Natural Language Can Help Bridge the Sim2Real Gap

Albert Yu, Adeline Foote, Raymond Mooney, and Roberto Martín-Martín
UT Austin
{albertyu, addiefoote, mooney}@utexas.edu, robertomm@cs.utexas.edu

***Abstract*—The main challenge in learning image-conditioned robotic policies is acquiring a visual representation conducive to low-level control. Due to the high dimensionality of the image space, learning a good visual representation requires a considerable amount of visual data. However, when learning in the real world, data is expensive. Sim2Real is a promising paradigm for overcoming data scarcity in the real-world target domain by using a simulator to collect large amounts of cheap data closely related to the target task. However, it is difficult to transfer an image-conditioned policy from sim to real when the domains are very visually dissimilar. To bridge the sim2real visual gap, we propose using natural language descriptions of images as a unifying signal across domains that captures the underlying task-relevant semantics. Our key insight is that if two image observations from different domains are labeled with similar language, the policy should predict similar action distributions for both images. We demonstrate that training the image encoder to predict the language description or the distance between descriptions of a sim or real image serves as a useful, data-efficient pretraining step that helps learn a domain-invariant image representation. We can then use this image encoder as the backbone of an IL policy trained simultaneously on a large amount of simulated and a handful of real demonstrations. Our approach outperforms widely used prior sim2real methods and strong vision-language pretraining baselines like CLIP and R3M by 25 to 40%. See additional videos and materials at https://robin-lab.cs.utexas.edu/lang4sim2real/.**

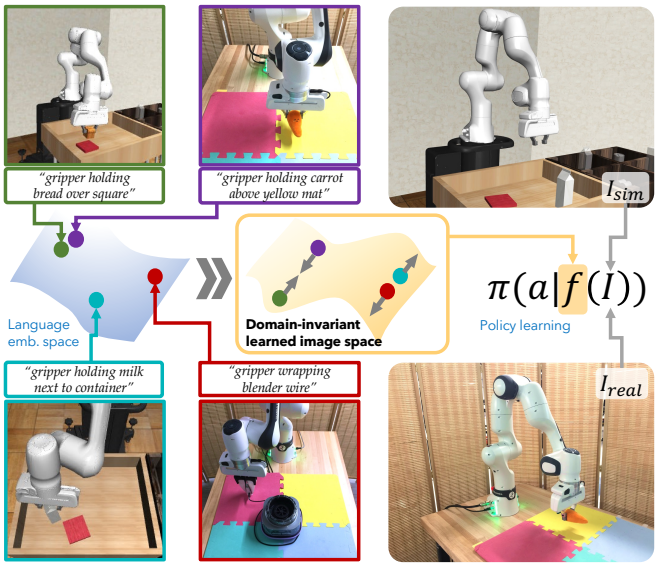

Fig. 1. **Bridging the sim2real gap with language**. Robot images from simulation and the real world with similar language descriptions (*green & purple borders*) are mapped to similar features in language embedding space, while sim and real images with different language descriptions (*teal & red*) are mapped to faraway locations. We propose using language embedding similarities to re-shape the image embeddings (*center*) to create a domain-invariant image space. A policy is learned conditioned on these image embeddings from both sim and real images (*right*).

I. Introduction

Recently, visual imitation learning (IL) has achieved significant success on manipulation tasks in household environments [46, 5]. However, these methods rely on large amounts of data in very similar domains to train data-hungry image-conditioned policies [5, 6, 39]. Some researchers are attempting to generalize visual IL to any target domain by collecting large datasets of demonstrations from mixed domains. In this work, we explore a different approach: can we transfer a policy trained on cheaply acquired, diverse simulation data to a real-world target task with just a few demonstrations?

A solution to effectively leverage cheap sim data while successfully fitting scarce real-world demonstrations is to create a domain-agnostic visual representation and use it for policy training. Such a representation should enable the policy to use the simulation image-action data as an inductive bias to learn with few-shot real world data. This representation must allow the policy to tap into the right distribution of actions by being broad enough to capture the task-relevant semantic state from image observations, yet fine-grained enough to be conducive to low-level control. For instance, a sim and real image observation, both showing the robot gripper a few inches above a pan handle, should lie close together in the image embedding space to lead to similar actions, even if the two images have large differences in pixel space.

How might we acquire supervision for learning such a visual representation? Language is an ideal medium for providing it. Descriptions of task-relevant features in image observations, such as whether or not a gripper is close to a pan handle, serve as a unifying signal to align the representations of images between sim and real. We hypothesize that if a sim and real image have similar language descriptions (e.g., "the gripper is open and right above the pan handle"), then their underlying semantic states are also similar, and thus the actions the policy predicts conditioned on each image should also be semantically similar (e.g., moving downward to reach the pan handle). The pretrained embedding space of large language models (LLMs) offers a well-tuned signal that can

be leveraged to measure the semantic similarity between real and sim images via their associated language descriptions (see Fig. 1). This simple insight allows us to learn a domain-agnostic visual representation to bridge the visual sim2real gap.

A popular paradigm in foundation model research is to first pretrain the backbone on large datasets, and then add and train a task-specific head to process the backbone outputs to perform a downstream task. We borrow from this paradigm by first pretraining an image encoder to predict the pretrained embeddings of language descriptions of images from roughly a few hundred trajectories in sim and real, with language labels on each image. Then we use this image encoder as the backbone of our IL policy and train on action-labeled data from both the sim and real domains simultaneously, where we only need a few action-labeled demonstrations from the real world.

In this paper, we introduce Lang4Sim2Real, a lightweight framework for transferring between any two domains that have large visual differences but contain data across a similar distribution of tasks. Our approach has the following main advantages over prior sim2real efforts:

1) Alleviates the need for the engineering-intensive task of system identification, or more broadly trying to exactly match a sim environment to the real environment both visually and semantically.
2) Enables sim2real transfer on tasks involving deformable objects that are hard to simulate with the same dynamics and visual appearance as the real-world version of the objects.
3) Bridges a wide sim2real gap that includes differences in: camera point-of-view (1st vs 3rd person), friction and damping coefficients, task goals, robot control frequencies, and initial robot and object position distributions.

In the few-shot setting, on long-horizon multi-step real-world tasks, these advantages enable Lang4Sim2Real to outperform prior SOTA methods in sim2real and vision representation learning by 25-40%. To our knowledge, this is the first work that shows that using language to learn a domain-invariant visual representation can help improve the sample efficiency and performance of sim2real transfer.

II. Related Work

Our main contribution is a method to learn domain-invariant image representations by exploiting natural language descriptions as a bridge between domains for sim2real transfer. While we believe this has not been explored before, significant related research has been done in vision-language pretraining, sim2real techniques, and domain-invariant representations for control.

A. Vision Pretraining for Robotics

Various works have found that **vision-only pretraining** improves performance on image-based robotic policies. Prior work has explored pretraining objectives ranging from masked image modeling [43], image reconstruction [64, 16, 48], contrastive learning [28, 18], video frame temporal ordering [24], future frame prediction [64], and image classification [62, 58] on internet-scale datasets such as ImageNet [10], Ego4D [15], Something-Something [14], and Epic Kitchens [9]. While these vision-only pretraining objectives learn good representations for robotic control within a specific domain distribution (such as the real world), they are not necessarily robust to the wide domain shifts encountered during sim2real.

In **vision-language pretraining**, contrastive learning [42, 63] has been shown to learn valuable representations for robotic tasks [51, 52]. However, these pretrained visual representations are often overly influenced by the semantics of language captions. This results in a representation that is too object-centric to differentiate between different frames of a robot demonstration, lacking the level of granularity needed for spatial-temporal understanding. R3M [37] addresses this by learning semantics from language labels of videos but also training with a time contrastive loss between video frames. Prior work in multimodal representations [65] found language to be effective in aligning representations learned across multiple modalities including depth and audio. Instead of using language to bridge modalities, our approach uses language to bridge visual representations between domains.

B. Sim2Real

While we approach sim2real through vision-language pretraining, there are many alternative, well-researched techniques. **Domain randomization** [3, 32, 55] involves varying physical parameters and visual appearances of the simulation to train a policy that functions in a wide distribution of domains that hopefully also covers the target domain distribution. However, domain randomization requires a large amount of diverse training data and attempts to be simultaneously performant in an overly broad distribution of states, leading to a suboptimal and conservative policy that takes longer to train. **System identification** [61, 26] involves tuning the simulation parameters to match the real world in order to create a custom-tailored simulation environment that easily transfers to the real domain. However, this process is very engineering intensive and time consuming, and it may be intractable to simulate all real world physical interactions with high fidelity and throughput. In contrast, our sim2real approach can handle large source and target domain discrepancies with a few target task demonstrations and does not require system identification or domain randomization.

C. Domain-Invariant Representations

Several methods have been proposed to learn domain invariant representations. The domain-adaptation community has extensively researched using **Generative Adversarial Networks (GANs)** to map images from one distribution into another, using pixel space as a medium of common representation [21, 4, 19, 45]. However, GANs require a large training dataset and are notorious for unstable training. Additionally, enforcing similarity on the input image side at the pixel level is

less efficient than our method, which encourages cross-domain distributional similarity in a compact, low-dimensional image encoder space. Furthermore, researchers in self-driving have studied using **semantic segmentation** and depth maps [35, 2] as a common representation space between domains, though their effectiveness has only been demonstrated in navigation tasks with binary segmentation masks, which is too simplified for the long-horizon manipulation tasks we consider.

D. Language and Robotics

A growing body of work has investigated training **multitask robotic policies** conditioned on language instruction embeddings [22, 30, 33, 34, 50, 54, 53, 25], or a combination of language instructions and goal images/demonstrations [23, 49, 60]. Our approach also involves learning a language-conditioned policy, but unlike prior work, our main novelty is using language for a second use-case: as scene descriptors during pretraining to pull together semantically similar image observations between two visually dissimilar domains. Language has also been used for **reward shaping** in RL [36, 12, 13, 11, 31], and as a high-level planner in long-horizon tasks [20, 1, 7, 44]. These areas of research are more ancillary to our contributions, as we demonstrate our approach with IL instead of RL and with fine-grained manipulation tasks that do not require extensive planning.

III. Problem Description

In this work, we address the problem of few-shot visual imitation-learning (IL): learning a visuomotor manipulation policy in the real world based on a few real-world demonstrations. We assume access to a large amount of simulation data and cast few-shot IL as a sim2real problem. More concretely, we render the few-shot IL problem as a $k+1$ multi-task IL problem: k tasks from simulation and the target task (with a few demonstrations) in the real world. In general terms, we assume a *source domain* in which data can be acquired cheaply and a *target domain* where data is expensive to collect.

In our setting, we consider access to two datasets across two domains: $\mathcal{D}^{\text{s}}$, which spans multiple tasks in the source domain, and $\mathcal{D}^{\text{t}}_{\text{target}}$, which contains a small number of demonstrations of the target task in the target domain we want to transfer to. Thus, we assume that $|\mathcal{D}^{\text{s}}| >> |\mathcal{D}^{\text{t}}_{\text{target}}|$, due to how expensive target domain data collection is (such as in the real world). We make two assumptions about the two domains. First, we assume the source and target tasks are all of the same general structure, such as multi-step pick-and-place task compositions, but with different objects and containers across different subtasks. Otherwise, transfer would be infeasible in the low-data regime if the source and target domain tasks lack similarity. Second, to train a common policy for both domains, we assume the domains share state and action space dimensionality. We make no further assumptions about the similarity between the two domains.

