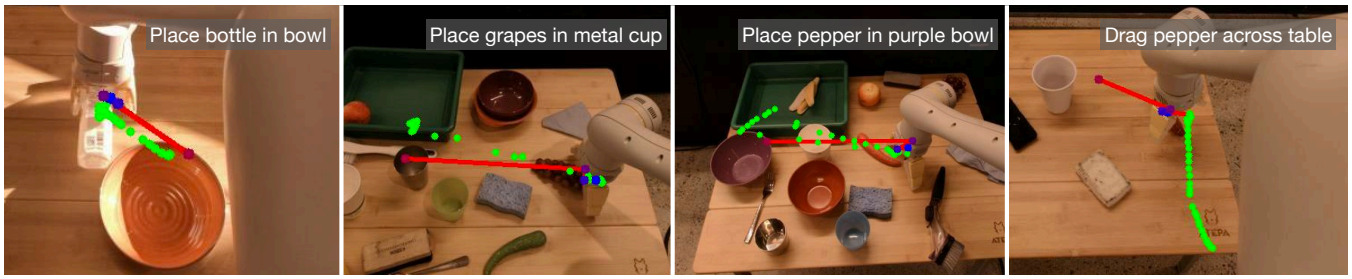

Fig. 5. We visualize 2D projections of 3D waypoints in blue, and the interpolated trajectory in red, as predicted by our model for 4 different held-out tasks (noted on top right of each image) from the BC-Z dataset [5]. Predicted trajectories match the ground truth trajectories (in green). Interestingly, for the *"drag pepper across table"* task, though our prediction does not match the specific ground truth, it is still consistent with the semantics of the depicted task.

TABLE III

SUCCESS RATES ON HELD-OUT TASKS FROM MOSAIC [4] TASK-SET

Task	Door	Drawer	Button	Blocks	B.B.	Nut A.	All
MOSAIC [4]	0.05	0.15	**0.05**	0	0	0	0.04
Full-1	**0.10**	**0.29**	0.01	0	0	**0.02**	**0.07**

(*Full-2*) improves upon not using any additional data (denoted *no AD*), particularly for the Meta-world [hard] tasks that require entirely novel motion at test time (11% for *no AD vs.* 30% for *Full-1* and 17% for *Full-2*). Furthermore, fitting this additional data through another head is crucial for maintaining performance on Meta-world [easy] tasks, which bear more similarity to training tasks: 66% for the two headed model *Full-1 vs.* 33% for the *single head* model.

• **Asymmetric demonstration mixup improves performance** beyond the standard image augmentations (random flip, crop, translation, color jitter, *etc.*) that are already in use for T-OSVI, DAML and all our models (*Full-1 vs. no ADM*).

• **AWDA gets good performance on real data from robots.** On the BC-z dataset [5], our method is able to predict the final interaction point (grasp, release, or reach point) to within 10 cm for $63\% \pm 4\%$ samples of *held-out* tasks. This clearly identifies what objects need to be interacted with. We expect appropriately designed motor primitives on physical robots will be able to successfully execute some of these tasks. Figure 5 shows some sample predictions.

VI. CONCLUSION AND FUTURE WORK

In this paper we analyzed the major failure modes of state-of-the-art action prediction methods for one-shot visual imitation. We find that they suffer from the DAgger problem, last centimeter errors, and mis-fitting to task contexts. Our proposed method, utilizing attributed waypoints and demonstration augmentation, is able to significantly boost success rates on existing benchmarks, even completely solving one.

As is, our system is limited to kinematic tasks, but could be expanded to reasoning about forces, given the proper motion primitives. While our motion primitives are closed-loop and account for slight changes in object locations, the high level plan cannot adjust to large changes in the scene after initial waypoint predictions. We leave this to future work.

ACKNOWLEDGMENT

This material is based upon work supported by NSF (IIS-2007035), DARPA (Machine Common Sense program), an Amazon Research Award, an NVidia Academic Hardware Grant, and the NCSA Delta System (supported by NSF OCI 2005572 and the State of Illinois)

REFERENCES

[1] Dasari, Sudeep and Gupta, Abhinav, "Transformers for one-shot visual imitation," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2020.

[2] Yu, Tianhe and Quillen, Deirdre and He, Zhanpeng and Julian, Ryan and Hausman, Karol and Finn, Chelsea and Levine, Sergey, "Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2020.

[3] Finn, Chelsea and Yu, Tianhe and Zhang, Tianhao and Abbeel, Pieter and Levine, Sergey, "One-shot visual imitation learning via meta-learning," in *Conference on robot learning*. PMLR, 2017.

[4] Mandi, Zhao and Liu, Fangchen and Lee, Kimin and Abbeel, Pieter, "Towards more generalizable one-shot visual imitation learning," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2022.

[5] Jang, Eric and Irpan, Alex and Khansari, Mohi and Kappler, Daniel and Ebert, Frederik and Lynch, Corey and Levine, Sergey and Finn, Chelsea, "Bc-z: Zero-shot task generalization with robotic imitation learning," in *Conference on Robot Learning*. PMLR, 2022, pp. 991–1002.

[6] Yu, Tianhe and Finn, Chelsea and Xie, Annie and Dasari, Sudeep and Zhang, Tianhao and Abbeel, Pieter and Levine, Sergey, "One-shot imitation from observing humans via domain-adaptive meta-learning," *arXiv preprint arXiv:1802.01557*, 2018.

[7] Pomerleau, Dean A, "Alvinn: An autonomous land vehicle in a neural network," in *Advances in Neural Information Processing Systems (NeurIPS)*, 1988.

[8] Ross, Stéphane and Gordon, Geoffrey and Bagnell, Drew, "A reduction of imitation learning and structured prediction to no-regret online learning," in *AISTATS*, 2011, pp. 627–635.

