Learning to Manipulate Using Diverse Datasets

Sudeep R. Dasari

CMU-RI-TR-24-30

June 2024

The Robotics Institute
Carnegie Mellon University
Pittsburgh, Pennsylvania 15213

Thesis Committee

Thesis document submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy in Robotics

Acknowledgements

This document is the culmination of a 5 year research journey that I couldn't have done without my mentors, labmates, friends, and family. I would like to take a moment now to thank you all. First, I'd like to thank the CMU Robotics Insitute, Meta AI, and the NDSEG fellowship for supporting my research. Next to my advisor Abhinav Gupta – thank you for teaching me how to conduct independent research. Your incisive (and always playful) commentary pushed me to focus on more interesting problems, and encouraged me to be the best version of myself. Abhinav also taught me strong presentation skills, the value of making contrarian research bets, and the thrill of vigorous research debates. On a personal level, I will always remember our fun lab socials/retreats, coffee chats, and mafia games!

I'm also fortunate to have recieved wonderful mentorship from my committee members. Shubham Tulsiani – thank you for reviewing, criticizing, and improving so many of my papers and presentations over the last few years. And thank you for opening my eyes to new vision problems and ways of thinking during the various Smith Hall reading groups. To Deepak Pathak – thank you for pushing me to work on more interesting problems and generate stronger results. I've truly enjoyed all the long research discussions/debates with you and your lab that always helped me question my core assumptions. And to Sergey Levine – thank you for teaching me the fundamentals of machine learning and robotics research, and getting me started down this road as an undergraduate student. I'm always inspired by your focus on generalization over quick and easy results, and hope to always carry that with me.

This PhD took 5 years, and it would've been unbearable without all the encouragement and support from my friends and colleagues. Thank you to Kenny Marino, Senthil Purushwalkam, Adithya Murali, Victoria Dean, Sam Powers, Gaoyue Zhou, Russell Mendonca, Mihir Prabhudesai, Kenny Shaw, Murtaza Dalal, Mohan Kumar Srirama, Jianren Wang, Dibya Ghosh, Katie Kang, Homer Walke, Oier Mees, Giri Anantharaman, Lerrel Pinto, Frederik Ebert, Suraj Nair, Stephen Tian, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Unnat Jain, Vikash Kumar, Chelsea Finn, and many more for being excellent collaborators, lab-mates, and coffee-walk-enjoyers. I'd also like to acknowledge the RoboFantasy League: Yufei Ye, Raunaq Bhirangi, Alex Li, Shikhar Bahl, Helen Jiang, Jason Zhang, Suvansh Sanjeev, Peter Manohar, and Magdalen Dobson for the fun hangouts, and late-afternoon board game sessions every Wednesday. Special shoutout to Jason, Helen, and Shikhar for being the best friends money can't buy and making Pittsburgh feel like home!

Finally, I'd like to thank my family for all their love and encouragement. PhDs are an inheritable disease, and I have to thank my wonderful parents – Drs. Anjali Yeolekar and Venkat Dasari – for inspiring me to work on research and explore my curiosity. I'd like to thank my brother, Shaunak Dasari, for being a life-long partner in crime. And I'd like to acknowledge my amazing fiancée, Varsha Venkat, for always encouraging me and believing in me. Thank you all for everything ❤️

Abstract

Autonomous agents can play games (like Chess, Go, and even Starcraft), they can help make complex scientific predictions (e.g., protein folding), and they can even write entire computer programs, with just a bit of prompting. However, even the most basic physical manipulation skills, like unlocking and opening a door, still remain literally out-of-reach. The key challenge is acquiring the manipulation primitives themselves – there are infinite objects and environments in this world that a robot will have to interact with. Even worse, physics is unforgiving and even small errors can cause a task to fail entirely. In this thesis, I adopt a data-driven approach to address this challenge. Instead of hard-coding or planning actions within a known environment, I will explore methods/algorithms that acquire policies from increasingly scalable sources of offline data. The first section will demonstrate how highly effective policies can be learned from expert demonstrations and high-capacity neural networks. This work establishes the viability of data-driven policy learning for manipulation tasks, but requires the most expensive form of data to collect. Thus, the next part loosens the assumptions and demonstrates how diverse data can be collected across multiple institutions, and be used to boost performance even in specific domains/tasks. The third section pushes this same philosphy even further, and shows how human data can be used to improve robot policies. This is accomplished via representation learning – human data is used to learn robotic representations using various contrastive, self-supervised, and semi-supervised alogrithms that transfer strongly to downstream manipulation tasks. Finally, we look ahead and discuss how these methods can continue to scale by leveraging larger datasets and massive pre-trained vision/language models.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Autonomous agents can play games (like Go, Chess, and Starcraft) at a super-human level [204, 228, 191]. They can help solve complex scientific and engineering problems (like chip design [147] and protein folding [111]), and can even drive autonomously in limited geographic regions (e.g. https://waymo.com/). Now with the advent of Large Language Models [169, 56] (LLMs), we're watching computers automate high-level reasoning and planning tasks. These advances are breathtaking and they have certainly revolutionized the world of bits and pixels. But their impact on the real, physical world has been surprisingly limited. For example, a robot still cannot make a cup of coffee or fold clothes on command. To be clear, the robot can definitely reason (using LLMs) about the physical primitives required to transport coffee, but executing the primitives themselves in the real world remains *literally* out of reach!

But why are these primitives so hard to acquire? The major challenge is that manipulation tasks force the robot to precisely reason about object contact behaviors, which can vary subtly depending on object properties, environment details, and the exact task itself. For example, the robot may need to gently grasp a mug by the handle (when transporting coffee) or stably hold it upside down (when cleaning the mug). Furthermore, it will have to perform these tasks in any kitchen, given any novel mug. It is impossible to inumerate and explicitly plan across all these scenarios. Even if we could, slight differences (e.g., robot wear and tear) during test time could cause catostrophic failures, because the margin of error for these tasks is so small and physics is unforgiving. So what can we do instead?

A good idea is to take inspiration from other AI subfields (like Natural Language Processing [169]), where highly capable models/agents emerged naturally via data-driven learning. They key insight is that these systems can automatically *generalize* to new scenarios and tasks, by automatically learning fundamental skills and world knowledge from vast amounts of training data. If applied to robotics, this capability could allow agents to solve the novel test scenarios outlined in the previous section. However, collecting enough data remains a fundamental challenge. After all, both vision and language models are often trained on vast collections of data scraped from the internet. In contrast, robot learning researchers often have to collect their own data, and are thus limited to simple, lab scenarios.

This thesis seeks to scale both the data sources and the learning algorithms and models used in robotics. More precisely, the objective is to learn a goal-conditioned robotic policy $a_t \ \pi(*|o_t, g)$ (where a_t is robot actions, o_t is observations, and g is goal vector), using a mix of the robot's "prior experience" (i.e., expert demonstrations and autonomous rollouts collected on the robot itself) and offline, out-of-domain datasets (e.g., human video data). Note that there is a natural trade-off between these data sources – the robot data is most targetted to the test-time tasks, but is the most scarce due the difficulty of collecting it. As a result, this work will explicitly investigate both ends of this trade-off. Specifically, it investigates:

1. **Learning from Expert Robot Data:** In this setting π is learned exclusively from expert demonstrations. While this is a rich and highly targetted supervision signal, it is also the most expensive to collect – it requires human tele-operators to manually collect data in the real world for every task/scenario! This thesis demonstrates the power of this learning signal for training high capacity policy networks that can be used to scalably parameterize π.

2. **Learning from Autonomous Multi-Instituion Robot Data:** In this "middle ground" setting, π is learned using a mix of robot data collected from different institutions. To reduce human effort, this data is collected automatically using random robot interactions in a table-top bin setting. This allows us to learn significantly stronger policies using the increased amount of data, but is still limited to table-top settings.

3. **Learning from Internet Scale Data:** In this final setting, the policy is initialized (i.e., pre-trained) using Out-Of-Domain (OOD) data collected from more scalable sources (e.g., human video data and internet mining). The hope is to learn effective *representations* and physical/world knowledge from "cheaper" data. Then these representations are transferred to robotics domain to learn π, using a small amount of expert demonstration data.

1.0.1 Contributions

While investigating these data sources, this thesis makes the following contributions:

C1. Chapter 2 demonstrates how high-capacity transformer neural networks can learn complicated goal conditioned behaviors using a simple imitation learning algorithm. This showcases the power of expert demonstration data, but requires an unfeasible amount of demonstrations for real world tasks.

C2. Chapter 3 attempts to ease the data burden by developing a scalable strategy for cross institution robot data collection. This data is used to pre-train a robot policy that can then be quickly fine-tuned to solve new tasks in *novel hardware environments*. While much more scalable than the original expert demonstration assumptions, the level of diversity showcased in robot labs is still limited.

C3. Chapter 4 shows how representations trained on internet scale video data can be efficiently fine-tuned to solve robotic tasks in various domains. In other words, representations allow us to bring priors learned from entirely different domains into the robot setting.

C4. While representation learning is a powerful paradigm, it is unclear how to create a stronger representation for robotics. The core question is, "what out-of-domain data yields the strongest representations for downstream robotics tasks?" Chapter 5 analyzes this questions and finds that standard internet datasets transfer surprsingly well to robotics, but that self-supervised representation alone does not efficiently capture scene information required for robotics.

C5. Finally, Chapter 6 demonstrates how a semi-supervised approach can explicitly inject relevant object and environment level visual cues into the representation. This process yields a representation with stronger transfer to downstream manipulation tasks and can be applied to any pre-trained representation. Thus, we are able to more efficiently transfer data to the robotics domain from outside sources.

Part I

Learning from Expert Robot Data

Chapter 2

Transformers for One-Shot Visual Imitation

2.1 Motivation

Imitation is one of the most important cornerstones of intelligence. Watching other humans act, inferring their intentions, and attempting the same actions in our own home environments allows us to expand our skill set and enhance our representations of the world [229]. On the other hand, robots - while capable of imitating skills like table tennis [149] and driving [167] – are much less flexible when it comes to visual imitation. Most prior work in robotic imitation assumes that the agent is trying to acquire a single skill from demonstration(s) collected kinesthetically [161] (i.e. a human manually guides a robot) or via tele-operation [258]. These approaches can work so long as the target test-time task and environment are do not significantly differ from those seen during training. Is it possible to develop a robotic agent which can learn to imitate without these restrictions?

Visual imitation requires extracting a higher level goal from the visual demonstration and using the inferred goal to predict actions from pixels. But how does one represent goal/intention and how can this contextual information be incorporated into the policy function itself? There are three primary approaches in prior work: the first approach is to represent goals/intentions as pixels by generating goal images, and then inferring actions given current observations and inferred goals [200, 208]. While this approach is intuitive and interpretable, it is difficult to generate pixels, in a way that respects structural differences in the image. Figure 2.1 shows an example with well defined task semantics, but where a change in object positions makes it difficult to visually map the human state to the robot environment. The second approach has been to model visual imitation as a one-shot learning problem [59], which can be solved with meta-learning algorithms [68]. Here, a robot is given a single example, in the form of a video or demonstration (e.g. video + control telemetry), and must use that information to perform new instances of the same task. The demonstration is used to update the parameters of a policy function and the updated policy is executed on the robot. Domain gaps can be addressed with a learned adaptive los function [249]. While the one-shot formalism is very useful, estimating policy parameters from a single example can be an extremely difficult problem and prone to over-fitting.

In this chapter, we explore a third alternative: task-driven features for one-shot learning. We process both observations from the target agent and demonstrations frames from a ”teacher” agent in order to extract context-conditioned state representations. What neural network architectures can create task-driven features? While in the past, approaches such as LSTMs have been used, in this chapter, we focus on self-attention architectures. In particular, the Transformers architecture - while simple - has seen broad success in NLP [225] and Vision [233] tasks. Furthermore, using attention for control tasks has has basis in biology and psychology. Indeed, humans use attention mechanisms to create context driven representations [180], and directly supervising policies with human attention can dramatically increase task performance [257].

In this chapter, we propose using transformers [225] (or non-local self-attention modules [233]) to extract relational features which act as input state vectors for the policy function. Our transformers take as input both the spatial ResNet Features from teacher demonstration and the target agent. This allows the policy to automatically adapt its features to the task at hand, by using context frames to focus only on important task-specific details. For example, in Figure 2.1 the robot could use human context frames to focus only on relevant details like the red block’s location, and entirely ignore distracting elements like the table’s leg. However, transformer features could easily end up improperly weighting important details during test time. We propose to solve this issue by further supervising the state representation learning with an unsupervised inverse dynamics loss. This loss constrains the learning problem and ensures the final representations can model the underlying dynamics, as well as task specific details. Ultimately, our method achieves significant improvements over one-shot imitation learning baselines on a suite of pick and place tasks: our final policies demonstrate a 2x performance gain and can match baseline performance with 3x fewer data-points.

2.2 Related Work

Learning from Demonstration (LfD) is a rich and diverse field of study which focuses on enabling robots to learn skills from human or other expert demonstrations. A thorough review is out of scope for this chapter, so we gladly refer the reader to survey articles [8, 21, 187]. Of prior work, Behavior Cloning (BC) [179, 14], a common formulation of LfD, is most related to our project. BC involves imitating an expert agent given a set of trajectories (a.k.a time series of observations and actions), by fitting a function which approximates the expert’s action in a given state. This

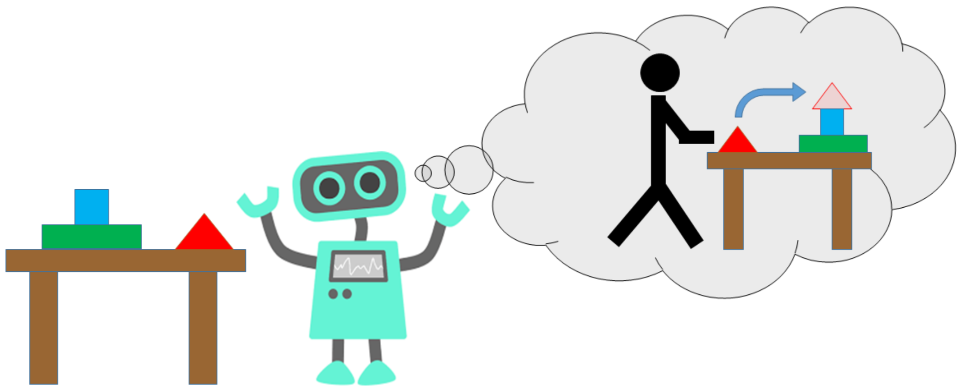


Figure 2.1: What should the robot do given video from another demonstration agent? A human would immediately know to place the red triangle on the blue square, and can use their past experience to execute the task. Is it possible to teach a robot to do the same?

simple formulae has proven successful in imitating a wide range of behaviors from visual inputs, including robotic manipulation tasks [173] and driving [22]. These methods have been extended to situations where expert observations are present without action labels [222], including prior work which linked this problem to inverse dynamics minimization [244]. However, both of these approaches require the demonstration agent match the imitator.

BC algorithms often assume that they are approximating a single state conditioned policy. In an environment with multiple tasks or multiple variations of the same task, this constraint can be limiting. Work on goal conditioned imitation learning seeks to relax these assumptions by allowing for policies which condition on a goal variable alongside the current state, and adjust their behavior accordingly. There are myriad ways to introduce goal conditioning, including with the robot's state [57], ”goal” images of the final state [143, 138, 70], natural language [139], and video or images of humans [198, 240]. In our project, we assume the robot has a single video of another agent (be it another robot or a human) doing a task, and must complete that same task itself using past experience. This is a specific instance of the one-shot learning problem [59], and has been investigated before previously using meta-learning with an adaptive loss [249]. Instead of using meta-learning, we propose to attack this problem with an attention mechanism over image frames.

A challenge in this line of work is learning visual representations which can enable the robot to deduce the task from video of another agent *and* perform the task itself. Work in computer vision demonstrated that deep neural networks are capable of learning such flexible representations for action recognition [205] and state estimation [130], but often require large image datasets to fully train. Unfortunately, collecting ImageNet [54] scale datasets on robotics platforms is prohibitively expensive, due to the cost of continuous robot operation and hardware fragility. Work in self-supervised learning [95, 35, 82] offers a glimmer of hope, by showing how large and (relatively) cheap sets of unlabelled images can be used to learn expressive and useful representations for other downstream tasks. These representations could be used directly as reward functions [194, 192], but it can be very difficult to define rewards for a suite of tasks. Instead, unsupervised learning techniques alongside simple data augmentation can be used to increase data efficiency when directly acquiring policies with reinforcement learning [210, 127, 120]. Even simpler self-supervised losses - like inverse modelling (i.e. predicting action between two sequential states) - can be used to learn robust policies which adapt to new environments [91]. Our goal in this project is to apply these insights in representation learning to the one-shot imitation learning problem.

2.3 Our Method

2.3.1 Problem Definition

Our method follows prior work [68, 249], and formalizes the one-shot imitation learning problem as supervised behavior cloning on a data-set of tasks. For each task $\mathcal{T}$ (e.g. place blue bottle in bin), we have several demonstration videos and target trajectories. Note that the demonstration videos and target trajectories are semantically similar tasks but could have different starting/end states. We represent each demonstration video as v_i and each target trajectory, t_i, as a temporal sequence of observations (o) and actions (a). Hence, $t_i = \{(o_i^{(1)}, a_i^{(1)}), \ldots, (o_i^{(k)}, a_i^{(k)})\}$.

Models are trained on a dataset of tasks $\mathcal{D} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$. During test time, new test tasks - $\mathcal{T}_{test}$ - are sampled which the model must successfully control the imitator agent to perform. Thus, all methods are evaluated on task success rates in held out environments. Our setup is challenging because: (a) morphological differences between demonstration and target agent (e.g. one is human and other is robot arm); (b) missing correspondence between demonstration videos and target trajectories.

2.3.2 Feature Learning with Transformers

Given video context from a demonstrator agent and image frames from the test environment, our representation module must deduce relevant features and efficiently pass them on to later stages of the pipeline for action prediction. For example, when given a video of a green bottle being dropped in a bin, the vision module should detect and represent the green bottle in its own environment while ignoring other distracting objects. We propose to learn this mechanism end-to-end using self-attention Transformer modules [225], in the hope that this powerful inductive bias helps the policy perform tasks successfully.