All of our datasets are in the form of expert trajectories. Each trajectory, $\tau = \{(I_t, s_t, [a_t, l_t], l_{task})\}$, is a sequence of tuples containing an image observation, I_t (128×128 RGB), robot proprioceptive state, s_t (end effector position and joint angles), and a language instruction of the task, l_{task}. Note that l_{task} is the same over all timesteps of all trajectories in a given task. $[a_t, l_t]$ denotes that a trajectory may optionally also include robot actions (in which case we consider the trajectory a full demonstration) and/or a language description of the image I_t. In the following sections, we identify with $\tau[L]$ a trajectory with language descriptions l_t, but no actions a_t. Similarly, $\tau[A]$ is a full demonstration with actions, a_t, but no language descriptions, l_t.

The language labels for images can be automatically generated from a programmatic function that maps image observations to language scene descriptions depending on the relative position between the robot and the objects in the scene. We elaborate on these language labels and how to automatically collect them in Section IV-A. Note that these language *scene descriptions*, l_t, are different from the language *instruction* associated with each task, l_{task}.

Different data elements and types of trajectories will be used during pretraining and policy few-shot training: during pretraining, we use $\tau[L]$ image-language (I_t, l_t) pairs from $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$. During policy learning, we use $\tau[A]$ data: $(I_t, s_t, a_t, l_{task})$ tuples from $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$. In the next section, we explain how these two steps are defined for Lang4Sim2Real.

IV. Lang4Sim2Real: Few-Shot IL with Sim&Real

In our method, we adopt the common *pretrain-then-finetune* learning paradigm (see Fig. 2). First, we pretrain an image backbone encoder on cross-domain language-annotated image data (Sec. IV-B). Then, we freeze this encoder and train a policy network composed of trainable adapter modules and a policy head to perform behavioral cloning (BC) [46] on action-labeled data from both domains (Sec. IV-C). To leverage the simulation data, we train a $k+1$ multi-task BC policy that learns for k tasks in the source domain (sim) and 1 in the target domain (real, few shot).

A. Automatic Language labeling of Images

To acquire image-language pairs for pretraining, we implement an automated pipeline for labeling the images of a trajectory that occurs synchronously during scripted policy demonstration collection (see Appendix IX-A). Each if-case in the scripted policy corresponds to a stage index, where in pick-and-place, the first stage corresponds to the gripper moving to a point above the object, the second stage corresponds to the gripper moving vertically down toward the object, and so on. We define a list of template strings describing the scene for each of these stages, so the stage indexes into the template string list, giving us our language annotation for the image. See Table III in the Appendix for all template strings, and Appendix IX-G1 for details about our language labeling procedure.

However, our language labeling process need not be synchronously coupled with scripted policy demonstration collection. We also implemented a labeling process using off-the-

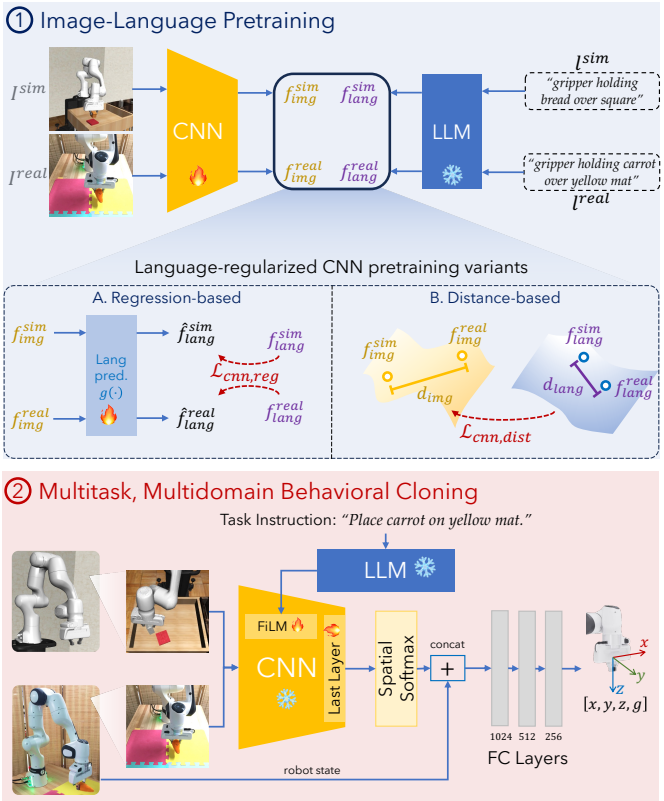

Fig. 2. **Method.** (i) Top: During **Image-Language Pretraining**, we train the image encoder f_{cnn} using the language embeddings associated with descriptions of both sim and real image observations. f^d_{img} and f^d_{lang} refer to the output features of the CNN and the LLM, respectively, in domain d. With regression-based loss (A) the image embeddings are pushed to predict the corresponding language embeddings whereas with distance based loss (B) the pair of image embeddings is pushed together/apart based on the similarity of the language embeddings. (ii) Bottom: During **Multitask, Multidomain BC**, we freeze our pretrained f_{cnn}, add adapter modules and a policy head and allow the last layer of the CNN to finetune, then train the resulting multitask language-conditioned policy on $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$.

shelf vision-language models to detect the location of objects and the gripper in the image to predict the stage number. This process can be run on previously-collected trajectories and requires only the images of a trajectory alone, without need for additional action or state information. We describe this second process in Appendix IX-G2. Empirically, using language from this second, more scalable automated approach does not degrade the performance of our method.

B. Cross-Domain Image-Language Pretraining

After collecting trajectories with language labels, our first step in Lang4Sim2Real involves learning a domain-invariant representation that will enable leveraging simulation data for few shot IL. For that, we need to learn an image observation encoder, $f_{cnn} : I_t \rightarrow \mathbb{R}^{d_{cnn}}$, that attains the following property: it should preserve the semantic similarity of scenes in images between the two domains. For instance, if both image I^s from $\mathcal{D}^{\text{s}}$ (sim) and image I^t from $\mathcal{D}^{\text{t}}_{\text{target}}$ (real world) show the robot's gripper open and a few inches above the object to grasp, even if from different viewing angles, then we want their image embeddings to be close together in the learned image encoding space. This will facilitate policy learning later, as the policy will need to draw from a similar distribution of actions for similar scene semantics, which are now already mapped into similar visual features.

Theoretically, off-the-shelf pretrained vision-language models (VLMs) [42, 37] should already possess these properties as they were trained on a massive distribution of image and language data. However, in the context of robot manipulation, pretrained VLMs tend to encode all observations of the trajectory into a very narrow region of the embedding space without sufficient distinction for task-relevant, semantic aspects of the image such as the location of the gripper in relation to the manipulated objects. This renders them unsuitable without additional finetuning for our application (see Sec. VI).

In Lang4Sim2Real, we propose an alternative approach to obtain a visual representation with the aforementioned desired property. We train a ResNet-18 [17] from scratch as our image encoder using image-language tuples (I^s, l^s) from $\mathcal{D}^{\text{s}}$ and (I^t, l^t) from $\mathcal{D}^{\text{t}}_{\text{target}}$. We denote this vision language pretraining dataset as $\mathcal{D}_{VL} = \{(I^d, l^d) : (I^d, l^d) \in \mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}\}$, where d is either the source or target domain. The images are observations collected during 100 demonstrations from each of the tasks in $\mathcal{D}^{\text{s}}$ and 25-100 demonstrations from $\mathcal{D}^{\text{t}}_{\text{target}}$, totaling around 10k images per domain. We assume that the two sets of language descriptions in $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}_{\text{prior}}$ are similarly distributed; otherwise, language may not help learn domain-invariant features between $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}$.

To effectively leverage language as a bridge between visually different domains, we need a well-tuned (frozen) language model, $f_{lang} : l \rightarrow \mathbb{R}^{d_{lang}}$, to map strings to $d_{lang}-$dimensional language embeddings. We use off-the-shelf miniLM [59], since prior work [34] has demonstrated its effectiveness for language-conditioned control policies compared to other small, off-the-shelf language models.

Given the data and the language embedding described above, we propose two variants in Lang4Sim2Real for the image-language pretraining step that can obtain a sim-real agnostic representation based on language supervision (see Fig. 2(i)A-B):

*1) **Language-Regression**:* Our first variant is a straightforward use of language supervision to shape the image embedding space: predicting the language embedding of the description, l^d, given the embedding of the corresponding image, I^d. We sample image-language pairs from the $\mathcal{D}_{VL}$ dataset defined above: $(I^d, l^d) \sim \mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$. Let $g : \mathbb{R}^{d_{cnn}} \rightarrow \mathbb{R}^{d_{lang}}$ be a single linear layer (language predictor in Fig. 2(i)(A)) trained to minimize the following loss:

$$\mathcal{L}_{cnn,reg}(\mathcal{D}_{VL}) = \left\| g\left(f_{cnn}(I^d)\right) - f_{lang}(l^d) \right\|_2^2 \qquad (1)$$

We use the loss to train both the language predictor and the CNN backbone. The loss provides strong language supervision by encouraging f_{cnn} to directly regress toward the frozen language embeddings of the image descriptions, effectively

making the pretrained image encoder reflect the LLM embedding space.

*2) **Language-Distance Learning**:* We also experiment with a second variant of image-language pretraining that incorporates language with a softer form of supervision. We posit that the exact values of the language embeddings do not themselves convey meaning; rather, key information about the semantic similarity of two images lies in the pairwise distances between their corresponding two language embeddings. Thus, we design an objective to regress the image embedding distances between a pair of images from the two domains to their corresponding language distance:

$$\mathcal{L}_{cnn,dist}(\mathcal{D}_{VL}) = \left\| f_{cnn}^{\top}(I^s) f_{cnn}(I^t) - d\big(l^s, l^t\big) \right\|_2^2 \quad (2)$$

where the language distance function we use, $d : l \times l \to \mathbb{R}$ is BLEURT [47], a learned similarity score between two strings commonly used in the NLP community. We normalize $d(\cdot,\cdot)$ between 0 and 1 for all possible (l^s, l^t) pairs in our image-language dataset, where 1 indicates the highest similarity between any two strings in the dataset. Empirically, over our set of language descriptions, we found BLEURT provided a richer signal than simply taking dot products or ℓ-2 distances between language embeddings. The output of f_{cnn} is unit normalized before taking the dot product. We compare both variants (see Sec. VI) to assess whether the additional degrees of freedom from the looser distance supervision are beneficial later on for policy training.

C. Multitask, Multidomain Behavioral Cloning

Our second step in Lang4Sim2Real involves learning a multi-domain, multi-task, language-conditioned BC policy (see Fig. 2(ii)). By leveraging our learned domain-invariant representation for robotic control, this policy should be able to perform well in real-world task with only a few demonstrations, thanks to the additional information it can extract from simulation.

During this phase of policy learning, we freeze all but the last layer to preserve the semantic scene information encoded in the learned, domain-invariant representation, f_{cnn}, while enabling the network to adapt to the new downstream task of low-level control. We also insert trainable FiLM layer blocks [40] as adapter modules in f_{cnn} to process the language instruction embeddings between the frozen convolution layers. Finally, we include a few fully-connected layers as a policy head to process the image feature, $f_{cnn}(I_t)$, and proprioceptive state, s_t, and train the resulting policy π with BC loss to predict the mean and standard deviation of a multivariate Gaussian action distribution, as described below.

Let our training dataset $\mathcal{D}_{BC} = \mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$ be a set of demonstrations τ^d, for domain $d \in \{\text{source, target}\}$. As explained in Sec. III, each demonstration is a sequence of tuples $x_t = \big(I_t^d, s_t^d, a_t^d, l_{task}\big)$ containing the image observation, proprioceptive state, language instruction for the task, and action at timestep t. We train with the following standard BC negative log probability loss [41]:

$$\mathcal{L}_{\pi}(\mathcal{D}_{BC}) = \frac{1}{B} \sum_{\substack{x_t \sim \tau^d \\ \tau^d \sim \mathcal{D}_{BC}}} -\log \pi\left(a_t^d \middle| f_{cnn}(I_t^d), s_t^d, l_{task}\right) \quad (3)$$

where B denotes the batch size.