[9] Duan, Yan and Andrychowicz, Marcin and Stadie, Bradly and Jonathan Ho, OpenAI and Schneider, Jonas and Sutskever, Ilya and Abbeel, Pieter and Zaremba, Wojciech, "One-shot imitation learning," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2017.

[10] Finn, Chelsea and Abbeel, Pieter and Levine, Sergey, "Model-agnostic meta-learning for fast adaptation of deep networks," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017, pp. 1126–1135.

[11] Pathak, Deepak and Mahmoudieh, Parsa and Luo, Guanghao and Agrawal, Pulkit and Chen, Dian and Shentu, Yide and Shelhamer, Evan and Malik, Jitendra and Efros, Alexei A and Darrell, Trevor, "Zero-shot visual imitation," in *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, 2018, pp. 2050–2053.

[12] Sharma, Pratyusha and Pathak, Deepak and Gupta, Abhinav, "Third-person visual imitation learning via decoupled hierarchical controller," *Advances in Neural Information Processing Systems*, vol. 32, 2019.

[13] Ahn, Michael and Brohan, Anthony and Brown, Noah and Chebotar, Yevgen and Cortes, Omar and David, Byron and Finn, Chelsea and Gopalakrishnan, Keerthana and Hausman, Karol and Herzog, Alex and others, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv preprint arXiv:2204.01691*, 2022.

[14] James, Stephen and Bloesch, Michael and Davison, Andrew J, "Task-embedded control networks for few-shot imitation learning," in *Conference on robot learning*. PMLR, 2018, pp. 783–795.

[15] Bonardi, Alessandro and James, Stephen and Davison, Andrew J, "Learning one-shot imitation from humans without humans," *IEEE Robotics and Automation Letters*, vol. 5, no. 2, pp. 3533–3539, 2020.

[16] Huang, De-An and Xu, Danfei and Zhu, Yuke and Garg, Animesh and Savarese, Silvio and Fei-Fei, Li and Niebles, Juan Carlos, "Continuous relaxation of symbolic planner for one-shot imitation learning," in *International Conference on Intelligent Robots and Systems*, 2019.

[17] Smith, Laura and Dhawan, Nikita and Zhang, Marvin and Abbeel, Pieter and Levine, Sergey, "Avid: Learning multi-stage tasks via pixel-level translation of human videos," *arXiv preprint arXiv:1912.04443*, 2019.

[18] Zakka, Kevin and Zeng, Andy and Florence, Pete and Tompson, Jonathan and Bohg, Jeannette and Dwibedi, Debidatta, "Xirl: Cross-embodiment inverse reinforcement learning," in *Proceedings of the Conference on Robot Learning (CoRL)*, 2022.

[19] Jin, Jun and Petrich, Laura and Dehghan, Masood and Jagersand, Martin, "A geometric perspective on visual imitation learning," in *International Conference on Intelligent Robots and Systems*, 2020.

[20] Xiong, Haoyu and Li, Quanzhou and Chen, Yun-Chun and Bharadhwaj, Homanga and Sinha, Samarth and Garg, Animesh, "Learning by watching: Physical imitation of manipulation skills from human videos," in *International Conference on Intelligent Robots and Systems*, 2021.

[21] Ho, Jonathan and Ermon, Stefano, "Generative adversarial imitation learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 29, 2016.

[22] Bahl, Shikhar and Gupta, Abhinav and Pathak, Deepak, "Human-to-robot imitation in the wild," *RSS*, 2022.

[23] S. Bansal, V. Tolani, S. Gupta, J. Malik, and C. Tomlin, "Combining optimal control and learning for visual navigation in novel environments," *CoRL*, 2019.

[24] Chaplot, Devendra Singh and Gandhi, Dhiraj and Gupta, Saurabh and Gupta, Abhinav and Salakhutdinov, Ruslan, "Learning to explore using active neural slam," *arXiv preprint arXiv:2004.05155*, 2020.

[25] Meng, Xiangyun and Ratliff, Nathan and Xiang, Yu and Fox, Dieter, "Neural autonomous navigation with riemannian motion policy," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 8860–8866.

[26] Müller, Matthias and Dosovitskiy, Alexey and Ghanem, Bernard and Koltun, Vladlen, "Driving policy transfer via modularity and abstraction," *arXiv preprint arXiv:1804.09364*, 2018.

[27] Kaufmann, Elia and Loquercio, Antonio and Ranftl, Rene and Dosovitskiy, Alexey and Koltun, Vladlen and Scaramuzza, Davide, "Deep drone racing: Learning agile flight in dynamic environments," in *Conference on Robot Learning*. PMLR, 2018, pp. 133–145.

[28] Kaufmann, Elia and Gehrig, Mathias and Foehn, Philipp and Ranftl, René and Dosovitskiy, Alexey and Koltun, Vladlen and Scaramuzza, Davide, "Beauty and the beast: Optimal methods meet learning for drone racing," in *2019 International Conference on Robotics and Automation (ICRA)*. IEEE, 2019, pp. 690–696.

[29] Nair, Ashvin V and Pong, Vitchyr and Dalal, Murtaza and Bahl, Shikhar and Lin, Steven and Levine, Sergey, "Visual reinforcement learning with imagined goals," *Advances in neural information processing systems*, vol. 31, 2018.

[30] Finn, Chelsea and Levine, Sergey, "Deep visual foresight for planning robot motion," in *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017, pp. 2786–2793.

[31] R. Chitnis, S. Tulsiani, S. Gupta, and A. Gupta, "Efficient bimanual manipulation using learned task schemas," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

[32] Dalal, Murtaza and Pathak, Deepak and Salakhutdinov, Russ R, "Accelerating robotic reinforcement learning via parameterized action primitives," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[33] Kaelbling, Leslie Pack and Lozano-Pérez, Tomás, "Hierarchical planning in the now," in *Workshops at the Twenty-Fourth AAAI Conference on Artificial Intelligence*, 2010.