Before the attention module, individual images from both the context video and current state are passed through a ResNet-18 architecture [97], and spatial features (size $[512, T, H, W]$) are collected before the average pooling step. At this stage, the

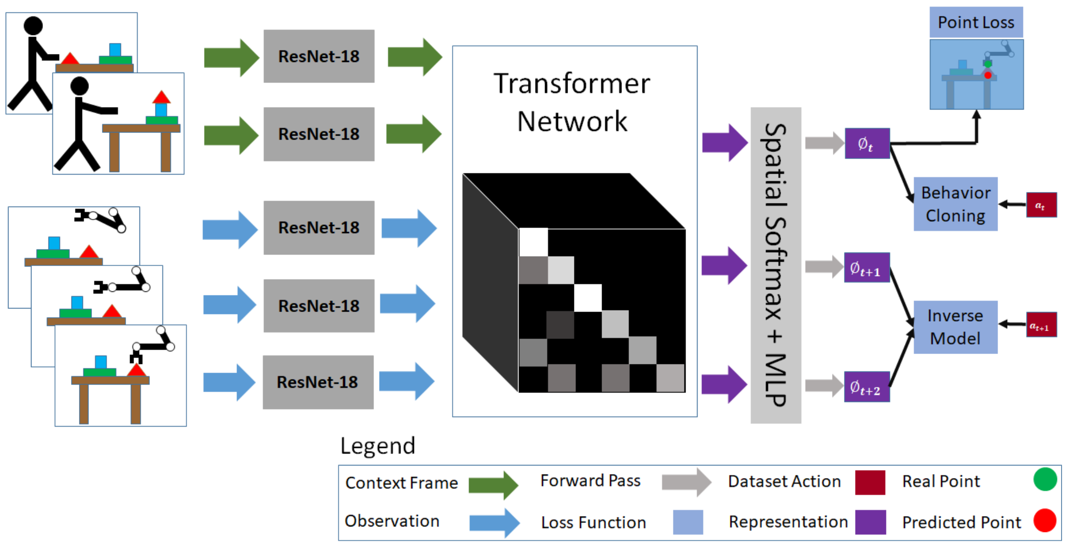


Figure 2.2: Our method uses a Transformer neural network to create task-specific representations, given context and observation features computed with ResNet-18 (w/ added positional encoding). The attention network is trained end-to-end with a behavior cloning loss, an inverse modelling loss, and an optional point loss supervising the robot's future pixel location in the image.

features are flattened (size $[512, T*H*W]$) and sinusoidal positional encodings [225] are added to the tensor (i.e. time and space treated as single dimension). These embeddings can allow neural networks to represent higher frequency functions [215], and we empirically found that they were crucial to preserving spatial and temporal information in the attention module. After adding positional encodings, the features are reshaped to their original size.

Next, the non-local multi-head attention operator is applied to the input tensor. We adopt a specific implementation of the Transformers self-attention module presented in Wang et al [233], which we augment with multi-headed self-attention [225]. First, the module generates **K**ey, **Q**uery, and **V**alue tensors by applying three separate 3D spatio-temporal convolutions (we use kernel size $k = 1$) with ReLU activation to the input tensor. To be clear, each convolution layer's input and output are $[d, T, H, W]$ tensors, where d is the Transformer's embedding size. These generated key, query, and value tensors are then flattened and projected down n separate times - once for each attention "head" - before attention is applied (final shape per head $[d, T*H*W]$). The self-attention operator is applied to each head individually. Considering attention head j, temperature parameter τ, and projected tensors K_j, Q_j, V_j, this amounts to:

$$A_j = \mathbf{softmax}(K_j^T Q_j/\tau) \qquad V_j^{(out)} = V_j A_j$$

The individual attention heads are then concatenated together channel-wise, and then projected back to the original 512 dimension size with another 3D convolution ($O = \textbf{Conv3D}(\textbf{concat}[V_1^{(out)}, \ldots, V_n^{(out)}])$). Note that this multi-head attention operator can be implemented with little overhead using batched matrix multiplication. Dropout [211], then a residual connection, and finally batch normalization [104] are applied to get the final output $f(x) = \textbf{batchnorm}(x + \textbf{dropout}(O))$, with final size $[512, T, H, W]$. In order to appropriately apply this to behavior cloning (where o_{t+1} is not known during test time), we make this operation causal by appropriately padding the 3D convolution operators and masking the attention.

2.3.3 Goal Conditioned Behavior Cloning

As discussed previously, our objective is to learn a policy $\pi(a_t|o_{1:t}, v)$ which ingests the current (or optionally all previous) state observations alongside a context video, and predicts a distribution over possible actions the expert policy would select. We process the input video stream with stacked attention modules to yield fixed size spatial features, with one feature map per time-step. The features are projected down to a fixed size representation vector using a spatial softmax operator [131], followed by a multi-layer perceptron with ReLU activations, and finally L2 normalization to unit length. This representation $\phi_t = F(o_{1:T}, v)$ is used for action prediction.

Multi-Modal Action Prediction: One of the most naive ways to predict $\pi(a_t|o_{1:t}, v)$ from ϕ_t is to simply parameterize the policy as a normal distribution $\pi(a_t|o_{1:t}, v) = \mathcal{N}(\mu(\phi_t), \sigma(\phi_t))$, and to sample actions from that. However, this approach can run into severe limitations when the real expert distribution is multi-modal. Consider a robot attempting to top-down lift a cup by its handle. Rotating the gripper by 90°or -90°, but not rotating at all (i.e. the mean action) would result in task failure since the gripper would close on top of the handle. Prior work [173, 171, 138] showed this limitation matters in practice, and rectifies the situation by predicting a mixture of uni-modal distributions. We adopt the same solution used by Lynch et al [138]. First, we discretize the action space (discussed in detail in Section 2.4.1) and then parameterize the policy as a discretized logistic mixture distribution [184]. For each timestep, we predict k logistic distributions with separate mean and scale, and form a mixture by convexly weighting them with vector α. The behavior cloning training loss is simply negative log-likelihood for this distribution:

$$\mathcal{L}_{BC}(\mathcal{D}, \theta) = -\ln(\Sigma_{i=0}^{k}\, \alpha_k(\phi_t)\; P(a_t, \mu_i(\phi_t), \sigma_i(\phi_t))$$

Where, $P(a_t, \mu_i(\phi_t), \sigma_i(\phi_t)) = F(\frac{a_t+0.5-\mu_i(\phi_t)}{\sigma_i(\phi_t)}) - F(\frac{a_t-0.5-\mu_i(\phi_t)}{\sigma_i(\phi_t)})$ and $F(\cdot)$ is the logistic CDF. During test time, actions are simply sampled from the distribution and executed on the robot without rounding. For most of our experiments, the model performed best when using two mixture components and learned constant variance parameters per action dimension.

2.3.4 Inverse Model Regularizer

Our method also adds a self-supervised inverse modeling objective to act as a regularizer to the behavior cloning loss during training. Context and trajectory snippets are sampled from the dataset, and images in them are randomized with sampled translations, color shifts, and crops. This randomization is applied consistently to frames from the context video, whereas images from the agent's observation stream (a.k.a trajectory images) are randomized *individually*. This randomized image stream is passed through the attention and representation modules to generate $\tilde{\phi}_t$. The representations $\tilde{\phi}_t$ and $\tilde{\phi}_{t+1}$ are used to predict a discretized logistic mixture distribution over intermediate actions. Thus, the inverse loss is:

$$\mathcal{L}_{INV}(\mathcal{D}, \theta) = -\ln(\Sigma_{i=0}^{k}\, \alpha_k(\tilde{\phi}_t, \tilde{\phi}_{t+1})\ \text{logistic}(\mu_i(\tilde{\phi}_t, \tilde{\phi}_{t+1}), \sigma_i(\tilde{\phi}_t, \tilde{\phi}_{t+1})))$$

We share parameters between the behavior cloning and inverse modeling objectives for the attention module, representation module, and distribution prediction heads (i.e. after first layer). In practice, we use the randomized image stream for both tasks as well, in order to minimize memory consumption.

2.3.5 Point Prediction Auxiliary Loss

Finally, our model uses ϕ_t to predict a 2D keypoint location corresponding to the location of the gripper in the image H timesteps in the future. Ground truth for this auxiliary loss is easy to acquire given either a calibrated camera matrix or object detector trained on the robot gripper. One could instead predict the 3D gripper position in world coordinates if neither is available. While not strictly needed for control, this loss is very valuable during debugging, since it lets us visually check during training if the model understand where the robot ought to be H timesteps in the future. The point prediction is parameterized with a simple multi-variate 2D normal distribution $\hat{p}_{t+H} \sim \mathcal{N}(\mu(\phi_t), \Sigma(\phi_t))$ with loss $\mathcal{L}_{pnt}(\mathcal{D}, \theta) = -\ln(\text{likelihood}(p_{t+H}, \hat{p}_{t+H}))$. Thus, the overall loss for our method is:

$$\mathcal{L}(\mathcal{D}, \theta) = \lambda_{BC}\, \mathcal{L}_{BC}(\mathcal{D}, \theta) + \lambda_{INV}\, \mathcal{L}_{INV}(\mathcal{D}, \theta) + \lambda_{pnt}\, \mathcal{L}_{pnt}(\mathcal{D}, \theta)$$

2.4 Experimental Results

Our model is evaluated on robotic manipulation tasks - namely pick and place tasks - in simulation using multi-agent MuJoCo [220] environments. Our evaluations investigate the following questions: (1) can our model perform new task instances (defined in 2.4.1) previously unseen during training? And (2) what components (e.g. inverse loss, etc.) are most crucial for successful control?

2.4.1 Simulation Environment and Tasks

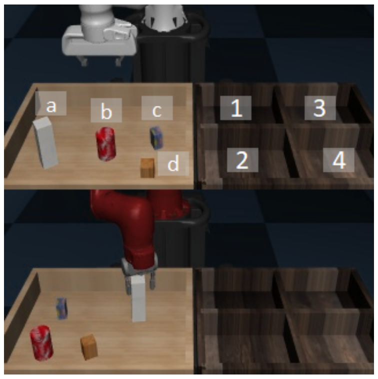


Figure 2.3: Our base environment is adopted from RoboTurk [145]. The 16 tasks consist of taking an object (a-b) to a bin (1-4). Top robot is agent and bottom is demonstrator.

Environment Description: The environments we use are modified variants of those originally presented in RoboTurk [145]. Visually, the *base environment* - shown in Figure 2.3 - is the exact same as the original from RoboTurk, except the object meshes are replaced with primitive geometric types (e.g. boxes and cylinders) in order to improve simulation contact stability and run-time. This modification results in only minor visual differences. In order to investigate visual imitation across agent morphology, we use duplicate versions of the environment with two visually distinct robots. The Sawyer robot (red robot in Figure 2.3) provides demonstration videos and the Panda robot (white robot in Figure 2.3) acts as the agent which our model must control. Both environment's action spaces are modified to support end-effector control. Given a target x, y, z position, rotation in axis-angle form, and gripper joint angle the environment solves for desired robot joint angles with inverse kinematics and sends joint velocities to the robot using a simple PD controller. Thus, the final action space consists of a target pose discretized into 256 independent bins per dimension in order to support our behavior cloning loss. It's important to note that the demonstrations we train on do not cover the whole state space, so the robot is mostly constrained to 3-DOF movement.

Task Definition: A "task instance" consists of picking an object from a specific start location - uniformly distributed on the table in Fig. 2.3 - and placing the object in one of the four bins on the right. Task instances are grouped into "tasks" based on shared properties. For example, picking a milk carton (from Fig. 2.3) and placing it into bin 1 is a task, and different task instances are constructed by changing the carton's start position. This precise definition allows us to collect a suite of train task instances, train models on that data, and test generalization to new task instances.

Data Collection Methodology: Training data is collected using an expert pick-place policy (built using privileged information from the simulator) in the target environment(s). For each task ($\mathcal{T}$) we repeatedly, sample a demonstration video (v_i) by executing the expert policy on the Sawyer robot, then shuffle the objects, and sample an expert trajectory (t_i) by executing the expert policy on the Panda robot. This way a dataset of tasks is formed from individual task instances.

2.4.2 Baseline Comparisons

Our investigation begins by evaluating our method's performance in 16 tasks in the base environment (Figure 2.3). We seek to determine the robot's physical compe-

Model	Reaching Success	Picking Success	Placing/Overall Success
Our Method	**99.4%** $\pm$ 1.2%	**92.5%** $\pm$ 4.1%	**88.8%** $\pm$ 5.0%
Contextual-LSTM	38.8% $\pm$ 7.6%	26.3% $\pm$ 6.9%	23.8% $\pm$ 6.7%
DAML [249]	36.9% $\pm$ 7.6%	10.6% $\pm$ 4.8%	6.9% $\pm$ 4.0%
DAML Auxiliary	47.8% $\pm$ 7.4%	17.8% $\pm$ 5.6%	13.3% $\pm$ 5.0%

Table 2.1: Comparison between our method and baselines in 16 pick and place tasks. Values indicate success rates and 95% confidence intervals for "stages"Footnote 1 in the overall pick and place task.

tency at manipulating all four objects, as well as its ability to deduce which task it should perform from context video. A natural way to quantify this is by breaking down the 16 pick and place tasks into "reach," "pick," and "place" stages[1], and reporting success rates on each stage individually. Failure modes can be successfully deduced from these rates. For example, since reaching is a physically easy task, if the robot does not reach the object then it is likely unable to deduce the target object from the context frames. Furthermore, if the robot reaches the object but is unable to pick it up, its physical dexterity (or lack thereof) is likely to blame.

We collect 100 train task instances using the methodology described previously for each of the 16 tasks. That amounts to 1600 total demonstration videos alongside 1600 expert robot trajectories. We train our method on the dataset and compare against the following baselines:

- **Contextual-LSTM:** This baseline utilizes a standard Encoder-Decoder LSTM [99, 212] (augmented with self-attention [11, 36]), to first consume the context video, and then predict actions from encoded observations. It uses the same mixture distribution our model uses. Before LSTM processing, images frames are embedded using a pre-trained ResNet-18 [97] neural net combined with spatial-softmax [131] and fully-connected layers. The whole network is trained end-to-end with a behavior cloning loss.

- **Domain Adaptive Meta-Learning:** DAML [249] uses a learned loss function to adapt a neural network's parameters to perform the desired task. We used a wider version of the network used in the original paper, since we found that using deeper models (like ResNet-18) resulted in overfitting on this task. To increase performance, the same discrete logistic action distribution is used. DAML is trained end-to-end with the MAML meta-learning algorithm [65] using a behavior cloning loss, along with explicit supervision of the pick and drop locations.

- **DAML-Auxiliary:** This method uses the same meta-learning model described

[1]Reaching is defined as placing the gripper within < 0.03 units from the target object, picking requires stably lifting the object > 0.05 units off ground, and placing requires putting the object in the correct bin (a.k.a task completion)

above, except only the predicted pick and place locations are used during test time. Given this prediction, a grasp motion is executed in the environment using a hard coded grasp policy.

For each of the 16 tasks, the models are prompted to perform new task instances (unseen during training) using freshly generated context videos. Success rates for our method and baselines (averaged across tasks) are shown in Table 2.1. As you can see, our method is the only one which can reliably perform new task instances. Its overall success rate is double the competing models' reaching success rate, including the DAML-auxiliary model which makes strong task assumptions, and the LSTM model which uses embedding level attention. The LSTM baseline's (which uses standard attention) relative failure supports our hypothesis that the Transformer architecture uniquely enables difficult visual processing. For additional experiments testing generalization to new objects (i.e. new tasks instead of new task instances) refer to Appendix H.1.1.

2.4.3 Architecture Ablation

While the our model clearly outperforms the other baselines, it is unclear if the Transformers architecture or additional losses deserve more credit. To test this thoroughly, the Transformers model is tested against an ablated version of itself without the attention mechanism (i.e. just temporal-spatial convolutions) using the same base environment comparison described before. Furthermore, models are trained with various versions of the baseline neural network architectures, alongside the additional loss terms. Specifically, 4 baseline architectures are considered: 2 of them adopt the small convolutional network used in prior work [249, 262] either with or without an additional LSTM [99] on top, and the other 2 use ResNet features [97] (again with or without LSTM). Note all architectures were tuned to maximize *their own* test performance rather than to match some other metric (e.g. number of parameters), since doing so often led to worse results for the baseline (e.g. larger LSTMs overfit more than Transformers). Results are presented in Figure 2.4. The key takeaways are encouraging. First, the Transformers architecture (w/ attention) outperforms a library of other architectures for this task by large margins, even using the same losses. Furthermore, the baselines perform better when trained with the additional losses compared to being trained purely with a behavior cloning loss as done before (contextual-LSTM's success rate improves 20% $\rightarrow$ 40%).

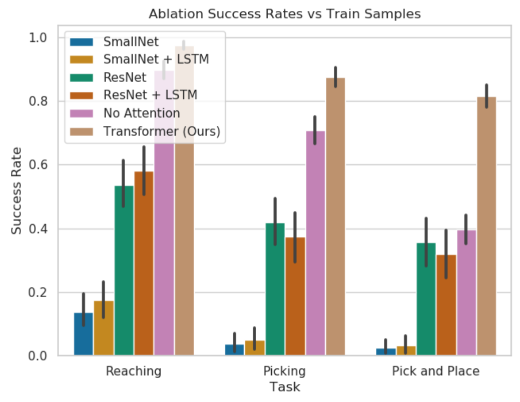


Figure 2.4: Our Transformer model is compared against other neural networks (all trained w/ our losses and code) to determine how useful the attention mechanism really is. The Transformer architecture outperforms all others, including a version of itself w/out attention.

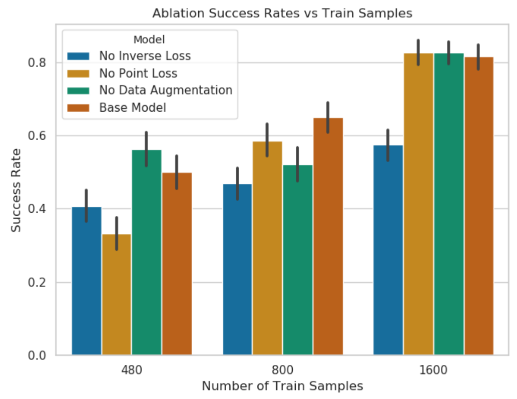


Figure 2.5: We compute success rate v.s number of train samples for our method and versions with one loss excluded (all w/ Transformer). Note the model without inverse loss is usually outperformed when compared to its peers trained on the same data.