The policy is trained on $k+1$ tasks: k from $\mathcal{D}^{\text{s}}$ (thousands of trajectories per task) and 1 from $\mathcal{D}^{\text{t}}_{\text{target}}$ (≤ 100 trajectories, see Sec. V). In each batch, we sample m tasks uniformly at random from the $k+1$ tasks, and then query $\mathcal{D}_{BC}$ for a fixed number of transitions from trajectories for each of the m selected tasks.

We hypothesize that cross-domain image-language pretraining (Sec. IV-B) improves policy learning because it helps ensure that image observations of different domains depicting semantically similar scenes map into similar regions of the learned embedding space. This accelerates learning not only on $\mathcal{D}^{\text{s}}$ data but also helps alleviate data scarcity in $\mathcal{D}^{\text{t}}_{\text{target}}$ because the pretrained image backbone encodes $\mathcal{D}^{\text{t}}_{\text{target}}$ images into an in-distribution region of the learned image embedding space, alleviating common issues with visual distribution shift and enabling our method to leverage simulation data to compensate for the lack of real-world action-labeled data, improving sim2real transfer.

V. Experimental Setup

We evaluate Lang4Sim2Real in two settings: a `sim2sim` setting where we test the transfer abilities between two simulated domains with visual and physical differences, and the `sim2real` setting, where the few shot IL is defined in the real world and we use simulation to address the data scarcity. `Sim2sim` serves as a platform to evaluate in depth the behavior of Lang4Sim2Real with a fully controlled domain gap, while `sim2real` is our setting of interest for this work. We will use three task suites that we explain below. See Figure 5 in the Appendix for detailed frame rollouts of each task. In a slight overload of notation from Sec. III, here we use $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}$ to denote the source and target domains, respectively.

A. `Sim2Sim` and `Sim2Real` Environment Differences

In `sim2sim`, $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}$ are both sim environments with large differences in camera point-of-view (third person vs. first person), joint friction, and damping. In the `sim2real` setting, we employ a setup with a wide sim2real gap that we aim to bridge using language that includes differences in control frequency, task goals, visual observation appearance, objects, and initial positions. More details between the two environments in `sim2sim` and `sim2real` can be found in Appendix IX-F.

B. Evaluation Metrics

For all `sim2sim` and `sim2real` experiments, we measure task success rate. In `sim2real`, this is calculated through ten evaluation trials for each of two seeds per task, for a total of 20 trials per table entry. In each set of ten trials, we place the

object in the same ten initial positions and orientations, evenly distributed through the range of valid initial object positions. In `sim2sim`, we also run two seeds per setting and take a success rate averaged over 720 trials between the two seeds in the final few hundred epochs of training.

C. Data

1) Environments: For each of our tasks, we design simulation environments on top of Robosuite [66] in Mujoco [56]. For the real environment, we use Operational Space Control [27] to control the position of the end-effector of the robot in Cartesian space. In both simulation and real, we work with a 7-DOF Franka Emika Panda arm and use a common action space consisting of the continuous xyz delta displacement and a continuous gripper closure dimension (normalized from $[-1, 0]$). The robot proprioception space is 22-dimensional, consisting of the robot's xyz end-effector position, gripper state, and sine and cosine transformations of the 7 joint angles. The image observation space is 128×128 RGB images.

2) Overview of Tasks: For each task suite, we collect data from simulated domain $\mathcal{D}^{\text{s}}$ and real target domain $\mathcal{D}^{\text{t}}$ (for `sim2real`) or sim target domain $\mathcal{D}^{\text{t}}$ (for `sim2sim`). All demonstrations in sim and real are collected with a scripted policy (see Appendix for further details). Sim trajectories range from 200-320 timesteps long, operated at 50 Hz, while real trajectories run at 2 Hz and range from 18-45 timesteps. Our three task suites allow us to test the effectiveness of Lang4Sim2Real for sim2real in a wide variety of control problems ranging from simple stacking in task suite 1, to multi-step long-horizon pick and place in task suite 2, to deformable, hard-to-simulate objects in task suite 3.

D. Task Suite 1: Stack Object

In our first suite of tasks, the robot must move an object to a target. In the simulated domain $\mathcal{D}^{\text{s}}$, the target is on top of a wooden coaster, and there are four objects: milk carton, soda can, bread, and cereal box, which correspond to the four tasks. Both the object and coaster positions are randomized over the entire workspace. We collect and train on 400 demonstrations per task (1600 total) as our $\mathcal{D}^{\text{s}}$ simulation data.

1) ***sim2sim****:* For `sim2sim` experiments on this task suite, we define a new $\mathcal{D}^{\text{t}}$ simulated environment with differences from $\mathcal{D}^{\text{s}}$ as enumerated in Sec. V-A. Policies are trained with the 1600 $\mathcal{D}^{\text{s}}$ demonstrations and 100 target task $\mathcal{D}^{\text{t}}_{\text{target}}$ demonstrations.

2) ***sim2real****:* For `sim2real`, $\mathcal{D}^{\text{s}}$ remains the same as `sim2sim`. $\mathcal{D}^{\text{t}}$ is a real world environment in which the object is randomly placed on the left mat and the target task $\mathcal{D}^{\text{t}}_{\text{target}}$ is to move the object onto the right mat and open the gripper by the end of 20 timesteps.

E. Task Suite 2: Multi-step Pick and Place

Our second suite of tasks is longer-horizon. In simulation, the robot must first put an object in the pot, then grasp the pot by its handle and move it onto the stove. We categorize this as a 2-step pick-and-place task. We use the same four object-task mappings from Sec. V-D. The object, pot, and stove locations are all randomized within a quadrant of the workspace. Since this task is longer horizon, we train on more data—$1,400$ trajectories per task in $\mathcal{D}^{\text{s}}$.

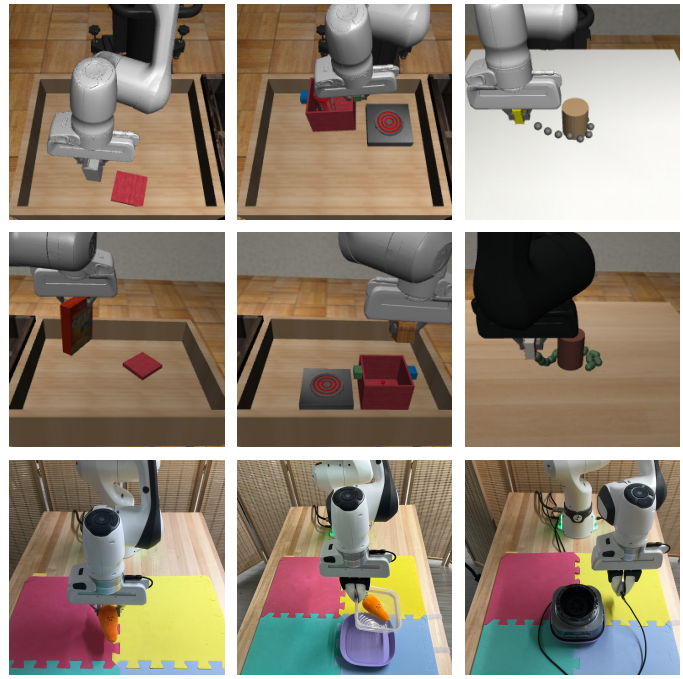

Fig. 3. The columns depict the three task suites while each row represents an image domain. *Rows from Top to Bottom:* Simulation $\mathcal{D}^{\text{s}}$, `sim2sim` $\mathcal{D}^{\text{t}}_{\text{target}}$, `sim2real` $\mathcal{D}^{\text{t}}_{\text{target}}$. *Columns from Left to Right:* Stack Object, Multi-step Pick and Place, and Wrap Wire tasks. While similar enough to transfer prior knowledge between them, our $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}$ task versions have a considerable gap (Sec. V-A) that we are able to bridge using language as regularization for the image representations.

1) ***sim2sim****:* Similar to the stacking task, in the `sim2sim` setting, we define a new $\mathcal{D}^{\text{t}}$ environment with differences from $\mathcal{D}^{\text{s}}$ enumerated in Sec. V-A and evaluate over the four tasks when given 100 target-task $\mathcal{D}^{\text{t}}_{\text{target}}$ demonstrations.

2) ***sim2real****:* In the `sim2real` setup, $\mathcal{D}^{\text{s}}$ remains the same, while $\mathcal{D}^{\text{t}}$ is the real task of putting a carrot into a bowl, then putting the bowl onto a plate (see Fig. 3), and ending with the gripper open after 50 timesteps. In addition to success rate (Section V-B), we measure average number of consecutive subtasks completed from the beginning, allowing partial credit if the robot only succeeds in the first step of placing the carrot in the bowl. However, if the robot does not finish the first step but finishes the second step, we do not count this as having completed any subtasks.

F. Task Suite 3: Wrap Wire

Our final suite of tasks involves wrapping a long deformable wire around a fixed object. In simulation $\mathcal{D}^{\text{s}}$, we approximate a wire with a chain of spheres connected with free joints, and the task is to wrap the chain around a fixed cylinder (see Fig. 3). A trajectory is successful if the first link of the chain has traveled $\geq \frac{5\pi}{3}$ radians ($5/6$ths of a full revolution) around the

TABLE I
SIM2REAL: PERFORMANCE BY NUMBER OF REAL WORLD TRAJECTORIES

Method	Action-labeled Data		Stack Object			Multi-step Pick and Place						Wrap Wire		
	Sim	Real	Success Rate (%)			Success Rate (%)			Subtasks Completed			Success Rate (%)		
	$\mathcal{D}^{\text{s}}$	$\mathcal{D}^{\text{t}}_{\text{target}}$	25	50	100	25	50	100	25	50	100	25	50	100
No Pretrain ($\mathcal{D}^{\text{t}}$)	–	✓	20	30	45	0	30	35	0.45	1.05	1.05	20	15	45
No Pretrain ($\mathcal{D}^{\text{s}} + \mathcal{D}^{\text{t}}$)	✓	✓	35	20	55	45	25	55	1.15	1.0	1.4	25	20	20
MMD	✓	✓	25	35	**80**	20	10	35	0.8	0.9	1.1	5	10	20
Domain Random.	✓	✓	40	60	40	10	10	25	0.7	0.6	0.7	0	0	0
ADR+RNA	✓	✓	35	30	35	15	25	40	0.85	0.8	1.3	0	10	0
Lang Reg. (ours)	✓	✓	40	**75**	**80**	**60**	**80**	**90**	**1.45**	**1.8**	**1.9**	**45**	**40**	45
Lang Dist. (ours)	✓	✓	**60**	45	**80**	55	70	75	1.35	1.65	1.6	30	25	**75**
Stage Classif.	✓	✓	40	60	60	50	60	50	**1.45**	1.55	1.5	30	**40**	50
CLIP (frozen)	✓	✓	25	5	15	10	15	40	0.3	0.45	1.0	35	35	30
R3M (frozen)	✓	✓	30	45	65	15	60	55	0.7	1.4	1.5	5	25	25

TABLE II
SIM2SIM: SUCCESS RATE BY TASK (%)