[34] Pirk, Sören and Hausman, Karol and Toshev, Alexander and Khansari, Mohi, "Modeling long-horizon tasks as sequential interaction landscapes," *arXiv preprint arXiv:2006.04843*, 2020.

[35] Nasiriany, Soroush and Liu, Huihan and Zhu, Yuke, "Augmenting reinforcement learning with behavior primitives for diverse manipulation tasks," *arXiv preprint arXiv:2110.03655*, 2021.

[36] Kumar, Ashish and Gupta, Saurabh and Malik, Jitendra, "Learning navigation subroutines by watching videos," in *CoRL*, 2019.

[37] Shankar, Tanmay and Tulsiani, Shubham and Pinto, Lerrel and Gupta, Abhinav, "Discovering motor programs by recomposing demonstrations," in *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[38] Shankar, Tanmay and Gupta, Abhinav, "Learning robot skills with temporal variational inference," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[39] Kipf, Thomas and Li, Yujia and Dai, Hanjun and Zambaldi, Vinicius and Sanchez-Gonzalez, Alvaro and Grefenstette, Edward and Kohli, Pushmeet and Battaglia, Peter, "Compile: Compositional imitation learning and execution," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2019.

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

[41] Chen, Ting and Kornblith, Simon and Norouzi, Mohammad and Hinton, Geoffrey, "A simple framework for contrastive learning of visual representations," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[42] Srinivas, Aravind and Laskin, Michael and Abbeel, Pieter, "CURL: Contrastive unsupervised representations for reinforcement learning," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[43] Laskin, Misha and Lee, Kimin and Stooke, Adam and Pinto, Lerrel and Abbeel, Pieter and Srinivas, Aravind, "Reinforcement learning with augmented data," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[44] Zhang, Hongyi and Cisse, Moustapha and Dauphin, Yann N and Lopez-Paz, David, "mixup: Beyond empirical risk minimization," *arXiv preprint arXiv:1710.09412*, 2017.

[45] Yun, Sangdoo and Han, Dongyoon and Oh, Seong Joon and Chun, Sanghyuk and Choe, Junsuk and Yoo, Youngjoon, "Cutmix: Regularization strategy to train strong classifiers with localizable features," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019, pp. 6023–6032.

[46] Berthelot, David and Carlini, Nicholas and Goodfellow, Ian and Papernot, Nicolas and Oliver, Avital and Raffel, Colin A, "Mixmatch: A holistic approach to semi-supervised learning," *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.

[47] Levine, Sergey and Pastor, Peter and Krizhevsky, Alex and Ibarz, Julian and Quillen, Deirdre, "Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection," *IJRR*, 2018.
[48] Pinto, Lerrel and Gupta, Abhinav, "Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours," *ICRA*, 2016.
[49] Cuturi, Marco and Blondel, Mathieu, "Soft-dtw: a differentiable loss function for time-series," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2017.
[50] He, Kaiming and Zhang, Xiangyu and Ren, Shaoqing and Sun, Jian, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

APPENDIX

VII. Motor Primitives Details

For the experiments in this paper we utilize four motor primitives: free-space motion, carrying, dropping, and grasping. We normalize the action spaces for all environments to behave like the Meta-world environments. They have 4 degrees of freedom, 3-DOF delta end-effector position control, and 1-DOF for desired gripper closure. We implement 3-DOF delta end-effector position control by driving the joints to a configuration that achieves the desired end-effector position as obtained through inverse kinematics.

Free-space Motion: Given the current end-effector position and a destination, for each time step we move the end-effector directly towards the destination point in a straight line, using the abstracted delta end-effector position control space defined above.

Carrying: Carrying is implemented using the above free-space motor primitive, just keeping the gripper closed throughout the movement.

Dropping: Dropping is implemented by simply opening the gripper in place.

Grasping: We utilize depth images from an eye-in-hand camera for the grasping primitive. At a high level, the grasping primitive attempts to localize the nearest object to the gripper, position the gripper above that object, then drop down to perform the grasp. We first outline the steps of the motor primitive, and then describe the object localization procedure.

1) Determine the target object position using depth images (described below).
2) Move the gripper to a point 15cm above the estimated object position, recomputing the object position at each time-step.
3) After reaching a point above the object, move the gripper down to the estimated object z position offset by some small value (3cm for the panda gripper, but this should be adjusted based on the agent end effector).
4) Close the gripper.
5) Lift 15cm.

Next, we describe the object localization procedure to recover the target object position (step 1 above):

1) Mask out background values, defined as having a depth greater than 1 meter.
2) Determine the ground plane distance by taking the median of the remaining depth values.
3) Produce a mask for potential objects, taking pixels 1cm or more above the floor plane.
4) Compute connected components of the potential object mask to get individual object proposals.
5) Select the connected component with the centroid closest to the center of the frame as the closest object (target object for grasping).
6) Use the camera intrinsics to project the depth point of the centroid back into 3D world coordinates. This is the estimated object position.

A. Mining Attributes

Whether or not the attribute of *"is there an object in the gripper"* is present at every timestep t, is mined directly from the sequence of joint angles and commanded actions $\{(s_1, a_1)\ldots\}$. We determine that an object is held when the commanded action is to close the gripper, but the gripper jaws do not close. At each timestep, we annotate if it potentially contains a grasp with a variable g_t.

$$g_t = c_t \wedge (s^g_{t+d} - s^g_t > \delta)$$

Where c_t is a boolean indicating if the gripper was commanded to close on frame t, and s^g_t is the component of the robot state at time t that corresponds to the distance between the gripper jaws. d and δ are tuned based on the timescale of the data, and dimensions of the robot. For the Meta-world data, we use $d = 10$ and $\delta = -0.05$, for the T-OSVI data we use $d = 1$ and $\delta = -0.05$. For BC-Z, we use $d = 1$, $\delta = -0.07$. The final sequence of potential grasp frames $g_0, \ldots, g_T$ is smoothed with a convolutional filter to give the final attribute annotations.