2.4.4 Loss Function and Method Ablations

Given that our training losses/code boosted baseline architecture performance compared to using just behavior cloning, we now seek to test exactly which component was most useful. It's entirely possible that some of the additional parts offer more utility in the "low-data" regime where over-fitting is more likely, and thus are less useful when more data is present. Thus, we collect two more versions of the base environment dataset with fewer samples (480 and 800 samples pairs), and train three ablations - one model without the inverse loss, one without the point loss, and one without data augmentation - alongside our base model on all three datasets (two new sets + original). That results in a total of 12 models, all of which we evaluate in the same manner as before. Overall success rates for all models are in Figure 2.5. Note that the model without the inverse loss is outperformed by its counterparts in two out of three datasets, whereas the point loss only makes a significant difference in the smallest dataset. Indeed as the number of datapoints increases, so does the importance of the inverse loss: the model without inverse loss is more than 25% worse than its counterparts in the $N = 1600$ case! While the inverse loss clearly makes a difference, this cannot be observed as "positive transfer" in the behavior cloning train/test loss (see Appendix H.1.2). This suggests inverse loss regularization helps test time performance in ways not captured in the training objective. Finally, conditioning our policy on context video proved to be more effective than just feeding it the last frame, which indicates the demonstration helps our model determine which task to perform compared to using a "goal image" frame. For more check Appendix H.1.3.

2.5 Discussion

In this project we explore the one-shot visual imitation learning problem. Our experiments highlight two technical contributions - applying the Transformers architecture to one-shot imitation tasks and a self-supervised inverse modelling objective - which both result in large performance gains over baseline one-shot imitation learning approaches. More specifically, our ablations show that our model trained without the self-supervised inverse loss performs significantly worse when compared to other versions with the inverse loss, and all of our Tansformers models (even without inverse loss) outperform a Seq2Seq LSTM trained with traditional "embedding level" attention mechanisms by roughly 2x.

The main takeaway here is that injecting the right biases - both in terms of network design and the loss function - can help policies perform better during test-time. We believe that the Transformer's attention mechanism provides such a bias by allowing for task conditioned representations, whereas the inverse model forces the policy to preserve information which is needed for robust control during test time. We hope that these findings prove useful to others working on one-shot imitation learning and goal conditioned reinforcement learning in general.

Part II

Learning From Autonomous Multi-Institution Robot Data

Chapter 3

RoboNet: Large-Scale Multi-Robot Learning

3.1 Motivation

The key motivation for using machine learning in robotics is to build systems that can handle the diversity of open-world environments, which demand the ability to generalize to new settings and tasks. Such generalization may either be *zero-shot*, without any additional data from the target domain, or very fast, using only a modest amount of target domain data. Despite this promise, two of the most commonly raised criticisms of machine learning applied to robotics are the amount of data required per environment due to limited data-sharing, and the resulting algorithm's poor generalization to even modest environmental changes. A number of works have tried to address this by developing simulations from which large amounts of diverse data can be collected [182, 6], or by attempting to make robot learning algorithms more data efficient [50, 52]. However, developing simulators entails a deeply manual process, which so far has not scaled to the breadth and complexity of open-world environments. The alternative of using less real-world data often also implies using simpler models, which are insufficient for capturing the many details present in complex real-world environments such as object geometry or appearance.

Instead, we propose the opposite – using dramatically larger and more varied datasets collected in the real world. Inspired by the breadth of the ImageNet dataset [54], we introduce *RoboNet*, a dataset containing roughly 162,000 trajectories with video and action sequences recorded from 7 robots, interacting with hundreds of objects, with varied viewpoints and environments, corresponding to nearly 15 million frames. The dataset is collected autonomously with minimal human intervention, in a self-supervised manner, and is designed to be easily extensible to new robotic hardware, various sensors, and different collection policies.

The common practice of re-collecting data from scratch for every new environment essentially means re-learning basic knowledge about the world — an unneces-

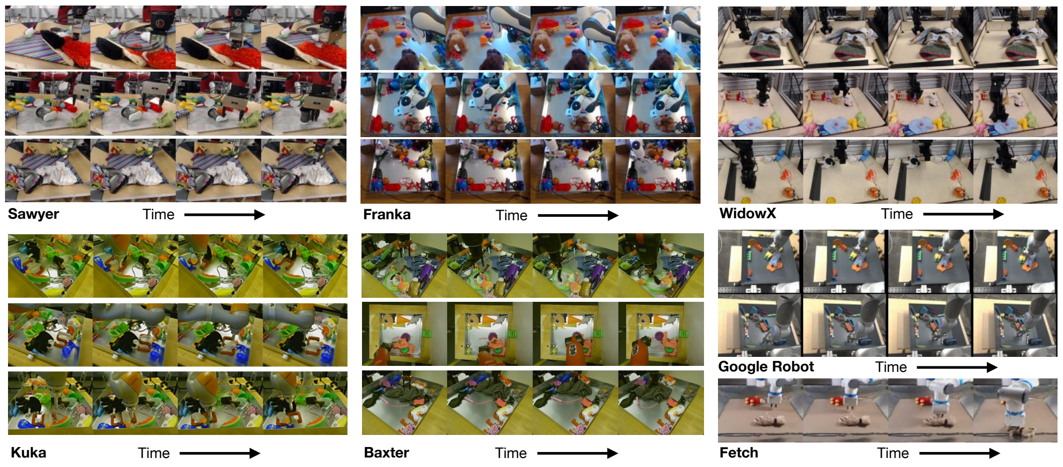


Figure 3.1: A glimpse of the RoboNet dataset, with example trajectories, robots, and viewpoints. We collected data with Sawyer, Franka, WidowX, Kuka, and Baxter robots, and augmented the dataset with publicly-available data from a robot from Google [66], a Fetch [251], and a Sawyer [61]. We use RoboNet to study the viability of large-scale data-driven robot learning, as a means to attain broad generalization across robots and scenes.

sary effort. In this chapter, we show that sharing data across robots and environments makes it possible to pre-train models on a large dataset of experience, thus extracting priors that allow for fast learning with new robots and in new scenes. If the models trained on this data can acquire the underlying shared patterns in the world, the resulting system would be capable of manipulating *any* object in the dataset using *any* robot in the dataset, and potentially even transfer to new robots and objects.

To learn from autonomously-collected data without explicit reward or label supervision, we require a self-supervised algorithm. To this end, we study two methods for sharing data across robot platforms and environments. First, we study the visual foresight algorithm [67, 61], a deep model-based reinforcement learning method that is able to learn a breadth of vision-based robotic manipulation skills from random interaction. Visual foresight uses an action-conditioned video prediction model trained on the collected data to plan actions that achieve user-specified goals. Second, we study deep inverse models that are trained to predict the action taken to reach one image from another image, and can be used for goal-image reaching tasks [3, 138]. However, when trained in a single environment, robot learning algorithms, including visual foresight and inverse models, do not generalize to large domain variations, such as different robot arms, grippers, viewpoints, and backgrounds, precluding the ability to share data across multiple experimental set-ups and making it difficult to share data across institutions.

Our main contributions therefore consist of the RoboNet dataset, and an experimental evaluation that studies our framework for multi-robot, multi-domain model-based reinforcement learning based on extensions of the visual foresight algorithm and prior inverse model approaches. We show that, when trained on RoboNet, we can acquire models that generalize in zero shot to novel objects, novel viewpoints, and novel table surfaces. We also show that, when these models are finetuned with small amounts of data (around 400 trajectories), they can generalize to unseen grippers and new robot platforms, and perform better than robot-specific and environment-specific training. We believe that this chapter takes an important step towards large-scale data-driven approaches to robotics, where data can be shared across institutions for greater levels of generalization and performance.

3.2 Related Work

Deep neural network models have been used widely in a range of robotics applications [75, 254, 30, 255, 16, 103]. However, most work in this area focuses on learning with a single robot in a single domain, while our focus is on curating a dataset that can enable a single model to generalize to multiple robots and domains.

The multi-task literature [51, 5], lifelong learning literature [216, 217], and meta-learning literature [65, 4] describe ideas that are tightly coupled with this concept. By collecting task-agnostic knowledge in wide variety of domains, a robotic system should be able to rapidly adapt to new, unseen environments using relatively little target domain data.

Large-scale, self-supervised robot learning approaches have adopted a similar viewpoint [165, 133, 83, 67, 255, 3, 163, 61]. Unlike these methods, we specifically consider transfer across multiple robots and environments, as a means to enable researchers to share data across institutions. We demonstrate the utility of our data by building on the visual foresight approach [67, 61], as it further enables generalization across tasks without requiring reward signals. This method is related to a range of recently proposed techniques that use predictive models for vision-based control [26, 218, 234, 125, 153]. Further, we also study how we can extend vision-based inverse models [3, 163, 251, 138] for generalizable robot-agnostic control.

A number of works have studied learning representations and policies that transfer across domains, including transfer from simulation to the real world [182, 219, 108], transfer across different dynamics [38, 252, 164, 6], transfer across robot morphologies with invariant feature spaces [85] and modularity [55], transfer across viewpoints through recurrent control [183], and transfer across objects [68, 107], tasks [59] or environments [39] through meta-learning. In contrast to these works, we consider transfer at a larger scale across not just one factor of variation, but across objects, viewpoints, tasks, robots, and environments, without the need to manually engineer simulated environments.

Outside of robotics, large and diverse datasets have played a pivotal role in machine learning. One of the best known datasets in modern computer vision is the

ImageNet dataset [54], which popularized an idea presented earlier in the tiny image dataset [224]. In particular, similar to our work, the main innovation in these datasets was not in the quality of the labels or images, but in their diversity: while prior datasets for image classification typically provided images from tens or hundreds of classes, the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) contained one thousand classes. Our work is inspired by this idea: while prior robotic manipulation methods and datasets [66, 247, 29, 83, 145, 61, 199] generally consider a single robot at a time, our dataset includes 7 different robots and data from 4 different institutions, with dozens of backgrounds and hundreds of viewpoints.

This makes it feasible to study broad generalization in robotics in a meaningful way.

3.3 Data-Driven Robotic Manipulation

In this work we take a *data-driven* approach to robotic manipulation. We do not assume knowledge of the robot's kinematics, the geometry of objects or their physical properties, or any other specific property of the environment. Instead, basic common sense knowledge, including rigid-body physics and the robot's kinematics, must be implicitly learned purely from data.

Problem statement: learning image-based manipulation skills. We use data-driven robotic learning for the task of object relocation – moving objects to a specified location either via pushing or grasping and placing. However, in principle, our approach is applicable to other domains as well. Being able to perform tasks based on camera images alone provides a high degree of generality. We learn these skills using a dataset of trajectories of images $I_{0:T}$ paired with actions $a_{0:T}$, here T denotes the length of the trajectory. The actions are sampled randomly and need to provide sufficient exploration of the state space, which has been explored in prior work [61, 241]. This learning and data collection process is self-supervised, requiring the human operator only to program the initial action distribution for data collection and to provide new objects at periodic intervals. Data collection is otherwise unattended.

Preliminaries: robotic manipulation via prediction. We build on visual foresight [67, 61], a method based on an action-conditioned video prediction model that is trained to predict future images, up to a horizon h, from on past images: $\hat{I}_{t+1:t+h} = f(I_t, a_{t:t+h-1})$, using unlabeled trajectory data such as the data presented in the next section. The video prediction architecture used in visual foresight is a deterministic variant of the SAVP video prediction model [128] based heavily on prior work [66]. This model both predicts future images and the motion of pixels, which makes it straightforward to set goals for relocating objects in the scene simply by designating points $d_{0,i}$ (e.g., pixels on objects of interest), and for each one specifying a goal position $d_{g,i}$ to which those points should be moved. We refer to $d_{0,i}$ as designated pixels. These goals can be set by a user, or a higher-level planning algorithm. The robot can select actions by optimizing over the action sequence to find one that results in the desired pixel motion, then executing the first action in this sequence, observing a new image, and replanning. This effectively implements image-based model-predictive control (MPC). With an appropriate choice of action representation, this procedure can automatically choose how to best relocate objects, whether by pushing, grasping, or even using other objects to push the object of interest. Full details can be found in Appendix H.2.1 and in prior work [61].

Preliminaries: robotic manipulation via inverse models. To evaluate RoboNet's usefulness for robot learning beyond use with the visual foresight algorithm, we evaluate a simplified version of the inverse model in [138]. Given context data, $\{\dots, (I_{t-2}, a_{t-2}), (I_{t-1}, a_{t-1})\}$, the current image observation I_t, and a goal image I_{t+T}, the inverse model is trained to predict actions $a_t, \dots, a_{t+T-1}$ (where T is a given horizon) that are needed to take the robot from the start to the goal image. Our experiments train a one-step inverse model where $T = 1$, which can be trained with supervised regression. At test time, the model takes as input 2 context frame/action pairs, the current image, and a goal image and then will predict an action which ought to bring the robot to the goal. This process is can be repeated at the next time-step, thus allowing us to run closed loop visual control for multiple steps.

3.4 The RoboNet Dataset

To enable robots to learn from a wide range of diverse environments and generalize to new settings, we propose RoboNet, an open dataset for sharing robot experience. An initial set of data has been collected across 7 different robots from 4 different institutions, each introducing a wide range of conditions, such as different viewpoints, objects, tables, and lighting. By having only loose specifications[1] on how the scene can be arranged and which objects can be used, we naturally obtain a *large amount of diversity*, an important feature of this dataset. By framing the data collection as a cross-institutional effort, we aim to make the diversity of the dataset grow over time. *Any research lab is invited to contribute to it.*

3.4.1 Data Collection Process

All trajectories in RoboNet share a similar action space, which consists of deltas in position and rotation to the robot end-effector, with one additional dimension of the action vector reserved for the gripper joint. The frame of reference is the root link of the robot, which need not coincide with the camera pose. This avoids the need to calibrate the camera, but requires any model to infer the relative positioning between the camera and the robots' reference frames from a history of context frames. As we show in Section 3.5, current models can do this effectively. The action space can

[1]Specifications can be found here: `http://www.robonet.wiki`

also be a subset of the listed dimensions. We chose an action parametrization in end-effector space rather than joint-space, as it extends naturally to robot arms with different degrees of freedom. Having a unified action space throughout the dataset makes it easier to train a single model on the entire dataset. However, even with a consistent action space, variation in objects, viewpoints, and robot platforms has a substantial effect on how the action influences the next image.

In our initial version of RoboNet, trajectories are collected by applying actions drawn at random from simple hand-engineered distributions. We most commonly use a diagonal Gaussian combined the automatic grasping primitive developed in [60]. More details on the data collection process are provided in Appendix H.2.3.

3.4.2 The Diverse Composition of RoboNet

The environments in the RoboNet dataset vary both in robot hardware, i.e. robot arms and grippers, as well as environment, i.e arena, camera-configuration and lab setting, which manifests as different backgrounds and lighting conditions (see Figure 3.1 and 3.2). In theory, one could add any type (depth, tactile, audio, etc.) of sensor data to RoboNet, but we stick to consumer RGB video cameras for the purposes of this project. There is no constraint on the type of camera used, and in practice different labs used cameras with different exposure settings. Thus, the color temperature and brightness of the scene varies through the dataset. Object sets also vary substantially between different lab settings. To increase the number of tables, we use inserts with different textures and colors. To increase the number of gripper configurations, we 3D printed different finger attachments. We collected 104.4k trajectories for RoboNet on a Sawyer arm, Baxter robot, low-cost WidowX arm, Kuka LBR iiwa arm, and Franka Panda arm. We additionally augment the dataset with publicly available data from prior works, including 5k trajectories from a Fetch robot [251] and 56k trajectories from a robot at Google [66]. The full dataset composition is summarized in Table 3.1.

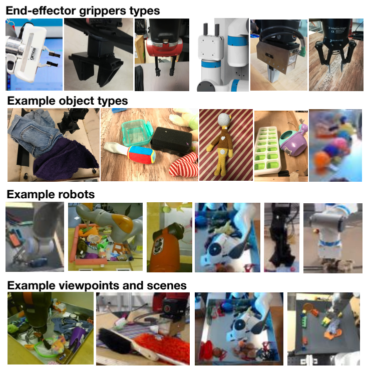
3.4.3 Using and Contributing to RoboNet

The RoboNet dataset allows users to easily filter for certain attributes. For example, it requires little effort to setup an experiment for training on all robots with a certain type of gripper, or all data from a Sawyer robot. An overview of the current set of attributes is shown in Table 3.1, and image examples are provided in Figure 3.2. We provide code infrastructure and common usage examples on the project website.[2]

Scripts for controlling common types of robots, for collecting data, and for storing data in a standard format are available on the project website. On the same webpage we are also providing a platform that allows anyone to upload trajectories. After data has been uploaded we will perform manual quality tests to ensure that the

[2]The project webpage is at http://www.robonet.wiki/

trajectories comply with the standards used in RoboNet: the robot setup should occupy enough space in the image, the action space should be correct, and the images should be of the right size. After passing the quality test, trajectories are added to the dataset. An automated quality checking procedure is planned for future work.



Robot type (number of trajectories)	Sawyer (68k), Baxter (18k), WidowX (5k), Franka (7.9k), Kuka (1.8k), Fetch (5k) [251], GoogleRobot (56k) [66]
Gripper type	Weiss Robotics WSG-50, Robotiq, WidowX, Baxter, Franka, Kuka
Arena types	7
Arena inserts	10
Gripper configurations	10
Camera configuration	113
Lab environments	4

Figure 3.2: Qualitative examples of the various attributes in the RoboNet dataset.

Table 3.1: Quantitative overview of the various attributes in the RoboNet dataset, including the 7 different robot arms and 7 different grippers.

3.5 Robot-Agnostic Visual Control: Model Training and Experiments

A core goal of this chapter is to study the viability of large-scale data-driven robot learning as a means to acquire broad generalization, across scenes, objects, and even robotic platforms. To this end, we design a series of experiments to study the following questions: (1) can we leverage RoboNet to enable zero-shot generalization or few-shot adaptation to novel viewpoints and novel robotic platforms? (2) how does the breadth and quantity of data affect generalization? (3) do predictive models trained on RoboNet memorize individual contexts or learn generalizable concepts that are shared across contexts? Finally, we evaluate a simple inverse model to test if RoboNet can be used with learning algorithms other than visual foresight.

3.5.1 Visual Foresight: Experimental Methodology

For our visual foresight robot experiments, we evaluate models in terms of performance on the object relocation tasks described in Section 3.3. A task is defined as moving an object not in the training set to a particular location in the image. After running the learned policy or planner, we measure the distance between the achieved object position and the goal position. We judge a task to be successful if the operator judges the object is mostly covering the goal location at the end of the rollout. Models within an experiment are compared on the same set of object relocation tasks. We use this evaluation protocol through the rest of the experiments. Please refer to Appendix H.2.5 for some images of the testing environments. Note that results should not be compared across different experiments, since task difficulty varies across robots and human operators.