Pretraining	Stack Object					Multi-step Pick and Place					Wrap Wire
	1	2	3	4	avg	1	2	3	4	avg	1
None ($\mathcal{D}^{\text{t}}$ data only)	15.2 ± 6.5	18.9 ± 6.7	31.9 ± 8.5	25.4 ± 9.2	22.9	20.8 ± 7.5	17.7 ± 4.4	16.3 ± 5.0	17.3 ± 8.1	18.0	69.2 ± 8.3
None ($\mathcal{D}^{\text{s}} + \mathcal{D}^{\text{t}}$ data)	22.5 ± 9.2	32.3 ± 9.8	37.9 ± 8.8	29.2 ± 8.3	30.5	28.4 ± 10.9	31.3 ± 10.7	13.9 ± 5.5	27.8 ± 10.2	25.4	82.1 ± 6.8
Lang Reg. (ours)	20.6 ± 8.1	$\mathbf{57.3 \pm 8.1}$	63.1 ± 7.7	$\mathbf{32.5 \pm 6.3}$	43.4	54.0 ± 7.2	$\mathbf{62.5 \pm 12.1}$	76.0 ± 8.7	58.5 ± 9.3	62.8	90.7 ± 5.4
Lang Dist. (ours)	23.8 ± 5.4	$\mathbf{57.3 \pm 10.6}$	66.9 ± 5.6	27.9 ± 10.8	44.0	$\mathbf{65.5 \pm 13.1}$	56.7 ± 9.9	$\mathbf{78.6 \pm 5.1}$	54.4 ± 11.5	**63.8**	90.0 ± 5.0
Stage Classif.	$\mathbf{30.4 \pm 10.4}$	52.7 ± 6.0	$\mathbf{67.5 \pm 8.3}$	27.9 ± 7.1	**44.6**	63.1 ± 9.9	62.1 ± 9.3	55.4 ± 8.5	$\mathbf{67.7 \pm 9.7}$	62.1	$\mathbf{91.4 \pm 3.6}$
CLIP (frozen)	1.7 ± 0.4	1.9 ± 1.9	3.8 ± 2.5	4.0 ± 2.7	2.9	36.1 ± 14.3	39.9 ± 8.9	28.8 ± 8.9	48.4 ± 11.9	38.3	75.6 ± 7.7
R3M (frozen)	4.5 ± 3.3	9.0 ± 4.8	19.8 ± 6.9	15.4 ± 5.4	12.2	49.4 ± 11.6	36.5 ± 11.9	47.0 ± 14.1	56.0 ± 10.0	47.2	90.2 ± 4.4

cylinder. Our simulation data consists of two tasks: wrapping counterclockwise and clockwise. The initial position of the end of the chain is randomized over a region to the left of the cylinder. $\mathcal{D}^{\text{s}}$ contains 400 trajectories per task.

*1) **sim2sim**:* For our $\mathcal{D}^{\text{t}}$ sim environment, we again apply the changes specified in Sec. V-A. We additionally swapped the spheres for capsules and changed the color and texture of the table, robot arm, and objects. This task has a wider sim2sim gap from additional visual and dynamics changes.

*2) **sim2real**:* In our `sim2real` experiments, the target task $\mathcal{D}^{\text{t}}_{\text{target}}$ is to first grasp the plug, then wrap the cord around the base of a blender in the middle of the workspace, and finally put the plug down, similar to what one might do before putting the appliance away. Like the sim environment, we define success if the following two conditions are met: (1) the plug travels $\geq \frac{5\pi}{3}$ radians around the blender, and (2) the plug is placed and the gripper is open at the end of 50 timesteps.

G. Baselines

To evaluate the effectiveness of Lang4Sim2Real, we consider two sets of baselines: non-pretrained baselines where the CNN is initialized from scratch, and baselines with pretrained visual encoders. For the non-pretrained baselines, we examine training with only $\mathcal{D}^{\text{t}}$ data, and training with both $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}$ data. This enables us to understand the benefits of our proposed training procedure. In `sim2real`, we also compare to three popular prior sim2real approaches:

- **MMD** [57], which aims to minimize the distance between the mean embedding of all sim images and all real images of a batch to prevent the real images from being out-of-distribution relative to the sim images.
- **Domain randomization** [55] of the colors, textures, and physics of the $\mathcal{D}^{\text{s}}$ environment.
- Automatic Domain Randomization with Random Network Adversary (**ADR+RNA**) [38], which keeps increasing/decreasing domain randomization bounds depending on the agent's performance, and also introduces a randomly initialized network for each trajectory to inject correlated noise into the agent's action conditioned on the state input.

For the pretrained baselines, we consider two strong foundation models as the visual backbone, CLIP [42] and R3M [37], commonly used visual representations for robotics that are, like our approach, also shaped by language descriptions of images/videos, and add a trainable policy head composed of fully-connected layers sharing the same dimensions as all other methods in our results.

For each task in `sim2real`, we train and evaluate with 25, 50, or 100 $\mathcal{D}^{\text{t}}_{\text{target}}$ demonstrations.

H. Our Method Variants and Ablations

In our evaluations, we compare language regression (Section IV-B1) and language distance (Section IV-B2), the two pretraining variants of our approach. We also ablate away the effects of language on our pretraining approach in a method called "stage classification," where the pretraining task is to predict the stage index of an image (see Section IV-A) instead the language embedding or embedding distance.

VI. Experimental Results

Our results for `sim2sim` experiments are shown in Table II, and the results for `sim2real` are shown in Table I. In both tables, the methods (rows) are grouped into non-pretrained baselines, our method variants and ablations, and pretrained SOTA baselines. In `sim2real`, we additionally include a group of three rows to show the performance of prior sim2real approaches.

A. Experimental Questions and Analysis

Across the three task suites in both `sim2real` and `sim2sim`, our method generally achieves the highest success rates. To further analyze the effectiveness of our method, we pose and investigate the following experimental questions.

What is the impact of our pretraining approach? Our method nearly doubles the success rate of both non-pretrained baselines in most task suites in `sim2real` and `sim2sim`. This indicates that Lang4Sim2Real can bridge a wide sim2real gap. One factor that may allow our method to perform well is that image observations with similar language descriptions may also have similar action labels. In Appendix IX-C, we further investigate this hypothesis with an analysis of the action distributions between images, split by their language descriptions.

Between the non-pretrained baselines, training on $\mathcal{D}^{\text{s}}$ sim demonstrations in `sim2real` provides little benefit on stack object, increases average performance by $\approx 20\%$ on multi-step pick-and-place, but decreases average performance by $\approx 10\%$ on wrap wire. However, in `sim2sim`, it provides a 10-15% increase on most tasks. This suggests that the sim2sim gap is small enough to benefit from using $\mathcal{D}^{\text{s}}$ even without pretraining, but that the sim2real gap is large enough for pretraining to be needed to leverage $\mathcal{D}^{\text{s}}$.

How does our method compare to prior sim2real baselines? Our method outperforms all of the prior sim2real baselines we tested against (second row-group in Table I), which collectively do relatively poorly in most settings, highlighting the difficulty of the sim2real problem in our setup.

MMD averages the best performance across the three sim2real baselines and even achieves competitive performance on the easiest task of stacking an object. However, on the two other more difficult tasks, its performance does not scale well with more trajectories, which we suspect arises from stability issues in trying to push together the mean of all sim and real image embeddings in each batch. Domain randomization only exacerbates the sim2real gap since enabling all randomizations does not move the distribution of simulation trajectories closer to the real world trajectories due to the large visual dissimilarity between our simulation and real environments. ADR+RNA, which only randomizes the environment as much as possible without severely hurting the scripted policy performance, averages slightly better performance than domain randomization, perhaps because the data is less diverse and easier to fit a policy to than the data from full-scale domain randomization.

How does our method compare to prior vision-language pretrained representations? In `sim2real`, our method outperforms both pretrained baselines across the board, including R3M, which is the strongest baseline on stack object and multi-step pick-and-place. When trained on increasing amounts of real-world data, both R3M and CLIP tend to plateau—CLIP performs no better than 40% on any task, R3M has an apparent ceiling of 65%, while our method achieves up to 90%. This suggests that CLIP and R3M do not scale as well as our method when provided more data, despite being pretrained on internet-scale video and image data while our method was pretrained on images from just a few hundred sim and real trajectories.

In `sim2sim`, our method also outperforms R3M and CLIP across the board. Averaging the performance on stacking and multi-step pick-and-place, our method outperforms R3M by 15-30% and CLIP by 25-40%. On the wrap wire task, our method and R3M perform comparably, probably because the task is quite a bit easier for all methods in simulation.

What is the effect of language in learning shared representations? We ablate the effect of language on our pretraining as the "stage classification" row in Tables I and II, as mentioned in Section V-H. In `sim2sim`, we see similar performance in language regression pretraining and stage classification pretraining. However, in `sim2real`, where the domain gap is larger, we see language providing a measurable benefit in all task suites, especially in multi-step pick-and-place, perhaps because pretraining with language leverages similarities in language descriptions between the first and second steps of the pick-and-place task.

How do our two image-language pretraining variants compare? We compare our two pretraining variants introduced in Sections IV-B1 and IV-B2, where language regression directly predicts language embeddings while language distance is encouraged to maintain pairwise distances based on BLEURT similarity scores. Again in `sim2sim`, there is no clear winner between the two, but in `sim2real`, language regression performs better on average. This suggests that when performing language pretraining for visual representations, the more constraining regression loss outperforms the less constraining distance-matching loss on `sim2real` performance.

B. Additional Experimental Questions and Results

Finally, we examine a few questions to better understand the performance of our method under slight changes to the data and problem setup.

What is the effect of pretraining on image-language pairs where the language granularity is reduced? We evaluate

the impact of reduced language granularity on sim2real performance. See Appendix IX-H for results.

How does our method perform if we cannot pretrain directly on image-language pairs from the target task? There are scenarios in which we might not have access to the real-world target task $\mathcal{D}^{\text{t}}_{\text{target}}$ during the pretraining phase, as pretraining is often done without knowledge of the downstream task. To investigate this, we introduce a real-world prior task $\mathcal{D}^{\text{t}}_{\text{prior}}$ that we pretrain on, and use real-world target task data $\mathcal{D}^{\text{t}}_{\text{target}}$ only during imitation learning. The advantage of this problem setup is that we can reuse the same f_{cnn} for multiple downstream real-world target tasks as long as they are sufficiently similar to the real-world prior task. In this modified problem setup, our method still mostly outperforms all baselines, which demonstrates that our method does not overfit to the real-world task it sees during pretraining. See Appendix IX-I for full results.

Can our method be combined with prior large-scale vision-language pretrained networks? We experiment with combining R3M and our method. Results are discussed in Appendix IX-J.

VII. Conclusion

Vision-based policies struggle with distributional shift during sim2real transfer. To address this challenge, we introduced a low-data-regime visual pretraining approach that leverages language to bridge the sim2real visual gap with only 25-100 real-world trajectories with automatically generated language labels. We evaluate the effectiveness of our approach on multi-step long-horizon tasks and hard-to-simulate deformable objects. In the few-shot setting, our approach outperforms state-of-the-art vision-language foundation models and prior sim2real approaches across 3 task suites in both sim2sim and sim2real.

VIII. Limitations and Future Work

One of the main limitations of our work is that the learned representation may have limited generalizability compared to existing pretraining methods that leverage internet-scale data to enable a large degree of generalization. Our approach targets a specific distribution and domain of real-world tasks and operates in the low-data regime for both pretraining and policy learning, so it does not yield general-purpose visual representations that can be applied to a wide distribution of target tasks. Future work could investigate scaling our method to large-scale datasets to further reduce the number of real-world demonstrations needed for effective sim2real transfer.

Our method also assumes that the template language descriptions used by the automatic labelling process describe similar aspects of images across the two domains, and may perform worse if the language between sim and real described images at extremely different levels of granularity. Furthermore, our approach relies on segmenting all trajectories of a task into stages of a certain granularity so that the associated template language is diverse enough to prevent the learned visual representation from mapping the entire input image distribution to a collapsed point. On contact-rich tasks involving continuous motions or complex object deformations, it may be harder to segment a trajectory and label these segments with language.