In addition to the grasping primitive used in our experiments, we have validated that this approach can work well for other attributes. We are able to detect a "pressing" attribute (*i.e.* button presses) using a similar methodology as for detecting object grasps. We identify frames in which the commanded action is to move the end-effector in a certain direction, but the end-effector does not move, or accelerate in a direction consistent with the commanded action. This means there must be some object obstructing the end-effector, *i.e.* the end-effector is pressing into something. We visualize the results of this technique in Figure 7. We show the first pressing frame detected in trajectories from four different Meta-World tasks which require pressing (`coffee-button-v2`, `button-press-v2`, `handle-press-v2`, and `button-press-topdown-v2`). We find that this form of automatic mining is able to accurately identify frames in which the agent is performing button presses or other pressing-like actions.

VIII. Implementation Details

A. Architecture

The architecture has three modules as shown in Figure 8: a convolutional image feature extractor, a transformer with self-attention layers for temporal processing, and a multi-headed MLP for waypoint prediction. These first two modules (feature extractor and transformer) are identical to the T-OSVI architecture. Our architecture differs only in the waypoint prediction heads. We notate the observation of the novel task instance as o_1, as only the first frame is used in our method. For the baselines, we use a frame-stack of 6 images $\{\ldots, o_{t-1}, o_t\}$, following T-OSVI.

Image feature extraction: Visual features are extracted by a pre-trained ResNet-18 [50]. We share weights between the feature extractor for $\mathbf{v}$, and o_1, frames. We remove the last two layers of the ResNet and use the last spatial features.

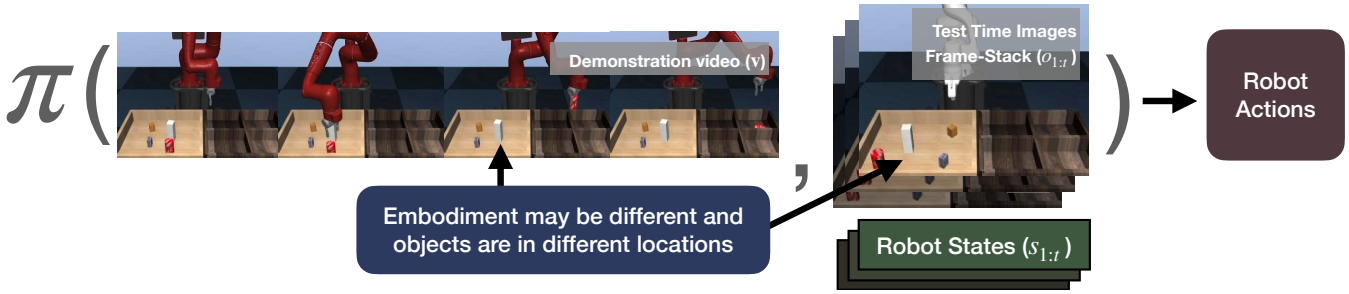

Fig. 6. **The one-shot visual imitation problem**: Given a video demonstration ($\mathbf{v}$) of an agent performing a novel task (a task not seen during training), and the sequence of images for the current execution so far ($o_{1:t}$), the policy $\pi(a_t|\mathbf{v}, o_{1:t}, s_{1:t})$ must predict the next action to take to accomplish the task depicted in the video demonstration. Note that the agent embodiment and the locations of the objects may differ between the video demonstration and the test time environment.

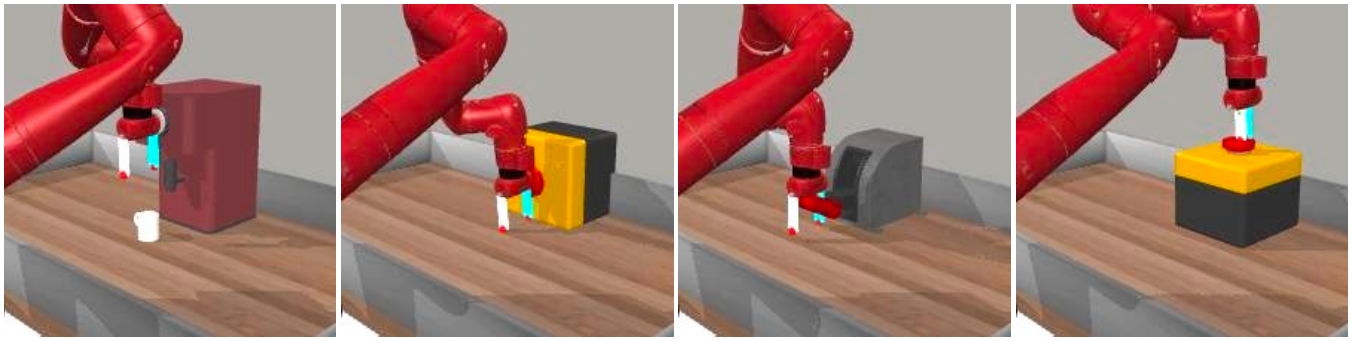

Fig. 7. The first frames in which a "pressing" attribute is detected in four trajectories from four different task in the Meta-World task-set, from left to right: `coffee-button-v2`, `button-press-v2`, `handle-press-v2`, and `button-press-topdown-v2`

Self-Attention Layers: We pass these spatial features into the self-attention module. We stack spatial features together along the time dimension, with o_1 (and any frame-stacking for non-waypoint baselines) placed after $\mathbf{v}$. We add sinusoidal positional encodings to the features, treating time and space as a single dimension. We then pass these spatial features with positional encodings into a multi-headed self-attention module. We create key, query, and value tensors by applying 3D convolutions with kernel size 1. We then apply dropout, a residual connection, and batch normalization after each self-attention layer. Following T-OSVI, all methods use 4 attention heads and 2 layers of self-attention. Finally, after the attention layers, we apply a spatial softmax, followed by a two-layer MLP. This projects the features from the final time-step, corresponding to the o_1, into a fixed length vector.