3.5.2 Visual Foresight: Zero-Shot Generalization to New Viewpoints and Backgrounds

In this section, we study how well models trained on RoboNet can generalize, without any additional data, to novel viewpoints and held-out backgrounds with a previously seen robot. Generalizing to a new viewpoint requires the model to implicitly estimate the relative positioning and orientation between the camera and the robot, since the actions are provided in the robot's frame of reference. We attempt five different object relocation tasks from two views in order to compare a model that has been trained on 90 different viewpoints against a model that was only trained on single viewpoint. The arrangement of the cameras is shown in Appendix H.2.5. In Table 3.2, we show object relocation accuracy results for both of these models when testing on both the seen viewpoint (left) and a novel viewpoint (right). The results show that the model trained on varied viewpoints achieves lower final distance to the goal on the benchmark tasks for *both* views, thus illustrating the value of training on diverse datasets.

We tested the same multi-view model on a similar set of tasks in an environment substantially different from the training environment. In Figure 3.3 we show a successful execution of a pushing task in this new environment. The multi-view model achieves an average final distance of 14.4 ± 2 cm (std. error) in the new setting. This performance is comparable to that achieved by the multi-view model in a novel viewpoint, which suggests the model is also able to effectively generalize to novel surroundings.

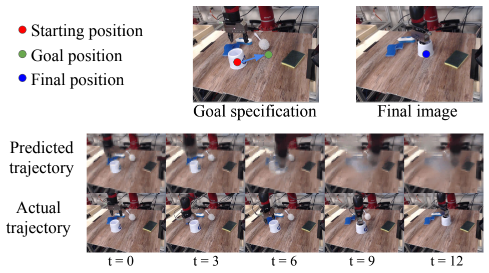


Figure 3.3: Zero-shot generalization to new backgrounds with a model trained across multiple views.

	Avg. dist. (cm) seen view	Avg. dist. (cm) held-out view
single view	14.8 ± 3.8	23.2 ± 2.6
multi-view	**9 ± 2.2**	**16.2± 2.9**

Table 3.2: Evaluation of viewpoint generalization, showing the average distance to the goal after executing the action sequence and standard error. A model trained on multiple views can better generalize to a new viewpoint.

3.5.3 Visual Foresight: Few-Shot Adaptation to New Robots

When evaluating on domains that differ more substantially from any domain present in the dataset, such as settings that contain an entirely new robotic arm, zero-shot generalization is not possible. In this section, we evaluate how well visual foresight can adapt to entirely new robots that were not shown to the model during training. This is one of the most challenging forms of generalization, since robots have not only different appearances, but also different dynamics when interacting with objects, different kinematics, and different work-space boundaries.

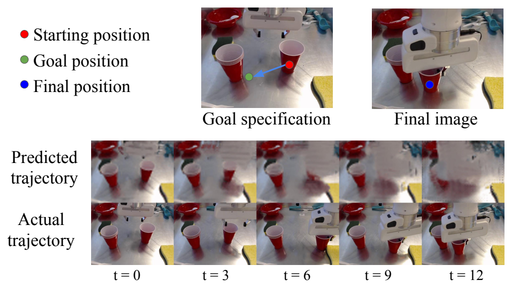


Figure 3.4: Example task of grasping and moving a thin plastic cup with the Franka robot, using visual foresight pre-trained on RoboNet w/o Franka and fine-tuned on 400 trajectories from the Franka robot.

To test our hypothesis, we collect a small number (300-400) of random trajectories from the target robot environment. Models are then pre-trained on the entirety of RoboNet, but holding out the data from the target robot. These models are then fine-tuned using the aforementioned collected trajectories. We compare to a separate model that is trained from scratch on those trajectories. Additionally, for the Franka experiments another model is trained on all the Franka data in RoboNet, and for the Baxter experiment one model is pre-trained on just Sawyer data in RoboNet and fine-tuned to Baxter. The R3 and Fetch were also not included in the pre-training data due to computational constraints.

Kuka Experiments	Success rate
Random Train on N=400	10%
Random Train on N=1800	30%
RoboNet w/o Kuka Finetune on N=400	**40%**

Table 3.3: Results for adaptation to an unseen Kuka robot. The model pre-trained on RoboNet without the Kuka, R3, and Fetch data, achieves the best performance when fine-tuned with 400 trajectories from the test robot.

Franka Experiments	Success rate
Random Train on N=400	20%
Random Train on N=8000	35%
RoboNet w/o Franka Finetune on N=400	**40%**

Table 3.4: Results for adaptation to an unseen Franka robot. The model pre-trained on RoboNet without the Franka, R3, and Fetch data, achieves the best performance when fine-tuned with 400 trajectories from the test robot.

Baxter Experiments	Success rate
Random Train on N=300	33%
Sawyer Finetune on N=300	**83%**
RoboNet w/o Baxter Finetune on N=300	58%

Table 3.5: Evaluation results for adaptation to an unseen Baxter robot. The model pre-trained on RoboNet's Sawyer data, achieves the best performance when fine-tuned with 300 trajectories from the test robot.

The quantitative results are summarized in Table 3.3, Table 3.4, and Table 3.5. The results show that RoboNet pre-training provides substantial improvements over training from scratch, on all three test robots. In the Kuka and Franka experiments, a model fine-tuned on *just 400 samples* is able to outperform its counterpart trained on all of RoboNet's data from the respective robot. These results suggest that RoboNet pre-training can offer large advantages over training tabula rasa, by substantially reducing the number of samples needed in a new environment. Figure 3.4 shows a successful rollout of visual foresight on a challenging task of positioning a plastic cup to a desired location.

In the Baxter experiment, we also find that pre-training on specific subsets of RoboNet (in this case the Sawyer, which is visually more similar to the Baxter than other robots) can perform significantly better than training on the entire dataset. Hence, this experiment (as well as the Robotiq gripper generalization experiment in Appendix H.2.6) demonstrates that increased diversity during pre-training can sometimes hurt performance when compared to pre-training on a subset of RoboNet. We hypothesize that more specific pre-training works better, because our models under-fit when trained on all of RoboNet, which we study in more detail in the next section.

3.5.4 Visual Foresight: Model Capacity Experiments

When training video prediction models on RoboNet, we observe clear signs of underfitting. Training error and validation error are generally similar, and both plateau before reaching very high performance on the training sequences. During test time, inaccurate predictions are often the cause of poor performance on the robot. Thus, we perform an additional experiment to further validate the underfitting hypothesis. We train two large models, using a simplified deterministic version of the network architecture presented in [226], on RoboNet's Sawyer data: one model has 200M parameters and the other has 500M parameters. The 200M parameter model has 0.104 ± 0.057 average ℓ_1 per-pixel error on a held out test set, whereas the 500M model has 0.0847 ± 0.045 ℓ_1 per-pixel error. These results suggest that current visual foresight models – even ones much larger than the 5M - 75M parameter models used in our control experiments – suffer from underfitting, and future research on higher capacity models will likely improve performance.

3.5.5 Inverse Model: Multi-Robot and Multi-Viewpoint Reaching

To evaluate RoboNet's applicability to different control algorithms, we train a simple version of the inverse model from [138] (refer to Section 3.3 for details) on a subset of RoboNet containing only Sawyer and Franka data. The *same* model is evaluated on both robots: the Sawyer experiments also contain a held-out view. We evaluate model performance on simple reaching tasks. Tasks are constructed by supplying a goal image, by taking an image of the gripper in a different reachable state.

After task specification, the model runs continuously, re-planning each step until a maximum number of steps is reached. Success is determined by a human judge. This model is able to perform visual reaching tasks on both robots, including from a novel viewpoint not seen during training. However, because of its comparatively greedy action selection procedure, we observe that it tends to perform poorly on more complex tasks that require object manipulation.

3.6 Discussion

We presented RoboNet, a large-scale and extensible database of robotic interaction experience that combines data from 7 different robots, multiple environments and backgrounds, over a hundred camera viewpoints, and four separate geographic locations. We demonstrated two example use-cases of the dataset by (1) applying the visual foresight algorithm [61] and (2) learning vision-based inverse models. We evaluated generalization across many different experimental conditions, including varying viewpoints, grippers, and robots. Our experiments suggested that finetuning models pretrained on RoboNet offers a powerful way to quickly allow robot learning algorithms to acquire vision-based skills on unseen robot hardware.

Inverse Model	Success
Sawyer Reaching Front View	4/5
Sawyer Reaching Unseen View	5/5
Franka Reaching Front View	4/5

Table 3.6: Inverse model results on 5 reaching tasks. The model is successful across multiple robot platforms and generalizes to a new viewpoint.

Our experiments further found that video prediction models with $\leq$ 75M parameters tend to heavily *underfit* on RoboNet. While much better, we even observe underfitting on 500M-parameter models. As a result, prediction models struggle to take advantage of the breadth and diversity of data from multiple robots, domains, and scenes, and instead seem to perform best when using a subset of RoboNet that looks most similar to the test domain. This suggests two divergent avenues for future work. On one hand, we can develop algorithms that automatically select subsets of the dataset based on various attributes in a way that maximizes performance on the test domain. In the short term, this could provide considerable improvements with our current models. However, an alternative view is to instead research how to build more flexible models and policies, that are capable of learning from and larger and more diverse datasets across many robots and environments. We hope that the RoboNet dataset can serve as a catalyst for such research, enabling robotics researchers to study such problems in large-scale learning. Next, we discuss limitations of the dataset and evaluation, and additional directions for future work.

Limitations. While our results demonstrated a large degree of generalization, a number of important limitations remain, which we aim to study in future work. First and foremost, the tasks we consider are relatively simple manipulation tasks such as pushing and pick-and-place, with relatively low fidelity. This is an important limitation that hinders the ability of these models to be immediately of practical use. However, there are a number of promising recent works that have demonstrated how predictive models of observations can be used for solving tasks of greater complexity such as tool use [241] and rope manipulation [125], and tasks at greater fidelity such as block mating [153] and die rolling [218]. Further, one bottleneck that likely prevents better performance is the quality of the video predictions. We expect larger, state-of-the-art models [236, 226] to produce significantly better predictions, which would hopefully translate to better control performance.

Another limitation of our current approach and dataset is the source of data being from a pre-determined random policy. This makes data collection scalable, but at the cost of limiting more complex and nuanced interactions. In future work, we plan to collect and solicit data from more sophisticated policies. This includes demonstration data, data from modern exploration methods that scale to pixel observations [18, 24, 162], and task-driven data from running reinforcement learning on particular tasks. As shown in prior work [241], improving the forms of interactions in the dataset can significantly improve performance.

In selecting how and where to collect additional data, our experiments suggest that adaptation to new domains is possible with only modest amounts of data, on the order of a few hundred trajectories. This suggests that prioritizing variety, i.e. small amounts of data from many different domains, is more important than quantity in future collection efforts.

Future Directions. This chapter takes the first step towards creating learned robotic agents that can operate in a wide range of environments and across different hardware. While in this chapter, we explored two particular classes of approaches, we hope that RoboNet will inspire the broader robotics and reinforcement learning communities to investigate how to scale model-based *or* model-free RL algorithms to meet the complexity of the real world, and to contribute the data generated from their experiments back into a shared community pool. In the long term, we believe this process will iteratively strengthen the dataset, and thus allow the algorithms derived from it to achieve greater levels of generalization across tasks, environments, robots, and experimental set-ups.

Part III

Learning from Internet Scale Data

Chapter 4

Manipulate by Seeing: Creating Manipulation Controllers from Pre-Trained Representations

4.1 Motivation

The lack of suitable, large-scale data-sets is a major bottleneck in robot learning. Due to the physical nature of data collection, robotics data-sets are: (a) hard to scale; (b) collected in sterile, non-realistic environments (e.g. robotics lab); (c) too homogeneous (e.g. toy objects with fixed backgrounds/lighting). In contrast, vision data-sets contain diverse tasks, objects, and settings (e.g. Ego4D [81]). Therefore, recent approaches have explored transferring priors from large scale vision data-sets to robotics settings. What is the right way to accomplish this?

Prior work uses vision data-sets to pre-train representations [160, 154, 239] that encode image observations as state vectors (i.e. $s = R(i)$). This visual representation is then simply used as an input for a controller learned from robot data – e.g. a policy $\pi(a|s)$, trained with expert data via Behavior Cloning [179], or a Value function $V(s)$, trained using exploratory roll-outs via Reinforcement Learning [213]. Is this approach the most efficient way to use pre-trained representations? We argue that pre-trained networks can do more than just represent states, since their latent space already encodes semantic, task-level information – e.g. by placing semantically similar states more closely together. Leveraging this structure to infer actions, could enable us to use significantly less robotic data during train time.

This chapter achieves this by fine-tuning a pre-trained representation into: (a) a one-step dynamics module, $F(s, a)$, that predicts how the robot's next state given the current state/action; and (b) a "functional distance module", $d(s, g)$, that calculates how close the robot is to achieving its goal g in the state s. The distance function is learned with limited human demonstration data, using a contrastive learning objective (see Fig. 4.1). Both d and F are used in conjunction to greedily

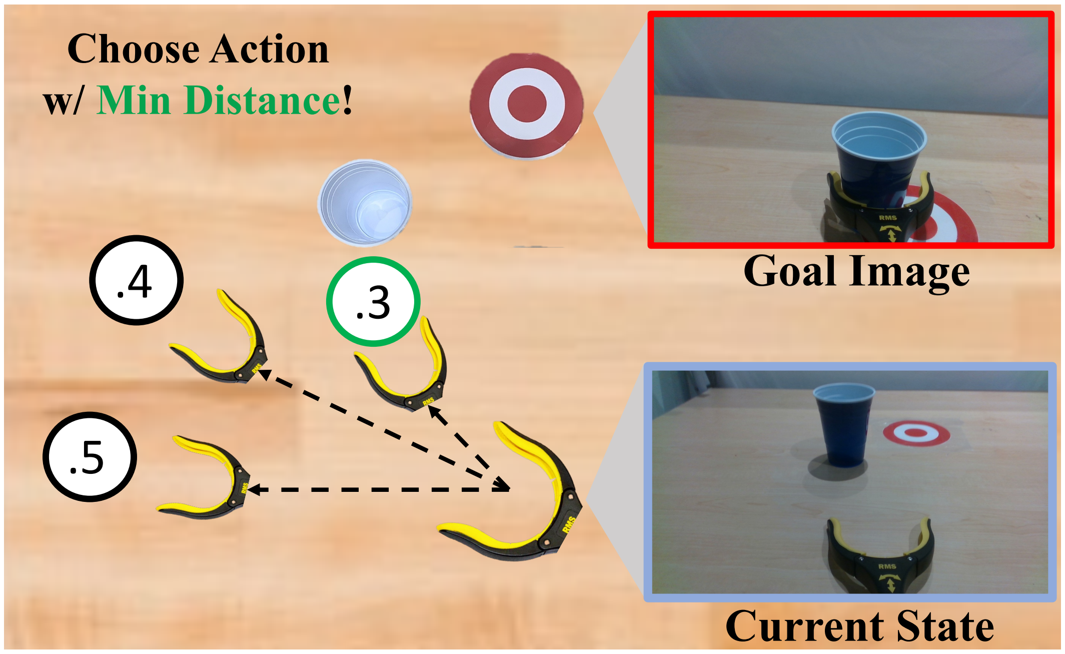


Figure 4.1: This chapter proposes to solve a range of manipulation tasks (e.g. pushing) by learning a functional distance metric within the embedding space of a pre-trained network. This distance function – in combination with a learned dynamics model – can be used to greedily plan for robot actions that reach a goal state. Our experiments reveal that the proposed method can outperform SOTA robot learning methods across four diverse manipulation tasks.

plan robot actions (see Fig. 4.1 for intuition). Our experiments demonstrate that this approach works better than policy learning (via Behavior Cloning), because the pre-trained representation itself does the heavy lifting (thanks to its structure) and we entirely dodge the challenge of multi-modal, sequential action prediction. Furthermore, our learned distance function is both stable and easy to train, which allows it to easily scale and generalize to new scenarios.

To summarize, we show a simple approach for exploiting the information hidden in pre-trained visual representations. Our contributions include:

- Developing a simple algorithm for acquiring a distance function and dynamics model by fine-tuning a pre-trained visual representation on minimal (human collected) data.
- Creating an effective manipulation controller that substantially outperforms State-of-the-Art (SOTA) prior methods from the robot learning community (e.g. Behavior Cloning [179], Offline-RL [119], etc.).
- Demonstrating that our approach can handle four realistic manipulation tasks,

generalize to new objects and settings, and solve challenging scenarios w/ multi-modal action distributions.

4.2 Related Work

Behavior Cloning This chapter adopts the Learning from Demonstration (LfD) problem setting [8, 21, 187], where a robot must acquire manipulation behaviors given expert demonstration trajectories. A standard approach in this space is to learn a policy (π) via Behavior Cloning [179] (BC), which directly optimizes π to match the expert's action distribution. While conceptually simple, realistic action distributions are difficult to model, since they are inherently multi-modal and even small errors compound over time. Thus, policy learning requires extensive intensive network engineering (e.g. transformer architectures [45, 34], multi-modal prediction heads [195, 138], etc.) and/or human-in-the-loop data collection algorithms [179, 109] to work in practice. Instead of learning a policy, we learn a *functional distance metric* that captures the how "close" a state is to reaching a target goal. This lets us build a manipulation controller using a simple greedy planner during test time, without any explicit action prediction!

Offline RL Broadly speaking, the field of Reinforcement Learning [213] (RL) seeks to learn a value/advantage/Q function by assuming access to a reward signal, which "evaluates" a state (e.g. +1 reward for reaching goal). RL algorithms usually require environment interaction to learn, though the field of Offline-RL [132] seeks to extend these systems to learn from offline trajectories. While these approaches have created some impressive robotics demos ranging from locomotion [129, 122] to dexterous manipulation [6, 90, 47], RL methods are data hungry [174, 6], difficult to implement, require extreme "reward engineering" (e.g. Meta-World [248] reward functions are 50 lines!), hampered by unstable learning dynamics [123], and poorly generalize in realistic robotic manipulation settings [48]. In contrast, our method learns a proxy for value functions (i.e. distances) purely from offline data, using representation learning algorithms (from the vision field) that are much more stable.