Another avenue for future work involves exploring sim2real by combining existing pretraining approaches such as time-contrastive learning and masked image modeling in conjunction with the language-based pretraining we propose, as adding temporal or masked prediction terms to the objective may enable more fine-grained representations that complement the coarseness of language.

IX. Acknowledgements

We would like to thank members of the RobIn Lab at UT Austin for their valuable suggestions and help with debugging real robot issues. This research was supported by NSF NRI Grant IIS-1925082.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as I can and not as I say: Grounding language in robotic affordances. In *arXiv preprint arXiv:2204.01691*, 2022.

[2] Bo Ai, Zhanxin Wu, and David Hsu. Invariance is key to generalization: Examining the role of representation in sim-to-real transfer for visual navigation. *arXiv preprint arXiv:2310.15020*, 2023.

[3] OpenAI: Marcin Andrychowicz, Bowen Baker, Maciek Chociej, Rafal Jozefowicz, Bob McGrew, Jakub Pachocki, Arthur Petron, Matthias Plappert, Glenn Powell, Alex Ray, et al. Learning dexterous in-hand manipulation. *The International Journal of Robotics Research*, 39 (1):3–20, 2020.

[4] Konstantinos Bousmalis, Nathan Silberman, David Dohan, Dumitru Erhan, and Dilip Krishnan. Unsupervised pixel-level domain adaptation with generative adversarial networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3722–3731, 2017.

[5] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, et al. Rt-1: Robotics transformer for real-world control at scale. *arXiv preprint arXiv:2212.06817*, 2022.

[6] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Xi Chen, Krzysztof Choromanski, Tianli Ding, Danny Driess, Avinava Dubey, Chelsea Finn, et al. Rt-2: Vision-language-action models transfer web knowledge to robotic control. *arXiv preprint arXiv:2307.15818*, 2023.

[7] Boyuan Chen, Fei Xia, Brian Ichter, Kanishka Rao, Keerthana Gopalakrishnan, Michael S. Ryoo, Austin Stone, and Daniel Kappler. Open-vocabulary queryable scene representations for real world planning. In *arXiv preprint arXiv:2209.09874*, 2022.

[8] Hao Chen, Ran Tao, Han Zhang, Yidong Wang, Xiang Li, Wei Ye, Jindong Wang, Guosheng Hu, and Marios Savvides. Conv-adapter: Exploring parameter efficient transfer learning for convnets, 2024.

[9] Dima Damen, Hazel Doughty, Giovanni Maria Farinella, Sanja Fidler, Antonino Furnari, Evangelos Kazakos, Davide Moltisanti, Jonathan Munro, Toby Perrett, Will Price, and Michael Wray. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[11] Linxi Fan, Guanzhi Wang, Yunfan Jiang, Ajay Mandlekar, Yuncong Yang, Haoyi Zhu, Andrew Tang, De-An Huang, Yuke Zhu, and Anima Anandkumar. Minedojo: Building open-ended embodied agents with internet-scale knowledge. *Advances in Neural Information Processing Systems*, 35:18343–18362, 2022.

[12] Prasoon Goyal, Scott Niekum, and Raymond Mooney. Using natural language for reward shaping in reinforcement learning. 2019. URL https://arxiv.org/abs/1903.02020.

[13] Prasoon Goyal, Scott Niekum, and Raymond Mooney. Pixl2r: Guiding reinforcement learning using natural language by mapping pixels to rewards. 2020. URL https://arxiv.org/abs/2007.15543.

[14] Raghav Goyal, Samira Ebrahimi Kahou, Vincent Michalski, Joanna Materzynska, Susanne Westphal, Heuna Kim, Valentin Haenel, Ingo Fruend, Peter Yianilos, Moritz Mueller-Freitag, et al. The" something something" video database for learning and evaluating visual common sense. In *Proceedings of the IEEE international conference on computer vision*, pages 5842–5850, 2017.

[15] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, et al. Ego4d: Around the world in 3,000 hours of egocentric video. *arXiv preprint arXiv:2110.07058*, 2021.

[16] Agrim Gupta, Stephen Tian, Yunzhi Zhang, Jiajun Wu, Roberto Martín-Martín, and Li Fei-Fei. Maskvit: Masked visual pre-training for video prediction. *arXiv preprint arXiv:2206.11894*, 2022.

[17] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *arXiv preprint arXiv:1512.03385*, 2015.

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[19] Daniel Ho, Kanishka Rao, Zhuo Xu, Eric Jang, Mohi Khansari, and Yunfei Bai. Retinagan: An object-aware approach to sim-to-real transfer. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pages 10920–10926. IEEE, 2021.

[20] Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, Pierre Sermanet, Noah Brown, Tomas Jackson, Linda Luu, Sergey Levine, Karol Hausman, and Brian Ichter. Inner monologue: Embodied reasoning through planning with language models. In *arXiv preprint arXiv:2207.05608*, 2022.

[21] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12627–12637, 2019.

[22] Eric Jang, Alex Irpan, Mohi Khansari, Daniel Kappler, Frederik Ebert, Corey Lynch, Sergey Levine, and Chelsea Finn. BC-z: Zero-shot task generalization with robotic imitation learning. In *5th Annual Conference on Robot Learning*, 2021. URL https://openreview.net/forum?id=8kbp23tSGYv.

[23] Yunfan Jiang, Agrim Gupta, Zichen Zhang, Guanzhi Wang, Yongqiang Dou, Yanjun Chen, Li Fei-Fei, Anima Anandkumar, Yuke Zhu, and Linxi Fan. Vima: General robot manipulation with multimodal prompts. *arXiv*, 2022.

[24] Ya Jing, Xuelin Zhu, Xingbin Liu, Qie Sima, Taozheng Yang, Yunhai Feng, and Tao Kong. Exploring visual pre-training for robot manipulation: Datasets, models and methods. In *2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 11390–11395. IEEE, 2023.

[25] Siddharth Karamcheti, Megha Srivastava, Percy Liang, and Dorsa Sadigh. Lila: Language-informed latent actions. In *5th Annual Conference on Robot Learning*, 2021. URL https://arxiv.org/pdf/2111.03205.

[26] Manuel Kaspar, Juan D Muñoz Osorio, and Jürgen Bock. Sim2real transfer for reinforcement learning without dynamics randomization. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 4383–4388. IEEE, 2020.

[27] 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):43–53, 1987.

[28] Michael Laskin, Aravind Srinivas, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[29] Shilong Liu, Zhaoyang Zeng, Tianhe Ren, Feng Li, Hao Zhang, Jie Yang, Chunyuan Li, Jianwei Yang, Hang Su, Jun Zhu, and Lei Zhang. Grounding dino: Marrying dino with grounded pre-training for open-set object detection, 2023.

[30] Corey Lynch and Pierre Sermanet. Language conditioned imitation learning over unstructured data. *Robotics: Science and Systems*, 2021. URL https://arxiv.org/abs/2005.07648.

[31] Yecheng Jason Ma, William Liang, Vaidehi Som, Vikash Kumar, Amy Zhang, Osbert Bastani, and Dinesh Jayaraman. Liv: Language-image representations and rewards for robotic control. *arXiv preprint arXiv:2306.00958*, 2023.

[32] Jan Matas, Stephen James, and Andrew J Davison. Sim-to-real reinforcement learning for deformable object manipulation. In *Conference on Robot Learning*, pages 734–743. PMLR, 2018.

[33] Oier Mees, Lukas Hermann, Erick Rosete-Beas, and Wolfram Burgard. Calvin: A benchmark for language-conditioned policy learning for long-horizon robot manipulation tasks, 2021. URL https://arxiv.org/abs/2112.03227.

[34] Oier Mees, Lukas Hermann, and Wolfram Burgard. What matters in language conditioned imitation learning. *arXiv preprint arXiv:2204.06252*, 2022.

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

[36] Suraj Nair, Eric Mitchell, Kevin Chen, Brian Ichter, Silvio Savarese, and Chelsea Finn. Learning language-conditioned robot behavior from offline data and crowd-sourced annotation. In *5th Annual Conference on Robot Learning*, 2021. URL https://arxiv.org/pdf/2109.01115.

[37] Suraj Nair, Aravind Rajeswaran, Vikash Kumar, Chelsea Finn, and Abhinav Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

[38] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand, 2019.

[39] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[40] Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron C. Courville. Film: Visual reasoning with a general conditioning layer. In *AAAI*, 2018.

[41] Dean Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Conference on Neural Information Processing Systems (NeurIPS)*, 1988.

[42] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, et al. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.

[43] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, pages 416–426. PMLR, 2023.

[44] Shreyas Sundara Raman, Vanya Cohen, David Paulius, Ifrah Idrees, Eric Rosen, Ray Mooney, and Stefanie Tellex. Cape: Corrective actions from precondition errors using large language models. In *2nd Workshop on Language and Robot Learning: Language as Grounding*, 2023.

[45] Kanishka Rao, Chris Harris, Alex Irpan, Sergey Levine, Julian Ibarz, and Mohi Khansari. Rl-cyclegan: Reinforcement learning aware simulation-to-real. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11157–11166, 2020.

[46] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999.

[47] Thibault Sellam, Dipanjan Das, and Ankur P Parikh. Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*, 2020.

[48] Younggyo Seo, Danijar Hafner, Hao Liu, Fangchen Liu, Stephen James, Kimin Lee, and Pieter Abbeel. Masked world models for visual control. In *Conference on Robot Learning*, pages 1332–1344. PMLR, 2023.

[49] Rutav Shah, Roberto Martín-Martín, and Yuke Zhu. Mutex: Learning unified policies from multimodal task specifications. *arXiv preprint arXiv:2309.14320*, 2023.

[50] Lin Shao, Toki Migimatsu, Qiang Zhang, Karen Yang, and Jeannette Bohg. Concept2robot: Learning manipulation concepts from instructions and human demonstrations. In *Proceedings of Robotics: Science and Systems (RSS)*, 2020.

[51] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *Proceedings of the 5th Conference on Robot Learning (CoRL)*, 2021.

[52] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Perceiver-actor: A multi-task transformer for robotic manipulation. *Conference on Robot Learning*, 2022.

[53] Andrew Silva, Nina Moorman, William Silva, Zulfiqar Zaidi, Nakul Gopalan, and Matthew Gombolay. Lanconlearn: Learning with language to enable generalization in multi-task manipulation. In *IEEE Robotics and Automa-*

tion Letters, 2021.

[54] Shagun Sodhani, Amy Zhang, and Joelle Pineau. Multi-task reinforcement learning with context-based representations. *arXiv preprint arXiv:2102.06177*, 2021.

[55] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. Domain randomization for transferring deep neural networks from simulation to the real world. In *2017 IEEE/RSJ international conference on intelligent robots and systems (IROS)*, pages 23–30. IEEE, 2017.

[56] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[57] Eric Tzeng, Judy Hoffman, Ning Zhang, Kate Saenko, and Trevor Darrell. Deep domain confusion: Maximizing for domain invariance, 2014.

[58] Che Wang, Xufang Luo, Keith Ross, and Dongsheng Li. Vrl3: A data-driven framework for visual deep reinforcement learning. *Advances in Neural Information Processing Systems*, 35:32974–32988, 2022.

[59] Wenhui Wang, Furu Wei, Li Dong, Hangbo Bao, Nan Yang, and Ming Zhou. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020.

[60] Albert Yu and Raymond J Mooney. Using both demonstrations and language instructions to efficiently learn robotic tasks. *arXiv preprint arXiv:2210.04476*, 2022.

[61] Wenhao Yu, Jie Tan, C Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *arXiv preprint arXiv:1702.02453*, 2017.