Waypoint Prediction: We pass the final features (fixed-length vector corresponding to o_1 from above) through a multi-headed 2-layer MLP, *i.e.*, there is a separate layer to transform the features into waypoints for task-driven samples and synthesized samples. We select predictions from one head or the other based on the dataset of origin (task-driven or synthesized) of each trajectory in the batch.

B. Loss Computation

We linearly interpolate between the waypoints to produce a set of points amendable for use with Soft Dynamic Time Warping (SDTW) [39] with a fixed temperature parameter of 0.001. All experiments downstream are conducted using 5 waypoints. However, at training time we predict a total of 5 trajectories consisting of 1, 2, 3, 4, and 5 waypoints respectively, for a total of 15 waypoints. The SDTW loss against ground truth is computed independently for each of these 5 trajectories. The final loss is the mean across the computed SDTW distance for all 5 trajectories.

C. Inference Speed

When running our model on a single modern GPU (Nvidia A40), our model is able to make a forward pass through the network to predict waypoints in 7 ms. After waypoints are predicted, using the motor primitives to command the agent requires less than 2ms per environment step.

TABLE IV
HYPER-PARAMETERS FOR TRAINING AND EVALUATION

Hyperparameter	Value
Image size (Meta-world)	(224, 224)
Image size (Pick-and-place)	(240, 320)
Learning Rate	0.0005
Batch Size	30
Non-Local Attention Layers	2
Attention Heads per Layer	4
# of Waypoints for Evaluation	5
Evaluation Episodes per Snapshot per Task	20
γ (SDTW Temperature)	0.001
# of Demonstration ($\mathbf{v}$) frames	10
Optimizer	Adam
Post-Transformer Hidden Dimension	256

IX. DAgger Problem Continued Analysis

One may wonder about the role of action sampling and frame-stacking in our analysis of the DAgger problem in Section III. We investigate this by evaluating the baseline (with frame-stack 6) by taking the mean action and mode action. Additionally, we train a model using no frame-stacking,