Learning Visual Rewards Finally, our distance learning approach is analogous to learning visual rewards. Prior work learned success classifiers [194, 242, 223] and/or video similarity metrics [192, 9, 189, 12, 31] from expert video demonstrations. Once learned, these modules were used to parameterize reward functions that could be used to train policies [1], and/or as cost functions for planners [7]. However, these papers mostly consider simple settings (e.g. simulation) or require the test setting to exactly match train time. This is because the learned reward functions are often noisy and/or poorly generalize to new scenes. In contrast, our learned distance metric is stable and can easily adapt to new scenarios, thanks to the pre-trained

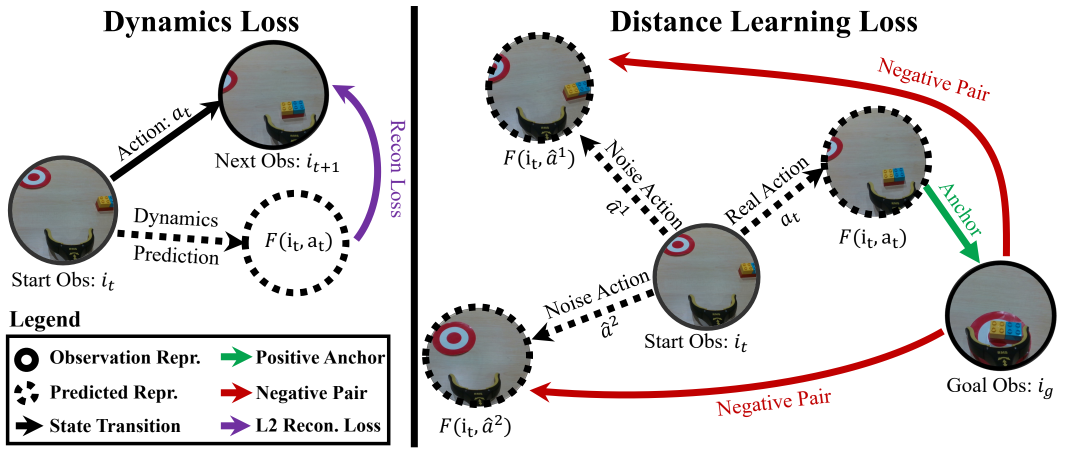


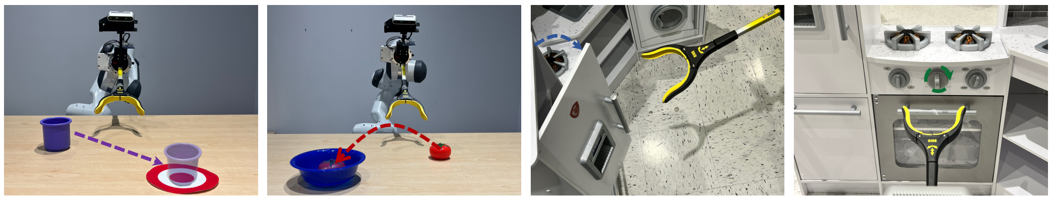
Figure 4.2: We visualize the loss functions used to train our method. The dynamics function is trained via reconstruction loss in embedding space (left). The distance function is trained via contrastive learning, with positive anchors chosen by predicting the next state using the ground truth action, $F(i_t, a_t)$, and predicting negative pairs chosen using noisy actions $F(i_t, \hat{a}^j)$.

network's latent structure. This allows us to solve diverse tasks during test time on a real robot, using a simple (and fast) shooting method planner.

4.3 Methods

Preliminaries This chapter considers learning goal-conditioned manipulation behaviors from image observations. The robot agent is provided a goal observation (I_g) – e.g. target object in robot gripper – and an observation (I_t) for the current time-step t. The robot must process these observations and decide an action to take (a_t). Note that all observations (I) are wrist-mounted RGB camera images with no depth or proprioceptive data. Additionally, the actions (a) are specified as arbitrary $SE(3)$ transforms for the robot's end-effector. Our goal is to learn a robotic manipulation controller, using a set of training trajectories $\mathcal{D} = \{\tau_1, \ldots, \tau_N\}$, where $\tau_i = \{I_g, I_1, a_1, \ldots, a_{T-1}, I_T\}$. The test and train settings are visualized in Fig. 4.4.

Our Approach Our method leverages a pre-trained representation network, R, to encode observations, $i_t = R(I_t)$, and enable control via distance learning. Specifically, we use contrastive representation learning methods [157, 148] to learn a distance metric, $d(i_j, i_k)$, within the pre-trained embedding space. The key idea is to use this distance metric to select which of the possible future state is closest to the goal state. But how do we predict possible future states? We explicitly learn a dynamics function, $F(i_t, a_t)$ that predicts future state for a possible action a_t. During



(a) Pushing (b) Pick and Place (c) Door Opening (d) Knob Turning

Figure 4.3: Our method is tested on 4 different manipulation tasks (pictured above). These tasks test different skill axis, ranging from task-level reasoning (e.g. detect target, move to object before goal, etc.) to fine-grained motor control (e.g. grab top of knob to turn).

test time, we predict multiple future states using different possible action and select the one which is closes to goal state. The following sections describe the learned dynamics module (see Sec. 4.3.1), distance learning method (see Sec. 4.3.2), and test-time controller for robot deployment (see Sec. 4.3.4) in detail. Fig. 4.2 provides a visual depiction of our training algorithm. Pseudo-code and hyper-parameters are presented in Supplement H.3.1.

4.3.1 Dynamics Prediction

An ideal dynamics function would perfectly capture how a robot's actions effects its environment. In our setting, this translates to predicting the next observation embedding, given the current embedding and commanded action: $F(i_t, a_t) = i_{t+1}$. Thus, F can be learned via regression by minimizing reconstruction loss in embedding space: $\mathcal{L}_F = ||F(i_t, a_t) - i_{t+1}||_2$ (see Fig. 4.2, left). This module has two primary uses: (1) during training it acts as a physically grounded regularizer that relays action information into the embedding space, and (2) during test time it enables the robot to plan actions in embedding space. As you will see, both properties are leveraged extensively in the rest of our method.

4.3.2 Learning Task-Centric Distances

Our distance module $d(i_j, i_k)$ seeks to learn functional distances that encode task-centric reasoning (e.g. must reach for object before pushing it) alongside important physical priors (e.g. object is to the left so move left). How can we learn such a distance space? Since we have access to expert trajectories τ, we know that the next state sequentially visited by the expert (i_{t+1}) is closer to the goal state (i_g) than arbitrary states ($\hat{i}$) reachable from i_t – i.e. $d(i_{t+1}, i_g) << d(\hat{i}, i_g)$.

Our insight is that a distance metric with these properties can be learned via contrastive learning. Specifically, we define $d(i_j, i_k) = -cos(i_j, i_k)$, where cos is cosine similarity, and sample a observation-state-goal tuples (i_t, a_t, i_g), alongside random noise action candidates $\hat{a}^1, \dots, \hat{a}^n$ sampled from $\mathcal{D}$. We apply NCE loss and get:

$$\mathcal{L}_d = \frac{exp(-d(F(i_t, a_t), i_g))}{exp(-d(F(i_t, a_t), i_g)) + \Sigma_j exp(-d(F(i_t, \hat{a}^j), i_g))}$$

This loss creates a warped embedding space where i_{t+1} (hallucinated by $F(i_t, a_t)$) is pushed closer towards the goal state i_g, than other arbitrary states reachable from i_t (again hallucinated by F). This process is shown visually in Fig. 4.2 (right), and precisely satisfies our criteria for good distance functions.

4.3.3 Training Details

Both modules are trained jointly resulting in a final loss: $\mathcal{L} = \lambda_d \mathcal{L}_d + \lambda_F \mathcal{L}_F$. Note that F is implemented as small neural networks with 2 hidden layer, while d does not require any extra learned networks since its implemented via contrastive loss in the metric space. The shared representation network, R, we use is ResNet-18 initialized by R3M weights. The ADAM [117] stochastic gradient descent optimizer and back-propagation are used to train the network end-to-end. Please refer to Supplement H.3.1. for additional details.

4.3.4 Test Time Robot Deployment

During test time our learned modules must be able to solve real manipulation tasks when deployed on robot hardware. Thus, we develop a simple inference policy that solves for robot actions using our learned distance and dynamics function. First, a cost function $C_g(i) = d(i, i_g)$ is parameterized given a goal observation I_g and the learned distance function. C_g encodes how far an arbitrary image observation is from the goal image. The optimal action, a_t^*, will take the robot closest to the goal, hence: $a_t^* = \mathbf{argmin}_a C_g(F(i_t, a))$. Note that we use the dynamics function F to predict the next state, since we do not know the real i_{t+1}^* during test time. The cost minimization is performed using the shooting method: random candidate actions, $a_1, \ldots, a_N$, are sampled from $\mathcal{D}$, passed through the dynamics function, and the action with minimum cost $C_g(F(i_t, a_i))$ is executed on the robot. This process is illustrated in Fig. 4.1. Policy execution ends once the distance between the current state and the goal falls below a threshold value: $C_g(i_t) < \lambda$. While a more complicated planner could've been used, this solution was adopted due to its speed, simplicity, lack of hyper-parameters, and good empirical performance.

The final detail to discuss is gripper control for prehensile tasks (e.g. pick and place). Since the gripper status is binary (open/close) and highly correlated with the current state (i.e. close when object in hand), it makes more sense to handle it implicitly rather than as part of the action command a_t. Thus, we trained a gripper action classifier $\mathcal{G}(i_t) \in [0, 1]$ that predicts the probability of closing the gripper given the current image embedding. This is also implemented by adding a single layer to ResNet-18 initialized by R3M weights and can be trained using a small amount (< 100) of human labelled images from the train dataset $\mathcal{D}$. As a

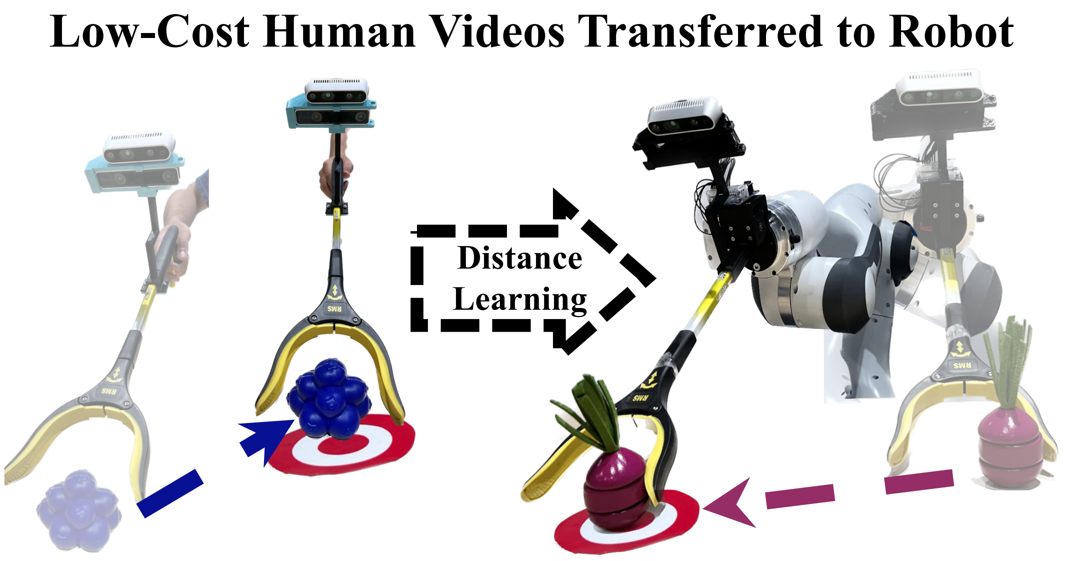


Figure 4.4: In our problem setting we use a low-cost reacher grabber tool (left) to collect training demonstrations. These demonstrations are used to acquire a robot controller purely through distance/representation learning. The final system is deployed on a robot (right) to solve various tasks at test-time.

result, our system can seamlessly handle gripper control in prehensile tasks, without requiring gripper labels for every frame in $\mathcal{D}$!

4.4 Experimental Setup

Our method is tested on four different manipulation tasks (see Fig. 4.3) that require a mix of *high-level reasoning* (e.g. go to object before target), *low level precision* (must carefully grab knob to turn it), and *time-time generalization* (e.g. push novel object). We collect behavior data for each task from human demonstrators as described below. The trained policies are deployed on a real Franka Pandas robot (see Fig. 4.4, right). Additional details on the hardware setup and control stack are presented in Supplement H.3.2.

4.4.1 Tasks

We now introduce the tasks used in our experiments (pictured in Fig. 4.3), and provide more details on their train and test conditions:

Pushing For this task (see Fig. 4.3a) the robot must approach a novel object placed on the table and push it to the goal location (marked by target printout). **Training:**

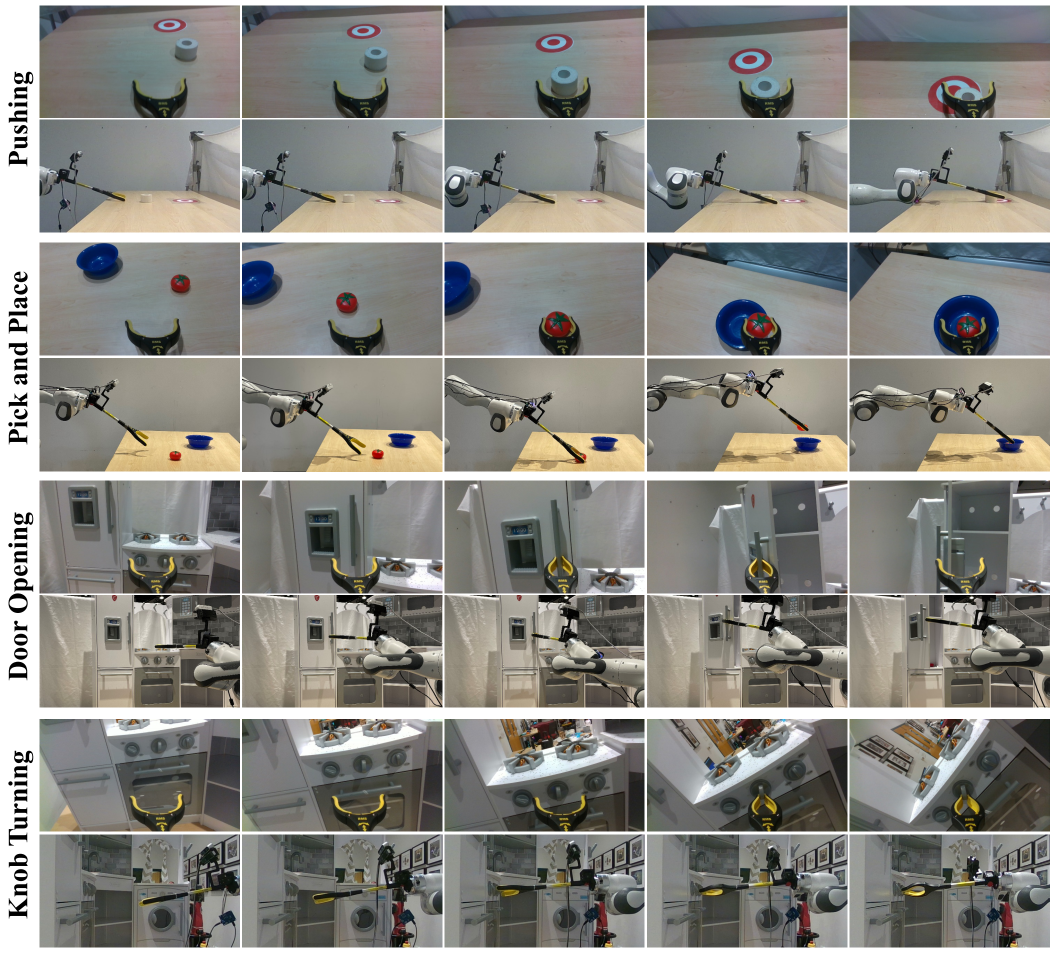


Figure 4.5: Trajectories executed on the robot using our learned distance function. For each task, we show the 1st person view (top) and 3rd person view images (bottom). We show the learned visual embedding can encode functional distances between states for challenging tasks, like pushing, pick and place, door opening, and knob turning.

Our method was trained on a dataset of 100 demonstrations with diverse objects and randomized targets. **Testing:** During test time, the method was evaluated using 20 trails that involved unseen objects and new target printouts placed in randomly sampled positions. A trail is deemed successful if the robot pushes the object onto the target. This is the only task that did not require gripper control.

Pick and Place This task (see Fig. 4.3b) is analogous to the pushing task (described above), except the robot must now pick the object up from its initial location and place it in a target bowl. This task requires more precision than pushing, since grasping an object is harder than localizing it. **Training:** We collect a dataset of 400 demonstrations for training, using randomized train objects and target bowls. **Testing:** The method is evaluated using 20 trails that used novel objects and unseen target bowls (i.e. analogous to pushing). A trail is successful if the robot places an object into the target.

Door Opening The opening task (see Fig. 4.3c) requires the robot to approach a handle, grasp it, and then pull the door open. While conceptually simple, this task requires a great deal of precision, since the commanded actions must (almost) exactly match the direction of the door hinge. **Training:** We created a toy kitchen setting for this task, and collected 100 demonstrations of opening the door when starting from various initial positions. **Testing:** We evaluated on 20 real world test trials, where the robot's initial position and the door's initial state (e.g. position and hinge angle) were randomized. A trial was deemed successful if the door is opened fully.

Knob Turning In this final task (see Fig. 4.3d) the robot must approach a knob in the toy kitchen, and then turn it clock-wise. Similar to the opening task, knob turning requires very precise motor control, since the robot must grab the knob at exactly the right location and carefully rotate in order to turn. **Training:** We collect a dataset of 100 knob turning demonstrations. **Testing:** The method is evaluated on 20 trials, with randomized robot/knob initial positions. Success is determined by if the knob is turned far enough to cause an audible "click."

4.4.2 Robot-Free Data Collection

A major strength of our setup is that training data ($\mathcal{D}$) can come from a different agent than the test time robot. We leverage this property to collect videos *directly* from humans operating a low-cost grabber stick (see Fig. 4.4, left), and use structure from motion [190] to recover actions. This allows us to collect demonstrations *without* an expensive tele-op setup [258, 124] or time-intensive kinesthetic demonstrations (i.e. manually moving the robot by hand). While based on prior work, our stick setup makes two important contributions: it is easier build and use than the highly instrumented setup in Song et. al. [209], and it provides higher quality images and action labels than the GoPro used in Young et. al. [246]. Please refer to Supplement H.3.1 for more information on our training and testing setups. We will release our dataset publicly.