[62] Zhecheng Yuan, Zhengrong Xue, Bo Yuan, Xueqian Wang, Yi Wu, Yang Gao, and Huazhe Xu. Pre-trained image encoder for generalizable visual reinforcement learning. *Advances in Neural Information Processing Systems*, 35:13022–13037, 2022.

[63] Xiaohua Zhai, Xiao Wang, Basil Mustafa, Andreas Steiner, Daniel Keysers, Alexander Kolesnikov, and Lucas Beyer. Lit: Zero-shot transfer with locked-image text tuning. *CoRR*, abs/2111.07991, 2021. URL https://arxiv.org/abs/2111.07991.

[64] Tony Zhao, Siddharth Karamcheti, Thomas Kollar, Chelsea Finn, and Percy Liang. What makes representation learning from videos hard for control? 2022. URL https://api.semanticscholar.org/CorpusID:252635608.

[65] Bin Zhu, Bin Lin, Munan Ning, Yang Yan, Jiaxi Cui, HongFa Wang, Yatian Pang, Wenhao Jiang, Junwu Zhang, Zongwei Li, et al. Languagebind: Extending video-language pretraining to n-modality by language-based semantic alignment. *arXiv preprint arXiv:2310.01852*, 2023.

[66] Yuke Zhu, Josiah Wong, Ajay Mandlekar, Roberto Martín-Martín, Abhishek Joshi, Soroush Nasiriany, and Yifeng Zhu. robosuite: A modular simulation framework and benchmark for robot learning. *arXiv preprint arXiv:2009.12293*, 2020.

APPENDIX

A. Scripted Policy for Real-World Data Collection

Algorithm 1 Scripted Wrap Wire

```
centerPos ← blender center position
placeAttempted ← False
targetDistToCenter ← 0.15
numTimesteps ← 45
direction ← true if clockwise, false if counterclockwise
for t in [0, numTimesteps) do
    wirePos ← position of the graspable part of the wire
    eePos ← end effector position
    pickPosDist ← ‖eePos − wirePos‖_2
    done ← is wrapped > 11π/6 from the start to end of wire around centerPos in direction
    if placeAttempted then
        action ← 0
    else if object not grasped AND pickPosDist > distThresh then
        // Move toward wire
        action ← wirePos − eePos
    else if object not grasped then
        // gripper is very close to wire
        action ← pickPos − eePos
        close gripper  // Object is in gripper
    else if wire not lifted then
        action ← [0, 0, 1]  // Move up
    else if not done then
        relPos ← eePos − centerPos
        distToCenter ← ‖relPos‖_2
        normRelPos ← (relPos/distToCenter) ∗ targetDistToCenter
        actionMaintainDistance ← relPos ∗ (targetDistToCenter − distToCenter)  // move toward/away from center
        actionMoveTangent ← [−normRelPos[1], normRelPos[0], 0.0]  // Move tangent to the blender
        if direction then
            actionMoveTangent ← actionMoveTangent ∗ −1
        end if
        action ← actionMaintainDistance + actionMoveTangent
    else
        action ← open gripper  // Drop wire
        placeAttempted ← True
    end if
end for
```

Algorithm 2 Scripted Pick and Place Function

```
function PICKPLACE(pickPos, dropPos, distThresh, placeAttempted)
    eePos ← end effector position
    dropPosDist ← ‖eePos − dropPos‖_2
    pickPosDist ← ‖eePos − pickPos‖_2
    if placeAttempted then
        action ← 0
    else if object not grasped AND pickPosDist > distThresh then
        // Move toward target object
        action ← pickPos − eePos
    else if object not grasped then
        // gripper is very close to object
        action ← (pickPos − eePos, close gripper)  // Object is in gripper
    else if object not lifted then
        // Move gripper upward to avoid hitting other objects/containers
        action ← [0, 0, 1]
    else if dropPosDist > distThresh then
        // Move toward target container
        action ← dropPos − eePos
    else
        action ← open gripper  // Object falls into container
        placeAttempted ← True
    end if
    noise ∼ N(0, 0.1)
    action ← action + noise
return action, placeAttempted
end function
```

Algorithm 3 Stack Object

```
pickPos ← target object position
dropPos ← target container position
numTimesteps ← 18
distThresh ← 0.02
placeAttempted ← False
for t in [0, numTimesteps) do
    action, placeAttempted ← PICKPLACE(pickPos, dropPos, distThresh, placeAttempted)
    s′ ← env.step(action)
end for
```

Algorithm 4 Scripted 2-step Pick and Place

```
pickPos ← [object position, first container position]
dropPos ← [first container position, second container position]
numTimesteps ← 45
distThresh ← 0.02
placeAttempted ← [False, False]
s_i ← 0  // step index (starts at 0, and increments to 1 when first pick-place step is complete)
stepCompleted ← [False, False]
for t in [0, numTimesteps) do
    action, placeAttempted[s_i] ← PICKPLACE(pickPos[s_i], dropPos[s_i], distThresh, placeAttempted[s_i])
    if stepIsSuccessful(s_i) AND not stepsCompleted[s_i] then
        stepsCompleted[s_i] ← True
        s_i ← 1
    end if
    s′ ← env.step(action)
end for
```

B. Detailed Policy Network Architecture & Hyperparameters

For the policy backbone, we use a ResNet-18 architecture but made changes to the strides and number of channels to adapt the network to our $128 \times 128 \times 3$ image size. Hyperparameters are shown in Table V. A detailed layer-by-layer architecture figure of our policy is shown in Figure 6. During policy training, only the last CNN layer, FiLM blocks, and policy head (FC layers) are finetuned, while all other layers are kept frozen.

C. Does Language Similarity Imply Action Distribution Similarity?

We hypothesize that one of the ways language is an effective bridge for sim2real transfer is that the sim and real action distributions of the demonstrations are similar when the image observations have similar language descriptions. Figure 4 shows the action distribution similarities between sim and real when the language descriptions are similar (top row), and when the language descriptions are different (bottom row). Each column represents a component of the action distribution. We plot three components: z-axis actions, xy-magnitude (which is the $\ell 2$ norm of the (x, y) action dimensions), and the gripper dimension. We observe that action distributions are indeed more similar for images described by similar language than for images described by different language.

D. Task and Data Details

Figure 5 provides film strips of trajectories from the source domain data $\mathcal{D}^{\text{s}}$, target domain prior task data $\mathcal{D}^{\text{t}}_{\text{prior}}$, and target domain target task data $\mathcal{D}^{\text{t}}_{\text{target}}$, for each of the three task suites.

TABLE III
LANGUAGE DESCRIPTION TEMPLATES OF IMAGE OBSERVATIONS

Task	**Template String**
Pick and Place	gripper open, reaching for {*objName*}, out of {*contName*}
	gripper open, moving down over {*objName*}, out of {*contName*}
	gripper closing, with {*objName*}, out of {*contName*}
	gripper closed, moving up with {*objName*}, out of {*contName*}
	gripper closed, moving sideways with {*objName*}, out of {*contName*}
	gripper closed, with {*objName*}, above {*contName*}
	gripper open, dropped {*objName*}, in {*contName*}
Wrap Wire	gripper open, reaching for {*graspObjName*}
	gripper open, moving down over {*graspObjName*}
	gripper closing around {*graspObjName*}
	gripper closed, moving up with {*graspObjName*}
	{*direction*} left
	{*direction*} front
	{*direction*} right
	{*direction*} back
	gripper open, above {*graspObjName*} with {*flexWraparoundObjName*} fully wrapped
	gripper open, above {*graspObjName*} with {*flexWraparoundObjName*} fully unwrapped
Variable	**Possible Values**
objName	milk, bread, can, cereal, pot, carrot, bowl, bridge
contName	coaster, pot, stove, bowl, plate
flexWraparoundObjName	beads, cord, ethernet cable
graspObjName	last bead, white plug, bridge
direction	clockwise, counterclockwise

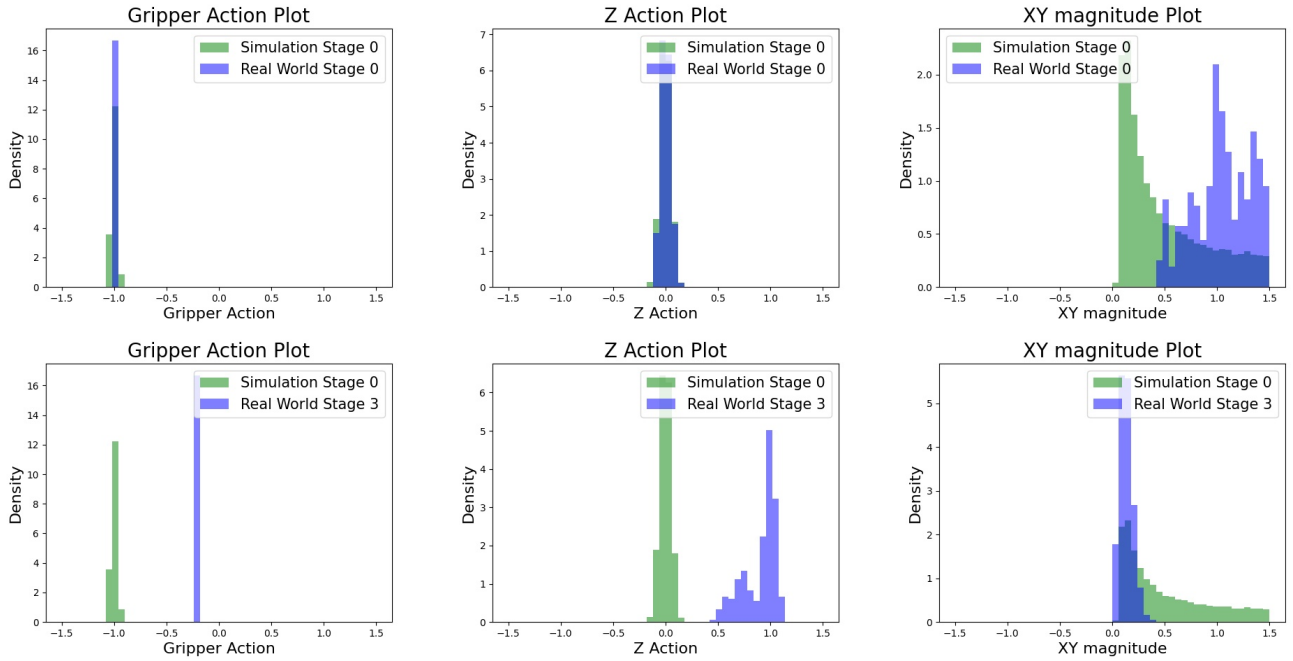

Fig. 4. These plots show the action distribution of demonstrations across both sim and real, broken down by each component of the action: xy-action magnitude, z-axis actions, and gripper actions. The first row shows simulation (green) and real world (blue) action distributions for images described by similar language. The second row shows the same distribution of simulation actions (green) as in the first row, but compared with real-world action distributions from images labeled with very different language from the sim actions (blue). Notably, the action distributions are generally similar for images with similar language (first row), and different for images with different language (second row). This suggests that pretraining our CNN on language embedding prediction benefits downstream policy learning because it allows the domain-invariant learned representations to tap into similar action distributions for completing a task.