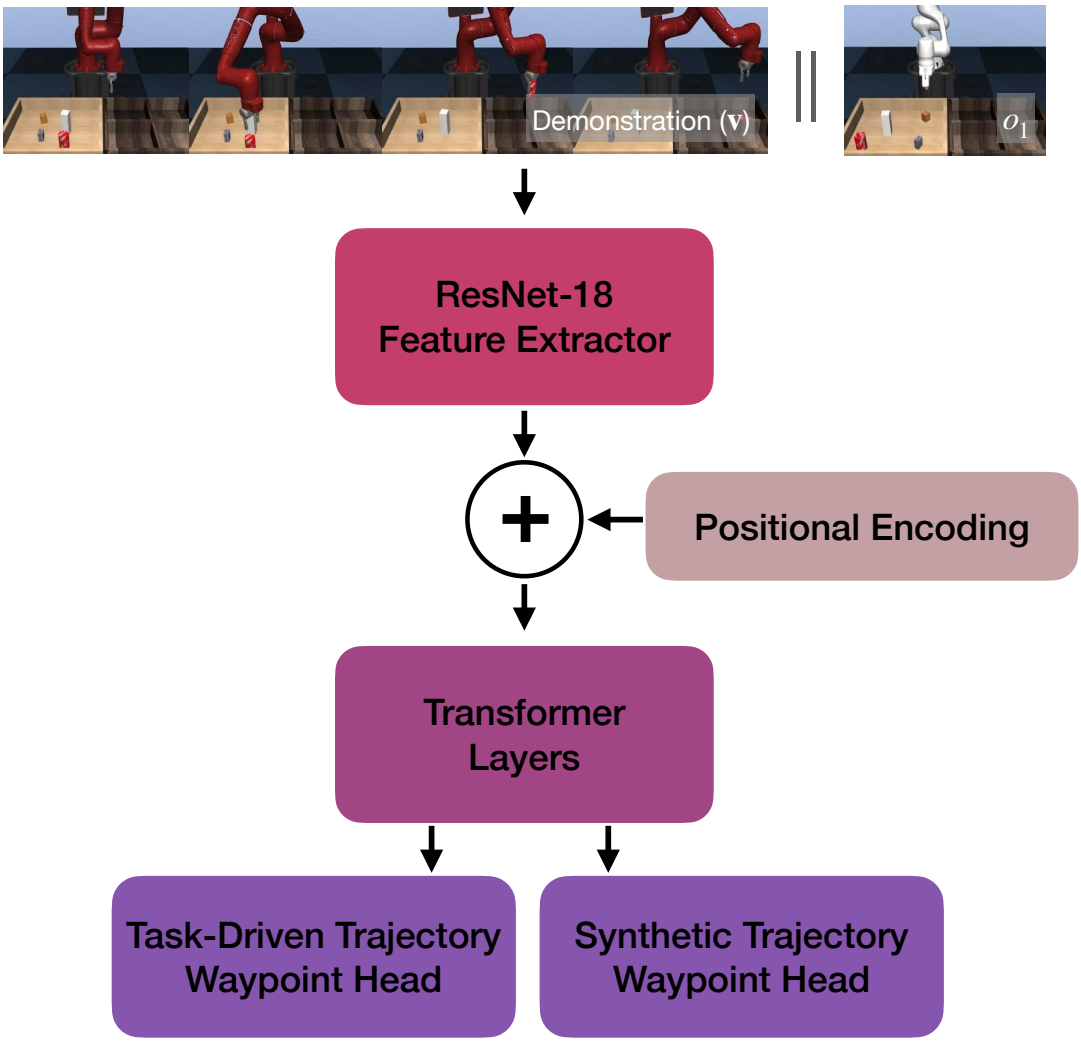

Fig. 8. Outline of our model architecture. Following T-OSVI [1], we extract features using ResNet-18. These features remain spatial, and have a sinusoidal positional encoding added, before being processed by a transformer module. Finally, the temporally processed features are projected down into waypoints by two separate heads, one to predict waypoints for task-driven trajectories, and one for trajectories synthesized as described in Section IV-B.

relying only on the current frame for prediction. We find that none of these modifications address the failure mode studied in Section III.

When training with no frame-stack we see the same poor performance, with this model achieving a reaching success of 0.12. We believe this is because, in any state where the recent frames dictate which object to reach, only the current frame is sufficient to make the same prediction, *i.e.*, the model can just look at which object the gripper has moved towards from the current frame. It doesn't need the past frames for this. Additionally, we find the model with no frame-stack achieves a significantly *lower* overall success rate than the baseline down to 0.01 from 0.1. This is because the frame-stack is useful for performing the grasp on the object. Demonstrations pause the griper above the object before grasping, and the model is unable to fit this behavior well without the previous frames: it pauses above the object and never descends to attempt a grasp. Using the mean or mode action did not resolve the reaching issue either, achieving a reaching success of 0.11 and 0.10 respectively.

X. Grasping Failure Details

Grasping analysis results (as reported in Section III, and above) are computed over 440 evaluations on the best snapshot from the action-prediction baseline T-OSVI. We consider the object to be successfully reached when the end effector comes within 4cm of the object and no grasp has yet been attempted in the current episode. A grasp is considered successful when the object has been reached, and then the object is raised at least 5cm off the ground.

As mentioned in Section III, even when the right object is reached, grasping control in an end-to-end learned model with limited data is often unreliable. In Figure 9 we visualize two of the typical grasping failure modes: missing the grasp, or acquiring an unstable grasp which drops the object.

XI. Experimental Details and Results

In Table V we report mean success rates with standard error over the 10 evaluated snapshots of our method. In addition to methods reported in the main paper, we include **only waypoints**, which has neither additional data, nor our asymmetric demonstration mixup augmentation, and

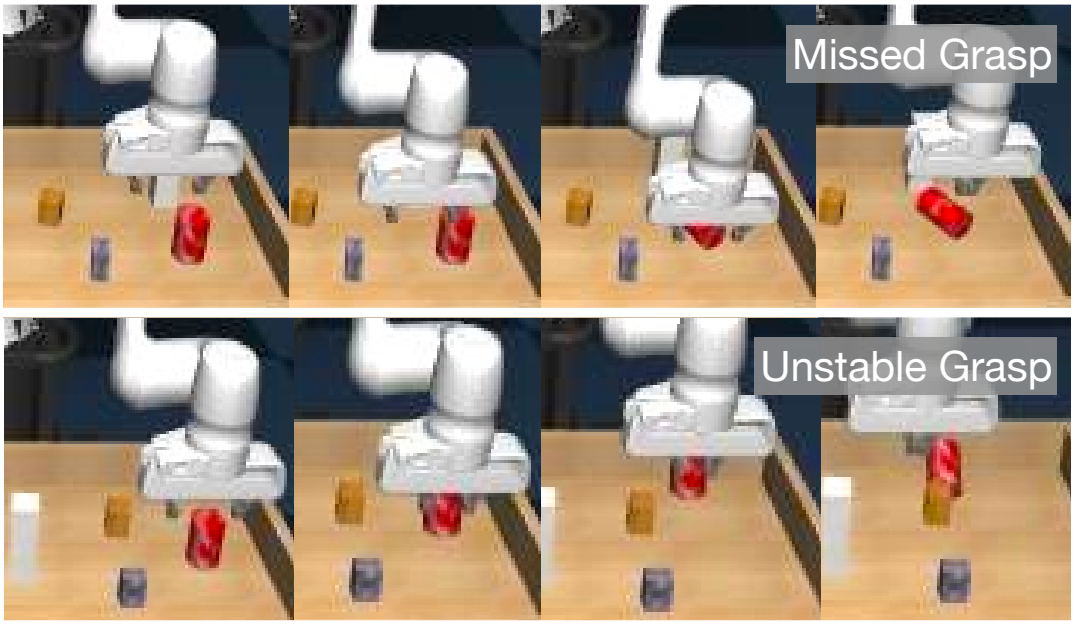

Fig. 9. The two most common grasping failure modes: missing the grasp entirely, and acquiring an unstable grasp.

T-OSVI +depth, which is T-OSVI augmented to include the same eye-in-hand depth images that our method uses for grasping. We do this by adding a separate ResNet for depth image feature extraction, and perform late-fusion, combining RGB and depth image features into one vector before transformer layers. This performs significantly worse, likely because this adds more capacity to over-fit, following our analysis in Section III . In Table VI, we report the error bars for the MOSAIC results from Table III. We follow the same procedure as other environments, reporting the mean and standard error of success rates over the last 10 snapshots from training. However, the results from MOSAIC [4] only report standard deviation over the last 3 snapshots from training. For fairer comparison we have computed the estimated standard error based on their reported standard deviations assuming normality of the underlying samples.