4.5 Experiments

The following experiments seek to validate our distance learning framework on real control tasks (described above), and give some fundamental insights as to why they

Task	**Ours**	BC [179, 246]	IBC [69]	IQL [119]
Pushing	**85%**	70%	70%	65%
Pick and Place	**60%**	30%	30%	45%
Door Opening	**55%**	40%	45%	**50%**
Knob Turning	**40%**	20%	20%	**35%**

Table 4.1: We compare success rates for our method versus the baselines on all four manipulation tasks. Our distance learning method outperforms a suite of representative robot learning baselines.

	Pushing			*Pick and Place*		
# Demos	**Ours**	BC	IBC	**Ours**	BC	IBC
50	**70%**	50%	40%	0%	0%	0%
100	**85%**	70%	70%	**10%**	**10%**	0%
200	90%	80%	**100%**	**20%**	**20%**	10%
400	**100%**	**100%**	**100%**	**60%**	30%	30%
600	**100%**	**100%**	**100%**	**70%**	50%	40%

Table 4.2: We compare our distance learning method versus the behavior cloning baselines (BC [179, 246] and IBC [69]) when trained on varying amounts of data. Distance learning is able to learn faster than the baselines and scale better with data.

work. First, our baseline study (see Sec. 4.5.1) compares our method against representative SOTA techniques in the field. In addition, in Sec. 4.5.2 we analyze a specific scenario with highly multi-modal actions, where our model especially shines. Finally, the ablation study (see Sec. 4.5.3) investigates which components of our system are actually needed for good performance.

4.5.1 Baseline Study

We first note that this chapter uses only demonstration data and no online data. Therefore, we compare our method against three SOTA baselines – ranging from standard behavior cloning, to energy based modeling, and Offline Reinforcement Learning (RL) – that cover a wide range of prior LfD work. The baseline methods are described briefly below. To make the comparisons fair, we parameterize all neural networks with the same R3M representation backbone used by our method, and tune hyper-parameters for best possible performance. Please check Supplement H.3.2 for in depth details.

- **Behavior Cloning** [179, 246] **(BC):** BC learns a policy (via regression) that directly predicts actions from image observations: $\min_\pi ||\pi(I_t, I_g) - a_t||_2$. This provides a strong comparison point for a whole class of LfD methods that focus on learning motor policies directly (i.e. learn policies that predict actions).

- **Implicit Behavior Cloning** [69] **(IBC):** IBC learns an energy based model that can predict actions during test time via optimization: $a_t = argmin_a E(a, I_t, I_g)$. This method is conceptually very similar to behavior cloning, but has the potential to better handle multi-modal action distributions and discontinuous actions.

- **Implicit Q-Learning** [119] **(IQL):** IQL is an offline-RL baseline that learns a Q function $Q(s, a) = Q((I_t, I_g), a_t)$, alongside a policy that maximizes it $\pi(I_t, I_g) = argmax_a Q(s, a)$. Note that IQL's training process require us to annotate our offline trajectories $\mathcal{D}$ with a reward signal r_t for each time-step. While this can be done automatically in our setting (Supplement H.3.2), this is an additional assumption that could be very burdensome in the general case.

Our method is compared against the baselines on all four of our tasks (see Sec. 4.4.1). Results are reported in Table 4.1. Note how our simple distance learning approach achieves the highest success rates on all of four tasks. In addition, our method is much simpler than the strongest baseline (IQL), which requires an expert defined reward signal and uses a more complicated learning algorithm. The final test time performance is visualized in Fig. 4.5 and on our website: https://agi-labs.github.io/manipulate-by-seeing/.

Data Scaling A common cited strength of BC (versus RL) is its ability to continuously improve with additional expert data. Can our method do the same? Table 4.2 compares our distance learning method versus the BC/IBC baselines (on pushing/pick-place tasks), when trained on various amounts of data. Note how our method both learns faster than BC/IBC, and continuously scales with added demonstrations. This suggests that distance learning is a powerful alternative to BC on robotic manipulation tasks, no matter how much training data is available.

4.5.2 Multi-Modality Experiment

A common challenge in policy learning is effectively handling multi-modal action predictions. We propose a simple test scenario that allows us to test our method in this setting without any other confounding variables. Specifically, we create a "obstacle pushing" task where the robot must push an object to the target without disturbing an obstacle placed in the middle (see Fig. 4.6). Observe that there are two

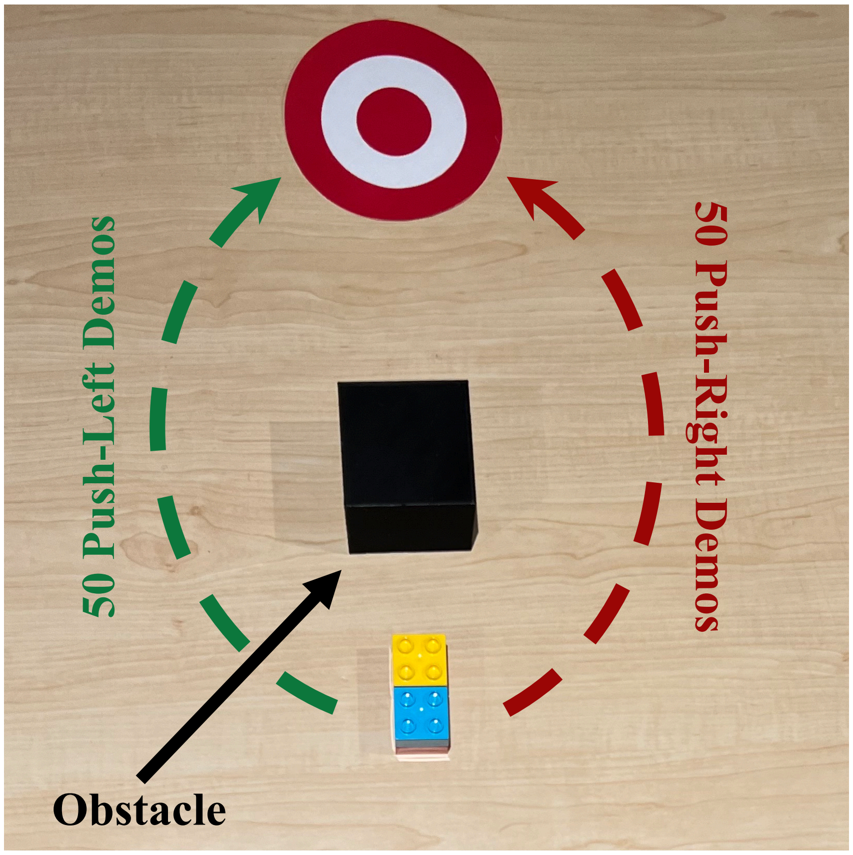


Figure 4.6: Our method can solve tasks with highly multi-modal action distributions that are difficult for the baselines. In this example, our distance learning controller successfully pushes the block around the obstacle, while Behavior Cloning learns to incorrectly push the block forward (i.e. predicts mean action).

equally valid ways that the robot could push the block; to the left or to the right. We collect a training dataset of 100 demos, equally split between going left and right. This creates action multi-modality where the agent must arbitrarily choose between two scenarios in order to successfully perform the task, but will fail if it simply averages the actions and pushes the block forward (into the obstacle). We expect that standard BC will fall into this failure mode, since it is trained using standard L2 action regression. In contrast, our method should still work since the low distance states will move the robot either to the left or right state, thus allowing it to make a discrete choice.

To test this hypothesis, we test our method, the BC baseline, and IQL (the strongest non-BC baseline) on 20 trials in this task setting. Success is judged similarly as to pushing (see Sec. 4.4.1), with the added condition that the robot should not push the obstacles into the target. Our method achieves a **95**% success rate,

	Ours	ImageNet Repr.	No Dynamics [88]
Pushing	**90%**	40%	60%
Pick and Place	**60%**	40%	35%
Door Opening	**55%**	45%	**50%**
Knob Turning	**40%**	15%	**40%**

Table 4.3: This table compares success rates for our method versus the ablations on the all four tasks. As you can see, removing the dynamics module from our method during training and training with a weaker representation both result in worse performance.

versus 0% for BC and 90% for IQL in this setting! As expected, our method is able to greatly improve upon BC in this setting, because it explicitly avoids action prediction. While IQL should not suffer as badly from multi-modality, thanks to its learned value function, our method is still able to outperform it despite being simpler algorithmically and trained without reward signal.

4.5.3 Ablation Study

Our final experiment removes components of our method in order to determine which parts were most crucial for control performance. The *representation ablation* replaces the R3M backbone in our neural network with a pre-trained ImageNet representation (using same ResNet-18 architecture for both). We expect the ImageNet weights to be be weaker, since they were trained on a smaller dataset that did not include ego-centric videos (ImageNet vs Ego4D used by R3M). Additionally, the *dynamics ablation* removes the dynamics module during training, thus leaving the distance model/embeddings without any physical grounding. This is effectively the method from Hahn et. al. [88], which trained a distance embedding for navigation tasks without considering state dynamics at all. We expect this ablation to perform worse on manipulation tasks, because dynamics is more important in this setting v.s. visual navigation.

The two ablations are evaluated and compared against our method on all four test tasks (see Table 4.3). We find that our method strongly outperforms both of our baselines, which suggests that our intuition outline above is correct. In particular, we conclude that stronger representations can directly stronger control performance using our method. Additionally, this shows that the dynamics grounding is important to ensure that our learned distances transfer onto real robot hardware during test time, especially in manipulation tasks that involve object interactions/contact.

4.6 Discussion

This chapter demonstrates that neural image representations can be more than just state representations: a simple metric defined within the embedding space can help infer robot actions. We leverage this insight to learn a distance function and dynamics function using minimal low-cost human data. These modules parameterize a robotic planner that is validated across 4 representative manipulation tasks. Despite its simplicity, our method is able to outperform standard imitation learning and offline-RL methods in the robot learning field. This is especially true in situations involving multi-modal action distributions, where our method entirely outclasses a standard BC baseline. Finally, the ablation study demonstrates that stronger representations directly result in stronger control performance, and that dynamics grounding is important for our scheme to work in practice.

Future Work We hope that this chapter inspires future work in the space of representation learning and robotics. Follow up works should improve visual representations specifically for robotics, by more precisely capturing fine-grained interactions between gripper/hand and objects. This could improve performance on tasks like knob turning, where the pre-trained R3M encoder often struggled to detect small shifts of the gripper relative to the knob. Finally, we hope that our contrastive learning method could be extended to learn entirely without action labels. This would allow us to train on large scale manipulation datasets (e.g. YouTube videos) and transfer the results directly to the control setting.

Chapter 5

An Unbiased Look at Datasets for Visuo-Motor Pre-Training

5.1 Motivation

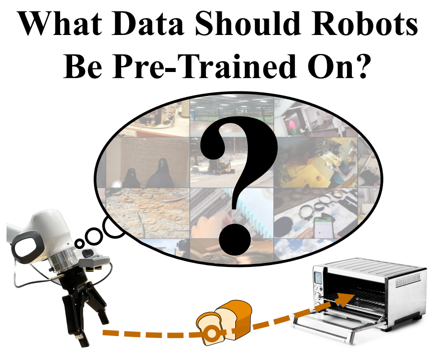


Figure 5.1: Due to the scarcity of diverse, large-scale robotic data, visuo-motor representations – which are necessary to solve tasks (e.g., put bread in toaster) from visual inputs – must be learned from external datasets [154]. But which datasets contain the best priors for robotics? Surprisingly, we find that simply pre-training on standard vision datasets (e.g., ImageNet) can outperform SOTA baseline representations from the robot learning community, despite using roughly 5x less data.

Consider a robot that must perform a manipulation task in an unstructured environment: e.g., toasting a bread slice. To accomplish this, the robot must locate the target objects (bread, toaster, etc.) in the scene and reason about their physical properties (e.g., Center-of-Mass, etc.), using RGB camera inputs. However, the real world has innumerable objects, lightning conditions, and environments that a robot may run into. This incredible range of scenarios makes hand-engineering a vision pipeline impossible. Thankfully, the computer vision and representation learning communities have highlighted a successful paradigm to overcome this challenge: learn end-to-end neural representations *directly from data* [53, 121], which can then be used for downstream vision tasks. We seek to do the same for policy learning.

But what data should these representations be trained on? In an ideal world, we would leverage task-specific robotic data (i.e., trajectories) to jointly learn a visual representation and a controller, using end-to-end reinforcement or imitation learning [213, 131, 178, 89]. Unfortunately, learning visual representations in conjunction with action policies is frequently intractable [33, 106] or requires a large amount of data, that may be too expensive to collect on real hardware. Furthermore, the homogeneity of robotics data (collected in single lab) hinders generalization to novel scenarios, which is the motivation for learning in the first place! To overcome this, the field has trended towards *pre-training* visual representations on large-scale, unlabeled vision datasets, using self-supervised learning algorithms – e.g., Masked Auto-Encoders [94] (MAEs), contrastive learning [158, 96, 35], etc. These pre-trained representations decouple policy learning from perception, allowing us to learn behaviors with far less robotic data [154, 170, 141, 115, 73].

The key insight is that self-supervised visual pre-training can learn useful priors from *out-of-domain* data that will be useful for robotics. Recent work in robot manipulation [170, 154, 140, 141] has investigated different neural architectures and algorithms for learning these priors. However, there is one important commonality – all these methods train (primarily) on the same dataset, Ego4D [81]. Indeed, this seems like an intuitive choice, because: (1) Ego4D contains first-person camera views, which are analogous to the robot's camera; (2) Ego4D focuses on human-object interaction – i.e., it is aligned with the downstream manipulation task; and (3) the dataset offers thousands of hours of video frames to train on. But is this *intuitive bias* empirically tested?

In this chapter, we empirically investigate these research questions from the perspective of robotic manipulation tasks. Specifically, we pre-train a total of 15 representations on various datasets using MAEs [94], a state-of-the-art (SOTA) self-supervised learning algorithm. We then fine-tune each of these representations to solve various manipulation tasks in simulated and real settings via Behavior Cloning (w/ ≤ 50 demonstrations). Our experiments reveal that many intuitive biases and **common assumptions in our field need to be revisited**. Surprisingly, we find that standard image datasets based on curated internet data (e.g., ImageNet [53], Kinetics [207], 100 Days of Hands [197]) can learn stronger visuo-motor representations than egocentric human interaction data (of Ego4D)! In fact, pre-training on the ImageNet compares favorably against SOTA (visuo-motor) baseline representations, which were trained on far more data (e.g. MVP [239] was trained on 2M+ frames) using the exact same algorithm and hyperparameters. This leads us to an importance insight – the pre-training image distribution is far more crucial for effective representation learning than naively increasing the number of images to train on. Building on this, we investigate various schemes for scaling pre-training dataset size while creating a broader image distribution. Our best model improves performance by 30% (v.s. SOTA baselines [141, 239]) on real world robotics tasks and is the direct result of this search. Finally, we show how simple implementation details (like using dropout [211] during evaluation) can have a significant impact on policy performance, and how these trends are poorly captured in simulation studies. Our project code and models are released publicly, and we encourage the reader to view our website for added context[1].

5.2 Related Works

Learning Actionable Representations The robotics field has long focused on learning actionable representations, which focus on task relevant details and are maximally predictive of the actions the robot should take. These representations can be learned end-to-end as part of policy learning, using data collected by expert demonstrations (e.g., Imitation Learning [187]) or the robot itself (e.g., Reinforcement Learning [213]). This paradigm has been successfully applied to a wide range of tasks like in-the-wild grasping [83], bin-picking [112, 133, 165], insertion [131], pick-place [23], and even self-driving [22, 166, 33]. Prior work also added tertiary optimization objectives (e.g. observation reconstruction [152], inverse modeling [46], dynamics modeling [237], etc.) on top of policy learning, in order to make representation learning more efficient. However, all of these techniques share the same flaw: they require a wealth of task-specific robotic data for learning representations.

Self-Supervised Visual Pre-Training Thus, the robotics community has trended towards *pre-training* representations on out-of-domain, vision datasets, (which are both larger and more diverse) and transferring them to robotics tasks. Prior works [154, 140, 239, 170, 141] all seem to follow a common formula: representations are trained using SOTA self-supervised vision algorithms (e.g., contrastive learning [158, 35, 96], masked image modeling [94, 17], etc.) on frames (primarily) sampled from the Ego4D dataset [81]. These representations are then evaluated mostly in sim [220], using a common policy learning framework [154, 141]. These choices may seem reasonable (see Sec. 5.1), but there is surprisingly little evidence backing them. *Importantly, R3M [154] and MVP [170] compared only with supervised ImageNet representations but not apples-to-apples with self-supervised ImageNet representations [94].* Our investigation fills in these critical gaps. We find that representations learned on standard image datasets (like ImageNet) are surprisingly applicable to the robotics space, and that common evaluation/experimental techniques can give a misleading sense of progress.

5.3 Experimental Methods

Our investigation follows a simple formula (see Fig. 5.2). Step 1: We pre-train visual representations on various datasets using the same self-supervised algorithm (masked image modeling). Step 2: We fine-tune each representation for downstream manipulation tasks in both simulated and real (via behavior cloning). For evaluation

[1] https://data4robotics.github.io

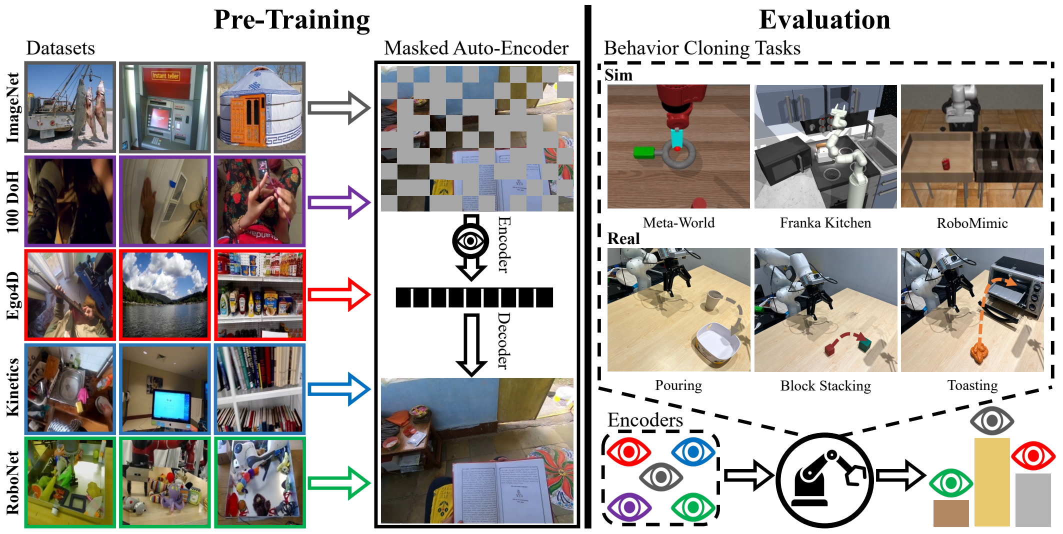


Figure 5.2: Our investigation considers 5 standard datasets from both the computer vision and robotics: ImageNet [53], 100 Days of Hands [197] (DoH), Ego4D [81], Kinetics [207], and RoboNet [44] (left). For each dataset, we pre-train a visual representation on it using the Masked Auto-Encoders (MAE) algorithm [94]. This masked image modeling method works by randomly masking patches in the image, and training an encoder/decoder to reconstruct them (center). Once pre-training is concluded, we fine-tune the representation to various robotics tasks, both in sim and in the real world (right).

of representation quality, we rate based on performance on downstream tasks, with emphasis on performance in the real world.