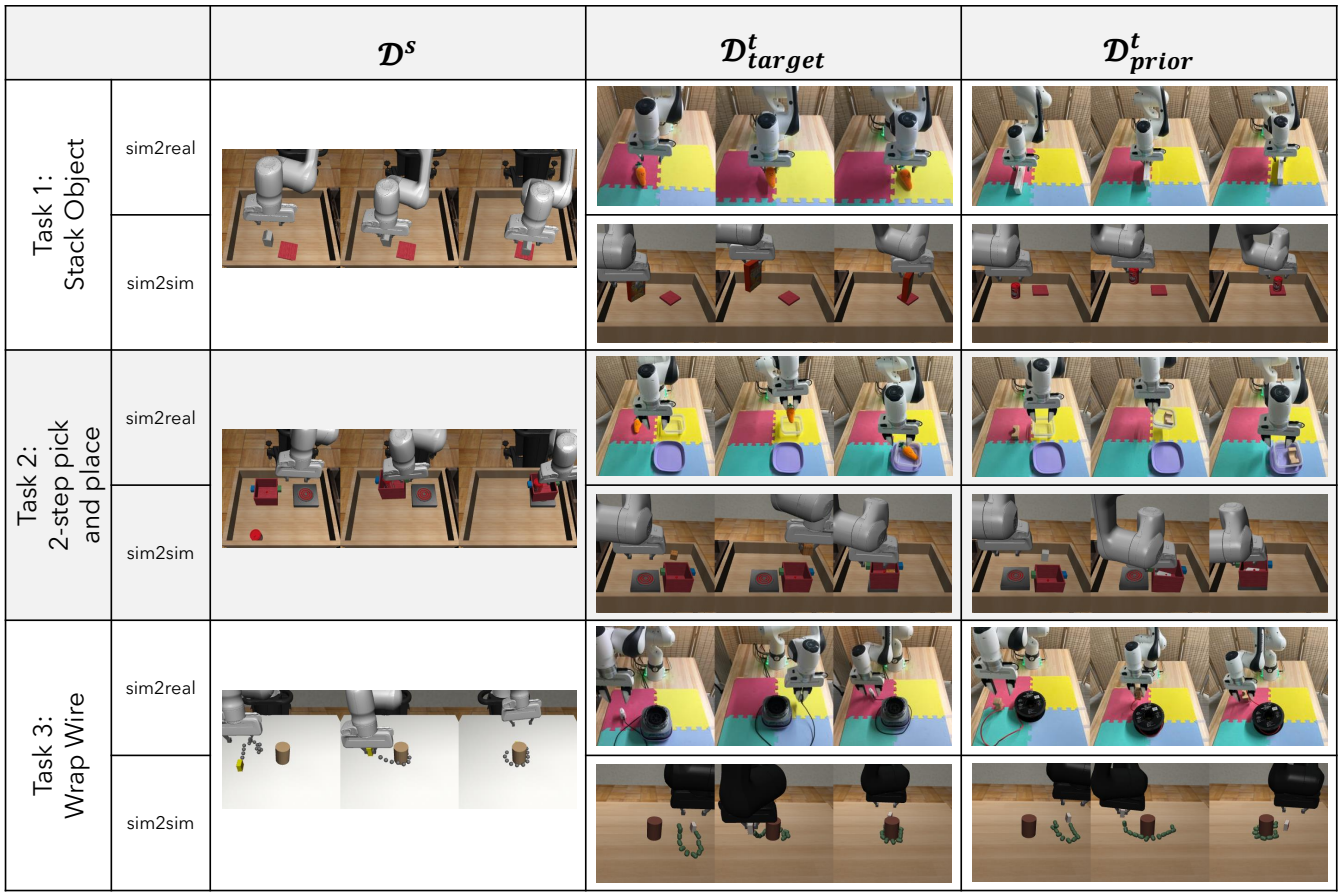

Fig. 5. This table builds on Figure 3 and depicts the 3 datasets for each task with filmstrips. The rows show the three task suites while each column represents one of the three datasets we use during pretraining or policy learning. Our main results in Tables I and II use $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$ for pretraining and policy learning, whereas our results in Table IV use $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{prior}}$ for pretraining and $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$ for policy learning. This table shows the visual differences between sim and real, as well as the task in $\mathcal{D}^{\text{t}}_{\text{prior}}$ versus $\mathcal{D}^{\text{t}}_{\text{target}}$.

E. Training Hyperparameters

Table VI shows our BC training hyperparameters.

In each training iteration, we sample 4 random tasks from our training buffer and get 57 samples per task, for a total batch size of 228.

TABLE VI
IMITATION LEARNING HYPERPARAMETERS.

Attribute	Value
Number of Tasks per Batch	4
Batch Size per Task	57
Learning Rate	3×10^{-4}

F. Sim2Sim and Sim2Real Differences

In our `sim2sim` experiments, $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}$ are both sim environments with the following differences:

1) **Camera point-of-view**: $\mathcal{D}^{\text{s}}$ image observations are third person (looking toward the robot), and $\mathcal{D}^{\text{t}}$ image observations are first person (over the shoulder), a large change of viewing angle.
2) **Friction and Damping**: Joint friction and damping coefficients are $5\times$ and $50\times$ higher in $\mathcal{D}^{\text{t}}$ than $\mathcal{D}^{\text{s}}$, which significantly changes the dynamics.

In our `sim2real` experiments, $\mathcal{D}^{\text{s}}$ in sim and $\mathcal{D}^{\text{t}}$ in real have the following differences:

1) **Control frequency**: The simulated $\mathcal{D}^{\text{s}}$ policy runs at 50Hz while the real world $\mathcal{D}^{\text{t}}$ policy runs at 2Hz.
2) **Objects**: The objects on the scene in each task differ between simulation and real data, except the robot itself.
3) **Visual Observation**: Backgrounds and camera angles are markedly different between the two domains.
4) **Initial positions**: The initial object and robot positions are different across sim and real.

G. Labeling Image Observations with Language

*1) **Language labeling during Scripted Policy**:* We automatically label image observations with language descriptions during the scripted policy data collection process. Each image is assigned a stage number based on the if-case of the scripted policy, which corresponds to a semantic positional arrangement between the gripper and the relevant objects on

TABLE IV
SIM2REAL: PERFORMANCE IN $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}} \cup \mathcal{D}^{\text{t}}_{\text{prior}}$ SETTING BY NUMBER OF TARGET TASK DEMONSTRATIONS

Method	Action-labeled Data			Stack Object			Multi-step Pick and Place						Wrap Wire		
	Sim	Real		Success Rate (%)			Success Rate (%)			Subtasks Completed			Success Rate (%)		
	$\mathcal{D}^{\text{s}}$	$\mathcal{D}^{\text{t}}_{\text{target}}$	$\mathcal{D}^{\text{t}}_{\text{prior}}$	25	50	100	25	50	100	25	50	100	25	50	100
No Pretrain ($\mathcal{D}^{\text{t}}$ data only)	–	✓	✓	45	30	65	40	20	30	1.15	0.9	1.15	25	45	35
No Pretrain ($\mathcal{D}^{\text{s}} + \mathcal{D}^{\text{t}}$ data)	✓	✓	✓	20	55	25	45	30	50	1.25	1.2	1.4	15	30	30
MMD	✓	✓	✓	35	30	40	70	45	35	**1.65**	1.25	1.2	15	0	20
Domain Random.	✓	✓	✓	25	45	60	15	15	20	0.9	0.55	0.85	0	5	5
ADR+RNA	✓	✓	✓	15	10	20	50	5	50	1.35	0.7	1.25	15	10	20
Lang Reg. (ours)	✓	✓	–	50	55	**85**	55	**80**	**95**	1.2	**1.8**	**1.95**	25	**50**	55
Lang Dist. (ours)	✓	✓	–	30	65	70	25	50	65	0.95	1.4	1.5	15	25	60
Stage Classif.	✓	✓	–	70	60	70	20	60	85	0.9	1.5	1.8	15	20	**70**
CLIP (frozen)	✓	✓	✓	30	25	35	25	45	35	0.55	0.95	0.95	**35**	40	45
R3M (frozen)	✓	✓	✓	**80**	**70**	80	**75**	75	85	1.6	1.55	1.75	30	25	20

TABLE V
POLICY π HYPERPARAMETERS.

Attribute	Value
Input Height	128
Input Width	128
Input Channels	3
Number of Kernels	[16, 32, 64, 128]
Kernel Sizes	[7, 3, 3, 3, 3]
Conv Strides	[2, 2, 1, 1, 1]
Maxpool Stride	2
Fully Connected Layers	[1024, 512, 256]
Hidden Activations	ReLU
FiLM input size	384
FiLM hidden layers	0
Spatial Softmax Temperature	1.0
Learning Rate	3×10^{-4}
Policy Action Distribution	Multivariate Isotropic Gaussian $\mathcal{N}(\mu, \sigma)$
Policy Outputs	(μ, σ)
Image Augmentation	Random Crops
Image Augmentation Padding	4

the scene. Stage numbers map 1-to-1 to the template language strings shown in Table III.

For example, for the pick-and-place/stack object task, we define 7 stages and 7 corresponding language string templates, where the first stage is when the gripper moves toward a point above the object, the second stage is when the gripper moves downward toward the object, and so on. For the 2-step pick-and-place task, we use 14 stages—2 consecutive lists of the 7 individual pick place string templates, where the object and container variables of each template are filled in with the proper names.

Though our approach to labeling image observations with language was done during demonstration collection, we emphasize that images can be automatically labeled with language in hindsight after demonstrations are collected. For instance, one can run an object detector on the images to estimate the position of the gripper in relation to the scene objects. This information can be used to determine what stage in a pick-and-place trajectory an image observation falls into.

*2) **Alternative Approach: Language labeling with off-the-shelf VLMs**:* To relax the requirement that our automated language labeling process must occur synchronously with a scripted policy collecting demonstrations, we implemented an alternative approach that is decoupled from the demonstration collection process. First, we use an off-the-shelf open-vocabulary object detector model, GroundingDINO [29], to output bounding boxes for the relevant objects on the scene. No finetuning of GroundingDINO is required. Second, we train a CNN-based gripper state predictor to predict the gripper position (x, y, z) as well as whether the gripper is opened or closed in a given image. This network is trained on previously collected (image, gripper position, gripper opened/closed) data from 100 trajectories, and takes one minute to train on a single A5000 GPU. Using these two models, we can get the gripper state and position relative to the objects, enabling us to predict a stage number that corresponds fairly closely with the actual stage number as outputted by our scripted policy. Finally, we verified that training our method on VLM-derived language annotations does not degrade performance. We performed image-language pretraining with language labels from either labeling method and tested on 2-step pick-and-place with 100 real-world trajectories. Both methods achieve 90% success rate averaged over 2 seeds.

H. Impact of Language Granularity on Performance

To examine the impact of decreasing language granularity on `sim2real` performance, we segment the trajectories into fewer and fewer stages, until the extreme case where the entire trajectory has only a single stage, which means that all images across all trajectories of a task have the same exact language description embedding. The language descriptions we use for each stage, for varying numbers of stages per task, are displayed in Tables VIII (2-step pick-and-place) and IX (wire wrap).

Results are shown in Table VII. The trend is noisy, but in general, decreasing language granularity hurts performance slightly. Still, our method is robust to lower granularity, which

matches our hypothesis that our pretraining approach provides significant performance gains simply by pushing sim and real images into a similar embedding distribution even if the language granularity is extremely coarse.

I. sim2real results with no pretraining on $\mathcal{D}^{\text{t}}_{\text{target}}$

In Tables I and II, we presented results in a setting where we both pretrained and did policy learning on two datasets, $\mathcal{D}^{\text{s}}$ and $\mathcal{D}^{\text{t}}_{\text{target}}$. Sometimes it is unrealistic to assume that during pretraining, we have access to the downstream target task we are ultimately interested in. In such scenarios, it may be more realistic to assume we instead have real-world data for a prior task, $\mathcal{D}^{\text{t}}_{\text{prior}}$. Thus, in this setting, we experiment with pretraining on $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{prior}}$ and training our policy on $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$.