XII. Visualizations

Figure 10 shows the waypoint predictions of our model on novel tasks. Notice that our method mimics the overall solution scheme of the demonstration while adapting to the different environment configuration in the current scene. We visualize trajectories from our trajectory synthesis method from Section IV-B, in Figure 11.

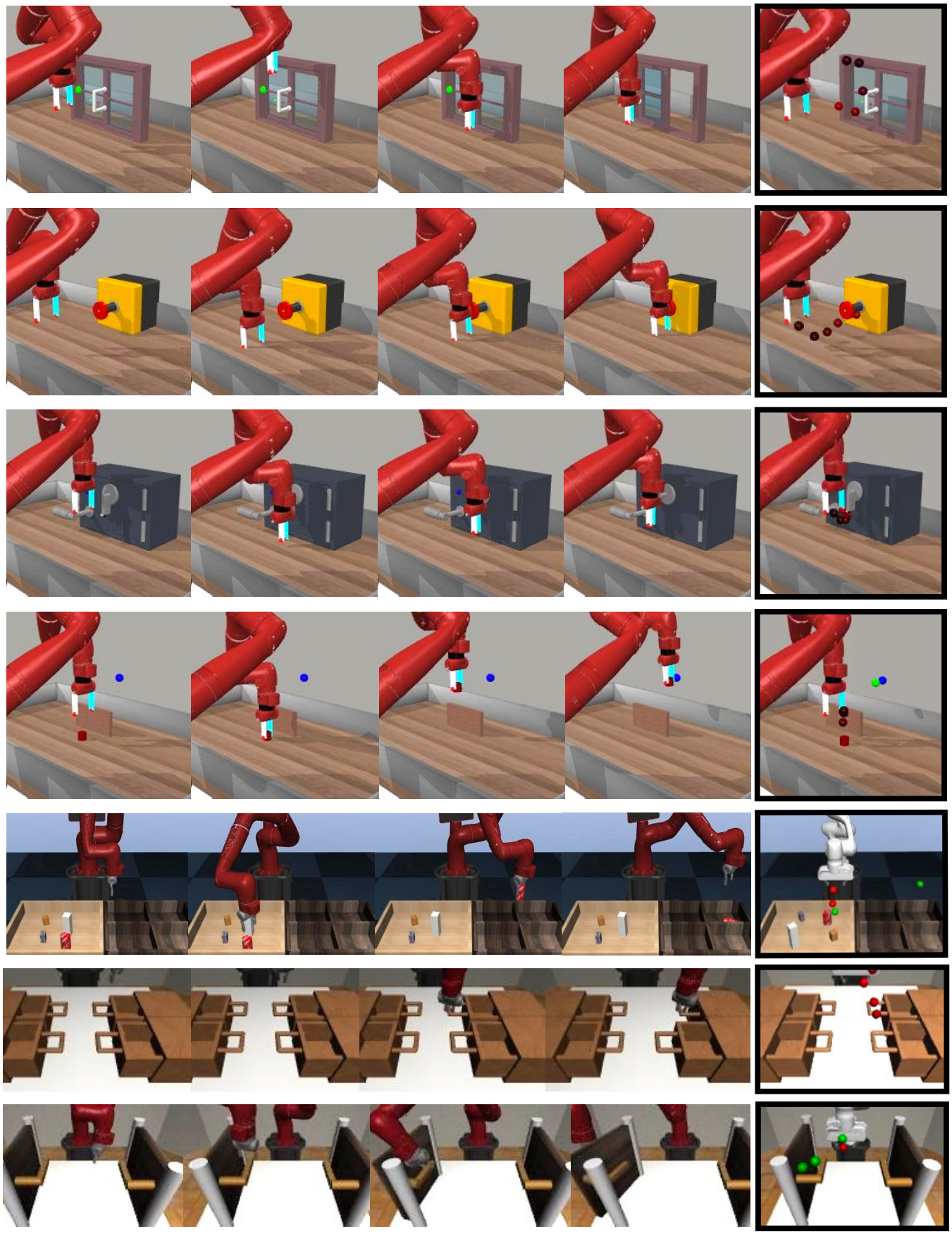

Fig. 10. We visualize example waypoint predictions on held-out tasks on Metaworld, Pick-and-place, and MOSAIC. Each row is a different novel task that the model has not seen during training. The frames on the left represent the video demonstration, and the frame on the right shows the waypoints predicted in a new setting. Red spheres represent free-space waypoints. The increasing order of waypoints is indicated by a brightening of the color. Green spheres indicate the end point of line segments attributed with the *object grasped* attribute. This means the grasping primitive should be invoked at the previous waypoint if it is red, and the object should be carried to the green waypoint.