Visual Pre-Training This project requires a scalable representation learning algorithm that can seamlessly operate on heterogeneous data sources, with the highest possible performance. We chose to use Masked Auto-Encoders (MAEs), a self-supervised representation learning algorithm with SOTA performance on various vision [94, 17, 221, 232, 206, 155, 64], multimodal [101, 10], and robotics tasks [141, 239]. The MAE encoder (E) is a Vision Transformer (ViT) network [256] that produces an embedding vector to represent an input image I: i.e. $E(I) \in \mathcal{R}^{768}$. During training E is tasked to represent I, using only 25% *random patches* sampled from the image. Then a decoder network (D) attempts to reconstruct I in its entirety (see Fig. 5.1, middle). Both E and D are trained end-to-end, minimizing the MSE reconstruction objective: $||D(E(I)) - I||_2$. During training, the visual encoder learns to reason spatially: i.e., it learns how patches relate to each other, and how they can come together to form the final image. Thus, E learns a highly efficient image descriptor that can be transferred to downstream tasks without any algorithmic changes (e.g. no masking needed during transfer). The MAE hyperparameters are described in Appendix H.4.1. Note that they are directly copied from the original MAE work by He *et al.* [94] and shared by prior works in robot learning [141, 239].

Fine-Tuning w/ Behavior Cloning Pre-trained visual representations are fine-tuned to solve downstream tasks of robotic manipulation. To this end, we adopt the paradigm of *Learning from Demonstration (LfD)* [8, 21, 187, 114, 98, 235]. Our goal is to learn a policy π that uses the given observation o_t to predict an optimal action distribution for the task: $a_t \sim \pi(\cdot|o_t)$. Note that the actions a_t are commands sent to the robot controller, while the observations consist of the current image and robot joint information: $o_t = [i_t, j_t]$. The policy π must be learned given a set of expert demonstrations ($\mathcal{D} = \{\tau_1, \ldots, \tau_n\}$), where each demonstration $\tau_i = [(o_0, a_0), \ldots, (o_T, a_T)]$ is a trajectory with optimal observation-action tuples (i.e. collected by a proficient agent).

π is parameterized using a 2-layer network (p), built atop the pre-trained encoder E. The forward pass works as follows: first, the observation image is encoded $E(i_t)$; then j_t is concatenated to the encoding and passed to the policy network – $p(E(i_t), j_t)$. p predicts a policy distribution, which in our case is a Gaussian Mixture Model [144, 172]. During test time, actions are sampled from this distribution and then executed on the robot. The entire policy network (both p and E) is fine-tuned end-to-end (using Adam [117] for $50K$ iterations) to maximize the log probability of actions from the expert demonstrations: $\min_\pi -log(\pi(a_t|E(i_t), j_t))$. This procedure was designed to closely match prior work with two important modifications: we apply dropout to the policy network p, and we apply data augmentation to i_t before passing it to the encoder. Both of these deviations are validated in our experiments (see Sec. 5.5). Please refer to Appendix H.4.2 for the exact hyperparameters.

Evaluation Procedure To evaluate a representation, we apply the above fine-tuning stack separately on 13 sim tasks and 3 real tasks. Each policy's final checkpoint is evaluated on N test rollouts for every task, with novel initializations (e.g., test objects, new initial positions, etc.). All evaluation hyperparameters (e.g., demonstration set, number of test-time rollouts, initial positions, test objects, BC hyperparameters, etc.) are kept fixed within a task. This allows for maximally fair evaluation.

Simulation Tasks: Our simulated tasks set spans a set of 3 MuJoCo [220] environments – MetaWorld [250], RoboSuite [144], Franka Kitchen [86] – that are frequently used by the robot learning community, and the exact setups (e.g., task rewards/success criteria, camera positioning, object sets, demonstration trajectories, etc.) were directly taken from prior work [141, 154, 144] (fully documented in Appendix H.4.3). As a result, our simulated results should be very accessible to the community.

Real World Tasks: While simulation is a useful tool, there is a significant sim2real gap in manipulation. Thus, we designed 3 distinct tasks for real world validation on a Franka Panda Robot (visualized in Fig. 5.2).

1. *Block Stacking* requires the robot to pick up the red block and place it on the green block. This is the simplest of three tasks as the robot only has to adapt to new object configurations during test time. However, the robot still needs to precisely localize and grasp the (small) red block.

2. *Pouring* requires the robot to lift the cup and pour almonds in the target bowl. At test time, the cup and target bowls are both novel objects (unseen during training), and are placed in random positions, requiring the robot to generalize to new visual inputs.

3. *Toasting* is our most challenging task, and it requires the robot to pick up the object, place it in the toaster, and then shut the toaster. At test time, we use a novel object and randomize both the object's initial pose and the toaster's initial orientation. Toasting requires the robot to execute a multi-stage manipulation strategy, while also generalizing to new visual scenarios.

Each of the three tasks use a **shared action space:** Cartesian velocity control; and a **shared observation space:** proprioceptive inputs and 3rd person camera view (visualized in Fig. 5.3, left). We collect $n = 50$ tele-op demonstrations per task. Please refer to Appendix H.4.3 for all other task hyperparameters and our website for task videos: https://data4robotics.github.io.

5.4 Probing Dataset Biases

In our empirical study, we evaluate 5 widely-used datasets as pre-training candidates (see Fig. 5.2, left): ImageNet [53], Ego4D [81], 100 Days of Hands [197] (DoH), Kinetics [207], and RoboNet [44] (see Sec. 5.4.1 for descriptions). We apply our experimental methodology (from Sec. 5.3) on various sub-samplings/combinations of these datasets. First, we conduct single dataset pre-training: i.e., we evaluate a dataset's performance in isolation to empirically determine which is most suited for our diverse downstream manipulation tasks (Sec. 5.4.1). In our second suite of experiments, we analyze how well various combinations of the data perform (Sec. 5.4.2). Finally, we investigate dataset scaling for pre-training, and find that the pre-training image distributions matter most (Sec. 5.4.3).

5.4.1 Comparing Datasets Apples-to-Apples

Before diving into the details, let's take a step back and add context about the datasets we probe.

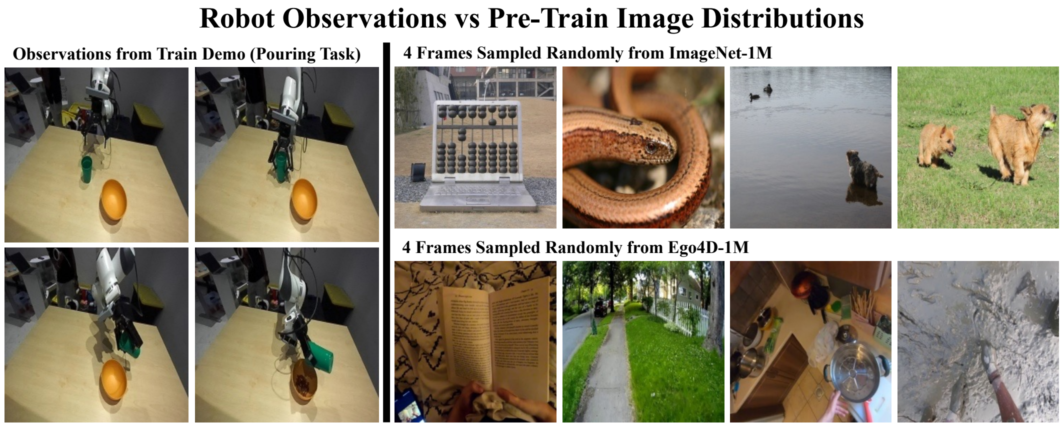

Robot Observations vs Pre-Train Image Distributions
Observations from Train Demo (Pouring Task)
4 Frames Sampled Randomly from ImageNet-1M
4 Frames Sampled Randomly from Ego4D-1M

Figure 5.3: Observations from our pouring task (left) are compared against random pre-training images from ImageNet-1M/Ego4D-1M (right). Note that all the pre-train images are very different from the evaluation task. Nonetheless, the curated, single-object images from ImageNet-1M yield stronger visuo-motor representations than the Ego4D-1M frames do (see Table 5.1).

		Single Dataset Models (1M Images)					**Baselines**		
	Task	ImageNet	Ego4D	Kinetics	100 DoH	RoboNet	Scratch	VC-1 [141]	MVP [239]
Sim	RoboSuite [144]	62%	61%	65%	52%	58%	2%	63%	51%
	MetaWorld [250]	90%	93%	87%	86%	74%	72%	70%	83%
	Franka Kitchen [86]	63%	68%	55%	62%	62%	40%	61%	61%
	Average (Sim)	72%	74%	69%	67%	65%	38%	65%	65%
Real	Block Stacking	68% ± 9.5%	64% ± 9.5%	72% ± 9.8%	55% ± 9.2%	76% ± 8.7%	0% ± 0%	60% ± 10%	52% ± 10%
	Pouring	44% ± 12%	19% ± 9.0%	50% ± 9.0%	19% ± 12%	13% ± 7.6%	0% ± 0%	25% ± 10%	13% ± 7.6%
	Toasting	10% ± 16%	0% ± 0%	10% ± 16%	40% ± 17%	0% ± 0%	0% ± 0%	10% ± 10%	10% ± 10%
	Average (Real)	**41%** ± 7.1%	28% ± 6.9%	**44%** ± 7.1%	**38%** ± 6.9%	30% ± 7.0%	0% ± 0%	33% ± 6.9%	25% ± 6.6%

Table 5.1: **Comparing Datasets Apples-to-Apples.** We compare pretrained representations, learned on 1M images from five datasets. We report success rates after finetuning representations with BC, and the real world evaluations also include standard error (i.e., Success% ± Std. Err.%). For additional context, we benchmark SOTA baselines [141, 239] and a "Scratch" representation with no pretaining. We find that visual representations learned on standard vision datasets with internet images and curation (e.g., ImageNet) provide surprisingly strong performance in the real world.

1. **ImageNet** (ImageNet-1K) contains 1000 train images for each of its 1000 classes. ImageNet is a popular and classical computer vision dataset, i.e., curated carefully from internet images. The broad image distribution may result in more expressive representations (as observed in purely visual tasks like classification). However, ImageNet is focus on centered, single-object, and high-quality internet images. As a result of this domain gap, many believe that ImageNet is an ill fit for robotics.

2. **Ego4D** is a modern, ego-centric, and in-the-wild video dataset, with 3.6K hrs of video collected by humans performing daily tasks. It is conjectured that Ego4D is well suited for robot learning as it contains realistic images that a robot may observe in real-world environments. However, frames collected within the same video tend to look similar to each other, and the lack of curation mean that some object classes (e.g. blocks, cups, etc.) rarely appear.

3. **DoH** is focussed on human hands and contains curated YouTube videos of people manipulating various household objects (e.g. in cooking video). The curation ensures that action classes are balanced and that videos look distinct from each other. Furthermore, the focus on manipulation may help the representations pick useful cues (e.g. where objects can be grasped). However, YouTube videos look quite different from robot's visual observations, so its an open question if priors learned from DoH would benefit downstream tasks in robotics.

4. **Kinetics**(-700) is similar to DoH in that it's curated from YouTube, but its videos contain a much wider distribution of human actions (e.g. with objects and/or other humans) instead of the focus of DoH on hands and manipulation.

5. **RoboNet** contains 13M+ image observations of robots randomly interacting with objects placed in a bin in front of them. RoboNet could be invaluable for pre-training, since its images are highly domain-specific for our use case. But robot data is collected in sterile lab setting, which could cause the representations to overfit to only those specific settings.

It is clear that these five datasets have complex trade-offs that may affect their usability for robotics. However, none of them clearly match the robot observation space (see Fig. 5.3). The only way to settle the question is to undertake an unbiased and empirical study, comparing them apples-to-apples. Thus, we apply our evaluation methodology to 1 Million frames sampled randomly from every dataset. This is easy to do in ImageNet, since it has 1M balanced train images. But for the video datasets (Ego4D/Kinetics/DoH), we first processed them into frames (sub-sampled at 3FPS) and then randomly select 1M images from the whole set. For RoboNet, we followed a similar procedure as used for the video datasets, but randomly sampled 1M image observations instead. Visual pre-training on each of these results in 5 representations that we evaluate on our task suite (via BC). For additional context, we also evaluate a '*Scratch*' model with no pre-training, i.e., randomly initialized weights before BC. We also evaluate pre-trained weights downloaded from two SOTA baselines (VC-1 [141], MVP [239]). Note that both MVP and VC-1 share

	Task	Soup-1M	Soup 2M	Soup-1M + 1M Extra Frames (2M Total) ImageNet	Ego4D	Kinetics	100 DoH	RoboNet
Sim	RoboSuite [144]	64	64	52	53	59	67	58
	MetaWorld [250]	87	89	92	86	87	92	88
	Franka Kitchen [86]	66	67	56	61	64	62	60
	Average (Sim)	72	73	67	67	70	74	69
Real	Block Stacking	76% ± 8.7%	44% ± 10%	72% ± 9.2%	60% ± 10%	76% ± 8.7%	**92**% ± 5.5%	76% ± 8.7%
	Pouring	38% ± 13%	38 ± 13%	32% ± 12%	38% ± 13%	32% ± 12%	**38**% ± 13%	32% ± 12%
	Toasting	10% ± 10%	40% ± 16%	0% ± 0%	10% ± 10%	22% ± 13%	**50**% ± 17%	0% ± 0%
	Average (Real)	41% ± 7.1%	41% ± 7.0%	35% ± 7.1%	36% ± 7.1%	43% ± 7.1%	**60**% ± 6.7%	36% ± 7.1%

Table 5.2: **Marginal Value of Each Dataset.** Soup-{1M,2M} models are trained {1M,2M} images with {200K,400K} images from each of the five target datasets. The models on the right are trained with the Soup 1M images and an additional 1M frames from the target dataset. We find that image distribution matters more than the number of images trained on: Soup-2M does not improve on Soup-1M, but Soup-1M + 1M DoH does. Results are reported as success rates for each task, and the real world evaluations also report standard error (i.e., Success% ± Std. Err.%).

our vision transformer architecture and pre-training recipe of masked image modeling, but were trained on significantly more # of frames (2.5M+), sampled primarily from Ego4D. The results are presented in Table 5.1.

Our first observation is that performance trends from popular simulation benchmarks do not transfer to the real world at all (see Sec. 5.5 for more). Thus, we focus the rest of our analysis on the real world trends, since that is the primary focus of this thesis. The real robot experiments reveal that **ImageNet/Kinetics/-DoH representations all perform better than those trained on RoboNet/Ego4D** (roughly 40% v.s. 30% success rate). Critically, this result goes beyond just MAE pre-training. As we show in Appendix H.4.4, our finding that ImageNet/Kinetics/-DoH performs best *also holds with contrastive pre-training* [35]! This is surprising and important since both Ego4D and RoboNet seem like better matches to the downstream tasks (e.g., RoboNet entirely contains images of robot interactions) and as more works in the research community implicitly assume/expect Ego4D to do better. Note that ImageNet/Kinetics/DoH were all sampled and curated from the internet (using YouTube/image search), so they contain cleaner images with a much greater range of content (e.g., 1000 classes [53] vs 4 robot labs [44]). These unbiased, empirical results strongly suggest that the **pre-training image *distribution* is far more important than the images' *content*.**

5.4.2 Combining Data from Different Sources

Another surprising result from Table 5.1 is that the baselines representations perform worse than the ImageNet/Kinetics/DoH representations, despite being trained on significantly more images. For example, VC-1 pre-trains on ImageNet alongside 2.5M+ images from Ego4D, while using the exact same pre-training strategy that we do. A possible explanation for this discrepancy is that VC-1's representation is functionally very similar to our only-Ego4D ablation, since the majority of its pre-training frames come from Ego4D. Consequently, each batch the encoder sees during pre-training primarily consists of Ego4D frames. The key insight here is that *distribution* of the pre-training set matters more than the sheer number of frames trained on. We experimentally test this hypothesis, and find that VC-1's representation performs only marginally better than Ego4D's (33% vs 28%).

The natural next question is, "*How does one optimally combine datasets for visuo-motor pre-training?*" A simple idea is to proportionally mix the datasets so that the model is pre-trained on an equal number of frames from each dataset. Particularly, we create a "Soup-1M" containing $200K$ images randomly sampled from each of the 5 datasets. We then evaluate this model on our test suite (see Table 5.2, left). Note that the Soup-1M model performs about the same as the ImageNet/Kinetics/DoH models (41%), even though it was trained on a significant amount of Ego4D/RoboNet frames (recall, these performed lowest in Sec. 5.4.1). **This suggests that scaling to multiple datasets can increase robustness, so long as the datasets are kept carefully balanced during pre-training.**

5.4.3 Analyzing the Marginal Value of Each Dataset

Soup-1M provides a sensible first step for combining datasets for visuo-motor pre-training – keeping training set size to 1M using equal proportions of data sources. This leads to the natural next question: "how can we effectively scale dataset size to improve performance?" To answer this question, we'll also need to understand the marginal value of adding additional data to the soup. To answer this, we undertake another empirical study. Particularly, we obtain visual representations on pre-training sets containing the aforementioned Soup-1M along with 1M images from each of the five subject datasets (e.g. Soup-1M + 1M ImageNet frames). In effect, this both scales the size of pre-training dataset, while shifting the train distribution towards that of the subject dataset. For fair comparison, we also train a Soup-2M model (identical to Soup-1M but with $400K$ images per dataset) that tests a naive scaling of the Soup-1M model. All six models are evaluated and results are present in Table 5.2.