Our method uses extra language labels during pretraining that the baselines do not have access to. While these language labels can be acquired at scale, to compensate for this data advantage, we decided to give all baselines an augmented $\mathcal{D}^{\text{t}}_{\text{prior}}$ dataset that includes action-labeled demonstrations, in addition to the target task, $\mathcal{D}^{\text{t}}_{\text{target}}$. *Note that our method is not given $\mathcal{D}^{\text{t}}_{\text{prior}}$ action-labeled data*: it is trained only on $\mathcal{D}^{\text{t}}_{\text{prior}}$ images with language labels during image-language pretraining (Sec. IV-B) but not during BC policy learning. Therefore, the baselines in a sense serve as upper bounds as they are given $\left|\mathcal{D}^{\text{t}}_{\text{prior}}\right| = 50$ additional action-labeled demonstrations. In other words, during policy learning, the baselines train on action-labeled demonstrations from $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{prior}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$ while ours are only trained on $\mathcal{D}^{\text{s}} \cup \mathcal{D}^{\text{t}}_{\text{target}}$. Results are shown in Table IV.

How different are $\mathcal{D}^{\text{t}}_{\text{prior}}$ and $\mathcal{D}^{\text{t}}_{\text{target}}$? In sim2sim and sim2real for stack object and 2-step pick-and-place, the robot interacts with different objects in the two real-world tasks. Instead of a carrot as in $\mathcal{D}^{\text{t}}_{\text{target}}$, in $\mathcal{D}^{\text{t}}_{\text{prior}}$, the robot interacts with a paper box for the stack object task suite and a rigid toy wooden block for 2-step pick-and-place.

In sim2sim on wire wrap, $\mathcal{D}^{\text{t}}_{\text{prior}}$ contains data of the beads being wrapped clockwise, instead of counterclockwise in $\mathcal{D}^{\text{t}}_{\text{target}}$. In sim2real for wire wrap, the plug, cord, and blender in $\mathcal{D}^{\text{t}}_{\text{target}}$ are replaced by a wooden block, ethernet cable, and spool, respectively, in $\mathcal{D}^{\text{t}}_{\text{prior}}$ data. The differences between $\mathcal{D}^{\text{t}}_{\text{prior}}$ and $\mathcal{D}^{\text{t}}_{\text{target}}$ can be visually examined in Figure 5.

What trends are different between Table IV (with $\mathcal{D}^{\text{t}}_{\text{prior}}$) and Table I (without $\mathcal{D}^{\text{t}}_{\text{prior}}$)? Most of the trends are similar. Re-examining our main experimental questions, we see that our method still nearly doubles the success rate of both non-pretrained baselines, outperforms all three prior sim2real baselines, and that using language regression is important to achieve the most gains from pretraining (language regression outperforms stage classification and language distance, on average). However, in this new problem setting in sim2real, R3M outperforms our method in the lowest data regime with 25 target task demonstrations, perhaps because of the additional 50 $\mathcal{D}^{\text{t}}_{\text{prior}}$ demonstrations that our method does not train on. However, on 50 and 100 trajectories for the longer-horizon multi-step pick and place task, our method achieves higher sim2real performance than the best of either pretrained baseline.

J. Combining R3M with Our Approach

We implemented and evaluated multiple ways to combine R3M with our image-language pretraining to see if it would be possible to leverage the benefits of both R3M's large-scale pretraining and our method's domain-invariant representation learning. Instead of initializing a ResNet from scratch before image-language pretraining, we experiment with using R3M weights and finetuning the last layer, the entire network, or inserted convolutional adapter modules [8]. Finetuning adapters (denoted R3M+adapters) performs the best in sim2sim, matching the performance of our method on 2-step pick-and-place.

Based on this sim2sim performance, we evaluated R3M+adapters on sim2real, but this performed worse than either frozen R3M or our method in sim2real (Table X). We hypothesize that this is because during image-language pretraining on both sim and real images, the trainable adapters learn to pick out features primarily in the simulation images as this is out-of-distribution for R3M which was trained on real-world videos, which enables R3M+adapters to do well in sim2sim but not sim2real.

TABLE VII

SIM2REAL: PERFORMANCE WITH VARYING LANGUAGE GRANULARITY

Method	Multi-step Pick and Place						Wrap Wire		
	Success Rate (%)			Subtasks Completed			Success Rate (%)		
	25	50	100	25	50	100	25	50	100
No Pretrain ($\mathcal{D}^{\text{t}}$)	40	20	30	1.15	0.9	1.15	25	45	35
No Pretrain ($\mathcal{D}^{\text{s}} + \mathcal{D}^{\text{t}}$)	45	30	50	1.25	1.2	1.4	15	30	30
all-stages	**55**	**80**	**95**	**1.2**	**1.8**	**1.95**	**25**	**50**	**55**
half-stages	45	60	65	1.15	1.45	1.55	5	35	25
2-stages	35	45	75	1.05	1.3	1.6	20	**50**	40
1-stage	**55**	65	80	1.3	1.55	1.75	15	15	45
1 stage per domain	10	50	50	0.65	1.3	1.25	15	15	20

TABLE VIII

SIM2REAL: LANGUAGE ANNOTATIONS AND LANGUAGE GRANULARITY ON 2-STEP REAL-WORLD PICK-AND-PLACE

All-stages	**Half-stages**	**2-stage**	**1-stage**
gripper open, reaching for carrot, out of bowl	gripper open, reaching for carrot, out of bowl	picking carrot and putting in bowl	random language embedding
gripper open, moving down over carrot, out of bowl			
gripper closing, with carrot, out of bowl	gripper closing, with carrot, out of bowl		
gripper closed, moving up with carrot, out of bowl	gripper closed, moving up with carrot		
gripper closed, moving sideways with carrot, out of bowl			
gripper closed, with carrot, above bowl			
gripper open, dropped carrot, in bowl	gripper open, dropped carrot, in bowl		
gripper open, reaching for bowl, out of plate	gripper open, reaching for bowl, out of plate	picking bowl and putting in plate	
gripper open, moving down over bowl, out of plate			
gripper closing, with bowl, out of bowl	gripper closing, with bowl, out of plate		
gripper closed, moving up with bowl, out of plate	gripper closed, moving up with bowl		
gripper closed, moving sideways with carrot, out of bowl			
gripper closed, with bowl, above plate			
gripper open, dropped bowl, in plate	gripper open, dropped bowl, in plate		

TABLE IX

SIM2REAL: LANGUAGE ANNOTATIONS AND LANGUAGE GRANULARITY ON WIRE WRAP

All-stages	**half-stages**	**2-stage**	**1-stage**
gripper open, reaching for plug	gripper open, reaching for plug	picking and wrapping beads around cylinder	random language embedding
gripper open, moving down over plug			
gripper closing around plug	gripper closing and lifting plug		
gripper closed, moving up with plug			
counter-clockwise left	counter-clockwise		
counter-clockwise front			
counter-clockwise right			
counter-clockwise back			
clockwise left	clockwise		
clockwise front			
clockwise right			
clockwise back			
gripper open, above plug with wire fully wrapped	gripper open, above blender with wire fully wrapped	beads fully wrapped	
gripper open, above plug with wire fully unwrapped	gripper open, above blender with wire fully unwrapped		

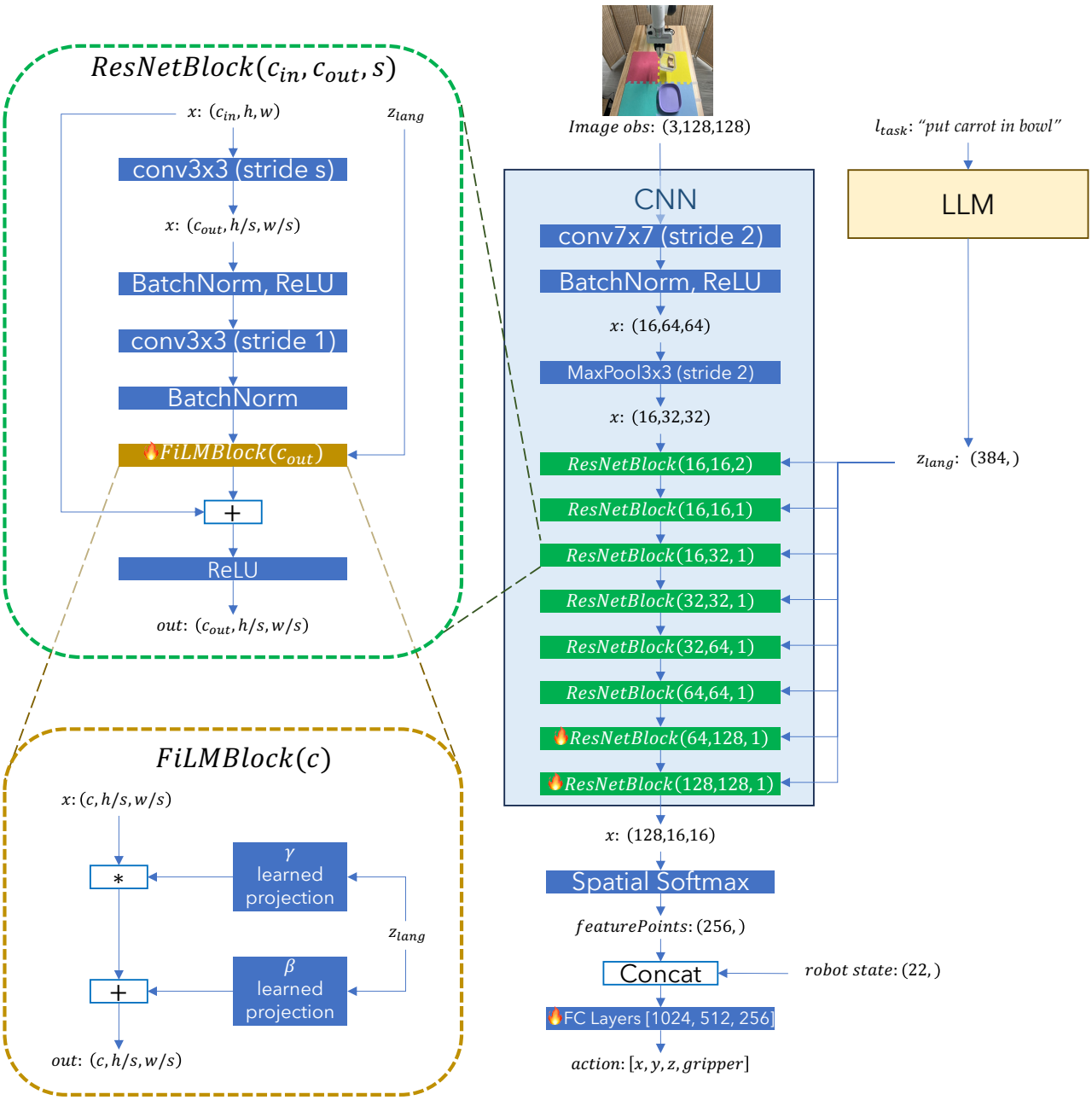

Fig. 6. Detailed Policy Network Architecture. Fire denotes layers trained during policy learning. The early CNN modules are kept frozen to maintain the intermediate representations learned from the pretraining phase.

TABLE X
SIM2REAL: FINETUNING R3M WITH OUR METHOD

Pretraining	Action-labeled Data			Multi-step Pick and Place						Wrap Wire		
	Sim	Real		Success Rate (%)			Subtasks Completed			Success Rate (%)		
	$\mathcal{D}^{s}$	$\mathcal{D}^{t}_{target}$	$\mathcal{D}^{t}_{prior}$	25	50	100	25	50	100	25	50	100
R3M + adapters, Lang Reg.	✓	✓	–	0	30	40	0.7	0.95	1.25	0	5	5
Lang Reg. (ours)	✓	✓	–	55	**80**	**95**	1.2	**1.8**	**1.95**	25	**50**	**55**
R3M (frozen)	✓	✓	✓	**75**	75	85	**1.6**	1.55	1.75	**30**	25	20