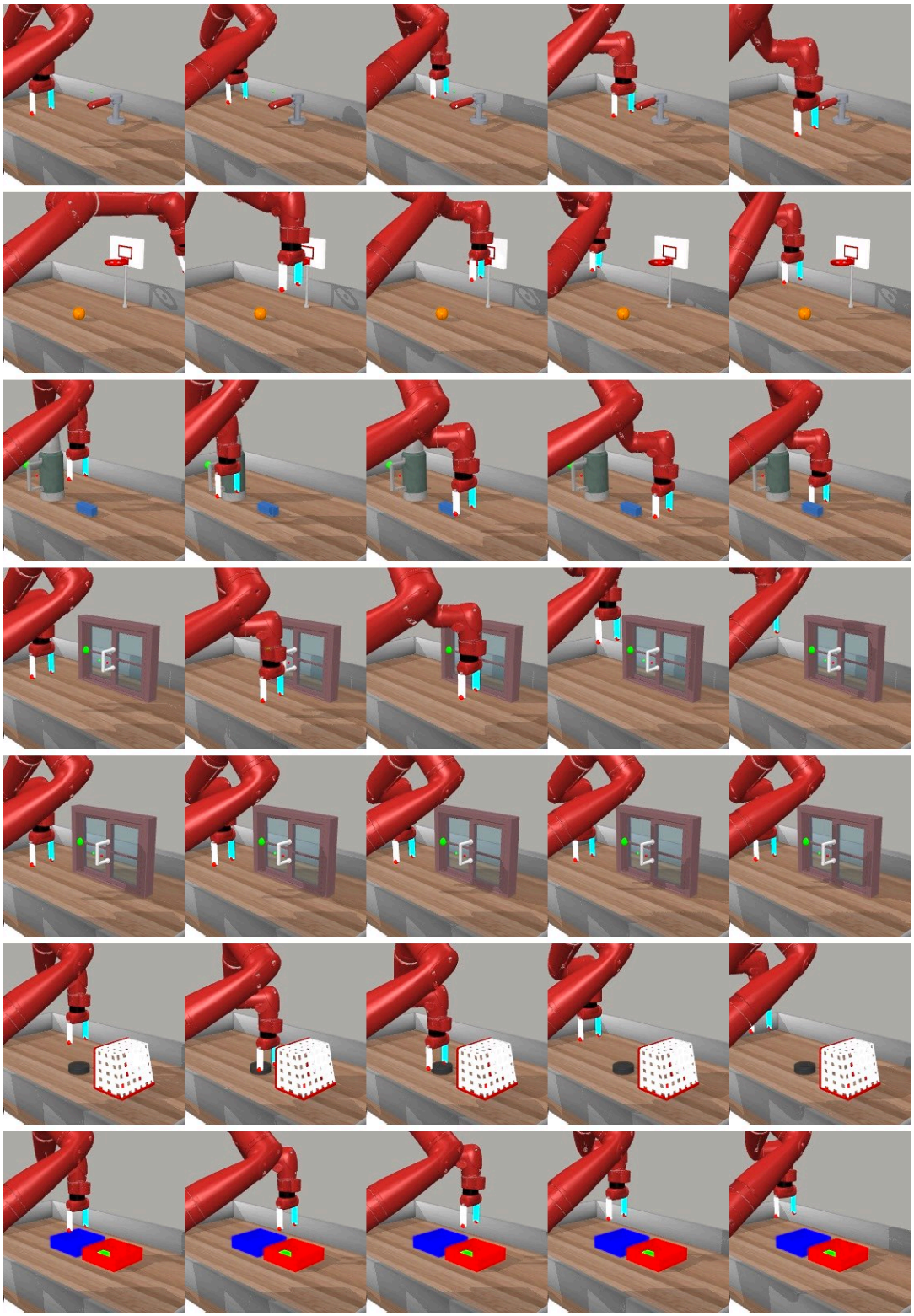

Fig. 11. We visualize example trajectories generated by our proposed trajectory synthesis method (Section IV-B). Trajectories are synthesized by driving the end-effector through randomly sampled waypoints in the environments used for training. Introducing this data into the training dataset helps decorrelate tasks from task contexts.

TABLE V

WE REPORT MEAN SUCCESS RATES AND STANDARD ERROR ON HELD-OUT TASKS ON THE PICK-AND-PLACE AND META-WORLD BENCHMARKS.

	DAML [6]	**T-OSVI [1]**	**T-OSVI +depth**	**Full-1**	**Full-2**	**single head**	**no AD**	**no ADM**	**no way points**	**only way points**
Asymm. Demo Mixup (ADM)?				✓	✓	✓	✓	✗	✓	✗
Additional Data (AD) Source?				TS	BC-z	TS	✗	TS	TS	✗
Waypoints?				✓	✓	✓	✓	✓	✗	✓
Pick-and-place	$.01 \pm .00$	$.10 \pm .05$	$.00 \pm .00$	$1.0 \pm .00$	$1.0 \pm .00$	$.99 \pm .00$	$1.0 \pm .00$	$.98 \pm .01$	$.01 \pm .02$	$.98 \pm .01$
Meta-world [easy]	$.04 \pm .01$	$.50 \pm .08$	$.02 \pm .01$	$.66 \pm .05$	$.73 \pm .08$	$.33 \pm .06$	$.74 \pm .05$	$.03 \pm .01$	$.33 \pm .10$	$.14 \pm .08$
Meta-world [hard]	$.08 \pm .03$	$.07 \pm .03$	$.02 \pm .01$	$.30 \pm .04$	$.17 \pm .04$	$.29 \pm .03$	$.11 \pm .04$	$.19 \pm .06$	$.06 \pm .02$	$.02 \pm .01$
Meta-world [all]	$.06 \pm .01$	$.28 \pm .06$	$.02 \pm .01$	$.48 \pm .05$	$.45 \pm .08$	$.31 \pm .03$	$.42 \pm .08$	$.11 \pm .04$	$.19 \pm .06$	$.08 \pm .04$

TABLE VI

SUCCESS RATES ON HELD-OUT TASKS FROM MOSAIC [4] TASK-SET

Task	Door	Drawer	Press Button	Stack Block	Basketball	Nut Assembly	All
MOSAIC [4][1]	0.05 ± 0.018	0.15 ± 0.038	$\mathbf{0.05} \pm 0.018$	0	0	0	0.04 ± 0.008
Full-1	$\mathbf{0.10} \pm 0.02$	$\mathbf{0.29} \pm 0.033$	0.01 ± 0.005	0	0	$\mathbf{0.02} \pm 0.01$	$\mathbf{0.07} \pm 0.007$

[1] The results from MOSAIC [4] only report standard deviation over the last 3 snapshots from training, where we report standar error over the last 10 snapshots for a lower variance estimate. For fairer comparison we have computed the estimated standard error based on their reported standard deviations assuming normality of the underlying samples.