As reported in Table 5.2 (left), we find that Soup-2M model performs marginally better in simulation than Soup-1M, and performs exactly the same (on average) in the real world. That is, **data scaling is more nuanced than naively increasing the number of frames**. In contrast, the strongest model, Soup-1M + 1M DoH, is able to perform 20% better than Soup-1M (and 30% better than the strongest baseline) on the real world tasks (bold results in Table 5.2)! Finally, the Soup-1M + 1M {Ego4D/ImageNet/RoboNet} models perform slightly *worse* than Soup-1M, whereas the Soup-1M + 1M Kinetics model performs slightly better. These results are mostly in line with our expectations from Sec. 5.4.1 (e.g. adding more RoboNet data reduces performance, while adding more Kinetics/DoH increases performance).

5.5 Ablating our Experimental Setup

This section presents some insights from our real-world experiments. We find: (1) that old-school dropout regularization is highly effective; and (2) sim *evaluation* does not transfer to real world.

Regularizing Policies w/ Dropout Early in our physical robot evaluations, we noticed that the policies often produced jerky motions that could damage the robot and its environment. Thus, we searched for a simple fix that could improve robustness in the real world. We found that adding dropout [211] to the policy network (w/ $p = 0.2$) significantly improved the robot's qualitative behavior: the commanded motions became smoother, with improved generalization to new scenarios. We quantify this with ablations on the *Block Stacking* task, i.e., fine-tuning the five 1M models (see Sec. 5.4.1) with and without dropout. Note how **adding dropout to visuo-motor policies almost consistently improves policy performance on the physical robot** (see Fig. 5.4, orange bars). However, the opposite effect is observed in simulation. This indicates that adopting sim benchmarks as a (fast) proxy to make policy design choices may warrant caution and a healthy doze of skepticism.

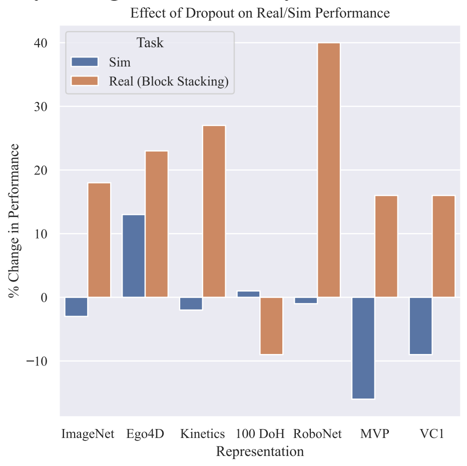


Figure 5.4: **Effect of adding dropout [211] in sim *vs.* real** (block stacking tasks). Dropout frequently harms performance in simulation (blue) but consistently improves real world (orange) success rate. Positive values on Y axis indicate improvement by adding dropout and vice-versa.

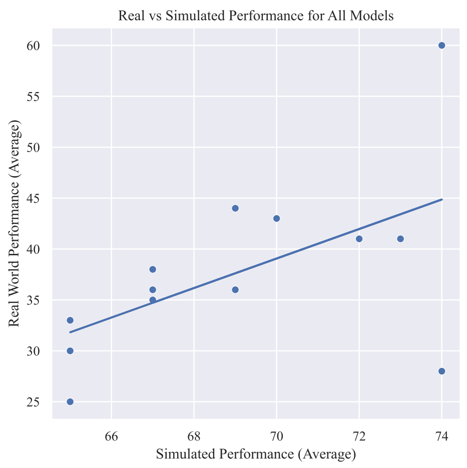


Figure 5.5: **Sim vs. real performance across pretraing datasets.** We plot average model performance, in sim and real, for all the models tested in our study. Note how the sim scores are only weakly predictive of real world performance ($R^2 = 32\%$).

We further test the regularization effect of dropout in this setting using a new task: *Block Stacking Robust*. In this task, a human adversary pushes the cube out of the robot's gripper in the middle of the episode (i.e. right before the robot grasps). This forces the robot to dynamically replan its actions, and adapt to a scenario it never saw during fine-tuning with BC (in the demonstrations). We find that the success rate on *Block Stacking Robust* (average across all models) increases to **24**% from 10% thanks to this regularization.

Analyzing Sim-to-Real Transfer On one hand the sim2real gap in manipulation is well known and on the other it's still very common practice in prior work [154, 141, 239, 140, 92] to draw inferences about pre-trained representations using simulated benchmarks (e.g. CortexBench [141], Franka Kitchen [86], Isaac Gym [239], etc.) In several unbiased experiments we undertook, we have found that trends in simulation are almost entirely disconnected from their real world performance. *First*, the simulation suite predicts that Ego4D is the best representation for robotics, but the real world results consistently disagree with that assessment (see Table 5.1). *Second*, key implementation details in the real world (like Dropout) can actually hurt performance in simulation (see how Dropout hurts sim performance in Fig. 5.4). To objectively investigate this, we plot the sim performance vs real performance for all our models (trained on dataset configurations detailed in Sec. 5.4) in Fig. 5.5. We find that **sim and real performance are almost entirely uncorrelated (a very low $\boldsymbol{R}^2 = 32\%$).** Even if we were to 'cherry-pick' the two most similar sim/real tasks (RoboMimic's block-lift [144] vs our stacking task) the correlation is still very low: $R^2 = 34\%$.

5.6 Discussion

In this chapter, we investigated how dataset biases and implementation choices can affect visual pre-training for robotics. With the modus operandi of masked image modeling [94], our experiments analyzed pre-trained visuo-motor representations trained on 15 different data distributions, sourced from 5 common computer vision and robotics datasets. These models were evaluated on standard sim environments, alongside 3 unique and challenging real world tasks (each with 50+ robot evaluations, for rigor). We find that traditional computer vision datasets (e.g. ImageNet) provide surprisingly strong performance for robotic pre-training. In fact, our simple ImageNet representations outperform both Ego4D representations and representative baselines in the field. The key insight is that the image distribution matters much more than the sheer number of images during pre-training. Guided by this insight, we explore various strategies for scaling data distribution, by carefully mixing data from different sources. As part of this investigation, we train a final model (Soup-1M + 1M DoH) that exhibits a 30% improvement over the baselines on real world robotic tasks. Finally, we analyze our experimental methodology, and show how simple regularization techniques (e.g. dropout [211]) can boost real world performance, and conclude that trends in simulation do not correlate to real world deployment. We hope that our unbiased empirical probes and associated findings will inspire others in the field to study how various sources of offline data can transfer to robotics tasks. To enable future efforts, we released all of our pre-trained representations and evaluation code on our website[2].

Limitations and Future Work While our experiments were extensive, there are some limitations that should be addressed by future work. First, there is no simple theory or experimental test that can predict if a representation will actually work well on the robot after pre-training. In fact, our experiments show that the easiest evaluation technique, i.e. the proxy of simulation, may give a misleading sense of progress in the real world! Thus, it is vital for our community to find faster ways to evaluate representations, and share reproducible results. One possibility is a standardized cloud robotics benchmark [49, 263] that could greatly reduce the load for researchers. Next, our experiments heavily focused on Behavior Cloning combined with MAE pre-training (though we did explore SimCLR pre-training in Appendix H.4.4). Finally, it would be valuable to extend our study in more scenarios (e.g., Reinforcement Learning), and on other robotic tasks, like visual navigation and grasping.

[2]https://data4robotics.github.io

Chapter 6

HRP: Human Affordances for Robotic Pre-Training

6.1 Motivation

A truly generalist robotic agent will need to acquire diverse manipulation skills (ranging from block stacking to pouring) that work with novel objects and are robust to realistic environmental disturbances (e.g., lighting changes, small camera shifts). Due to the scale of this challenge, the field has trended towards learning these agents directly from data [131, 165] (i.e., robot trajectories), which is either collected by expert demonstrators or (via Reinforcement Learning [213]) autonomously by the agent itself. Unfortunately, there are innumerable objects/environments, so roboticists cannot tractably collect enough real-world demonstration data and/or design a simulator that captures all this diversity.

One promising solution for this "data challenge" is for the robot to learn a *suitable representation* from Out-Of-Domain (OOD) data that can be transferred into the robotics domain. For example, prior work [154, 170, 141] trained self-supervised image encoders on large scale datasets of human videos (e.g., Ego4D [81]), using stan-

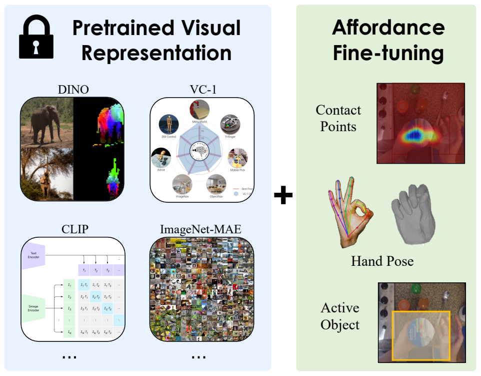


Figure 6.1: Pre-trained representations offer a scalable solution to the robotics data bottleneck [154, 170, 141], but existing methods fail to reliably improve over simple baselines like ImageNet (see Chapter 5 and Burns et. al. [25]). Thus, we present **HRP**, a method that mines affordances (e.g., contact, hand pose, and object labels) from human videos and uses them to improve self-supervised visual encoders. Our best HRP representation consistently outperforms 6 SOTA baselines by $\geq$ **20%** across 5 diverse tasks and 3 camera views.

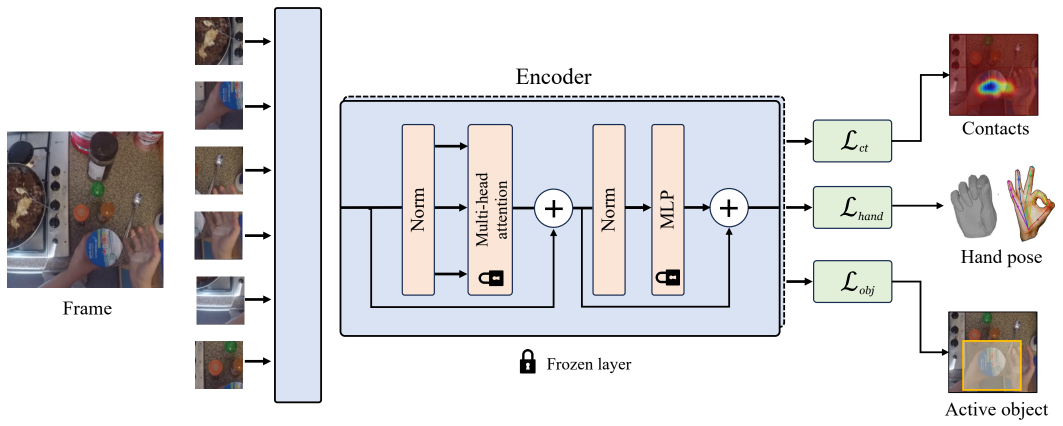


Figure 6.2: HRP fine-tunes a pre-trained encoder to predict three classes of human affordance labels via L2 regression. Specifically, the network must predict future contact points, human hand poses, and the target object given an input frame from the video stream. These affordance labels are mined autonomously from a human video dataset [81] using off-the-shelf vision detectors [197]. Representations produced by HRP are then fine-tuned to solve downstream manipulation tasks via behavior cloning.

dard reconstruction objectives and contrastive learning [158] objectives – e.g., Masked Auto-Encoders [94] (MAE) and Temporal Contrastive Networks [193] (TCN) respectively – developed by the broader learning community. After pre-training, these representations are used to initialize downstream imitation learning [188] algorithms. This formula is extremely flexible, and can substantially reduce the amount of robot data required for policy learning. However, the representations are often only effective when using specific camera views and robot setups. Furthermore, independent evaluations (see Chapter 5 and Burns et. al. [25]) recently showed that these representations cannot improve (on average) over the most obvious baseline – a self-supervised ImageNet representation [94, 53]!

This result is surprising since robotic trajectories and human video sequences share so much common structure: both modalities contain an agent (e.g., human or robot) using their end-effector (e.g., human hand, robot gripper) to manipulate objects in its environment. Ideally, representations trained on this data would learn useful object attributes (e.g., where to grasp a mug), and spatial relationships between the end-effector and target objects. We hypothesize that traditional, self-supervised learning objectives are unable to extract this information from human video data, and that explicitly predicting these object/spatial features would result in a stronger robotic representation (i.e., higher down-stream control performance). Our key insight is that abandoning self-supervision comes at minimal cost – the necessary object and hand labels can be scalably mined using off-the-shelf vision pipelines.

This chapter proposes Human affordances for Robotic Pre-training (HRP), a semi-supervised pipeline to learn effective robotic representations from human video. HRP works in two stages: first, it extracts hand-object "affordance" information – i.e., which objects in the scene are graspable and how the robot should approach them – from human videos using off-the-shelf tracking models [197, 176]. These affordances are then distilled into a pre-existing representation network (e.g., ImageNet MAE [94]), *before* the policy fine-tuning stage. This paradigm allows us to inject useful information into the vision encoder, while preserving the flexibility of self-supervised pre-training – i.e., all labels are automatically generated and the network can be easily slotted into downstream robotic policies/controllers via fine-tuning. To summarize, **we learn stronger robotic** representations by predicting object interactions and hand motion from human video dataset images (see Fig. 6.1). Our investigations and experiments reveal:

1. We present a semi-supervised learning algorithm – HRP– that leverages off-the-shelf human affordance models to learn effective robotic representations from human video. The proposed pipeline strongly outperforms representations learned purely via self-supervision.

2. Applying HRP to 6 pre-existing representations (including ImageNet [53, 94], VC-1 [141], and DINO [27]) substantially boosts robot performance. This conclusion is backed by **3000+ robot trials**, and replicates across 3 camera views, 3 distinct robotic setups, and 5 manipulation tasks!

3. Our ablation study reveals that HRP's three affordance objectives (hand, object, and contact based loss terms) are all critical for effective representation learning.

4. We show that HRP representations generalize across different imitation learning stacks – HRP improves diffusion policy [37] performance by 20%!

5. Our best representation, which increases performance by 20% over State-of-the-Art (SOTA), will be fully open-sourced, along with all code and data.

6.2 Related Work

Representation Learning in Robotics End-to-end policy learning offers a scalable formula for acquiring robotic representations: instead of hand-designing object detectors or image features, a visual encoder is directly optimized to solve a downstream robotic task [131]. Numerous works applied this idea to diverse tasks including bin-picking [112, 133, 165], in-the-wild grasping [83, 209], insertion [49, 131], pick-place [23], and (non-manipulation tasks like) self-driving [22, 166, 33]. Furthermore, secondary learning objectives – e.g., dynamics modeling [87, 237], observation reconstruction [152], inverse modeling [46], etc. – can be easily added to improve data efficiency. While this paradigm can be effective, learning purely from robot data requires an expensive data collection effort (e.g., using an arm farm [133, 112], large-scale tele-operation [23], or multi-institution data collection [44, 40]), which is infeasible for (most) task settings.

To increase data efficiency, prior work applied self-supervised representation learning algorithms on out-of-domain datasets (like Ego4D [81]), and then fine-tuned the resulting representations to solve downstream tasks with a small amount of robot data – e.g., via behavior cloning on a ≤ 50 expert demonstrations [154, 141, 170], directly using them as a cost/distance function to infer robot actions (see Chapter 4), or directly pre-training robot policies from extracted human actions [202, 142]. While this transfer learning paradigm can certainly be effective, it is unclear if these robotic representations [141, 154, 170] actually provide a substantial boost over pre-existing vision baselines (see Chapter 5 and Burns et. al. [25]), like ImageNet MAE [94] or DINO [27]. One potential issue is that roboticists often use the same exact pre-training methods from the vision community, but merely apply them to a different data mix (e.g., VC-1 [141] applies MAE [94] to Ego4D [81]). Thus, the resulting representations are never forced to key in on object/agent level information in the scene. This chapter proposes a simple formula for injecting this information into a vision encoder, using a mix of hand and object affordance losses, which empirically boost performance on robotic tasks by 25%.

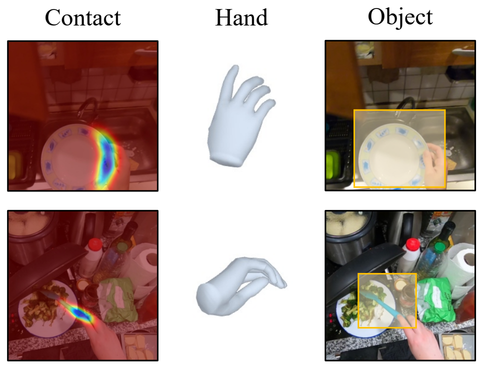


Figure 6.3: We extract 3 affordances – contact heatmaps, hand poses and active object bounding boxes – from human videos.

Affordances from Humans HRP is heavily inspired by the *affordance learning* literature in computer vision [78, 77]. These works use human data as a probe to learn environmental cues (i.e., affordances) that tell us how humans might interact with different objects. These include physical [63, 15, 84, 267, 93, 261, 150] and/or semantic [181, 186] scene properties, or forecast future poses [118, 175, 74, 105, 100, 230, 2, 126, 227, 81, 72, 146, 79] Affordances can also be learned at object or part levels [268, 71, 80, 151, 135, 245]. Usually such approaches leverage human video datasets [81, 41, 43, 41] or use manually annotated interaction data [136, 42, 197]. In addition to these cues, robotic affordances must consider how to move before and after interaction [13, 113]. A simple, scalable way to capture this information is by detecting these cues from human hand poses in monocular video streams [231, 113, 176, 137], which show robots reaching for and manipulat-

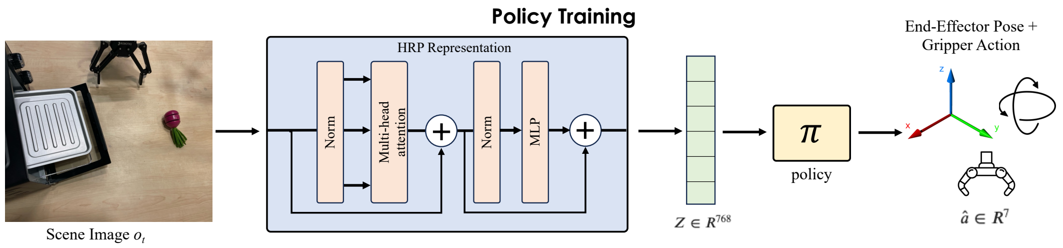


Figure 6.4: We present our policy training pipeline, which uses Behavior Cloning (BC) to train policy π, using optimal expert demonstrations. The image observation (o_t) is processed using our HRP representations resulting in a latent vector z. The policy uses z to predict end-effector velocity actions (delta ee-pose/gripper), which are directly executed on the robot during test-time.

ing diverse, target objects. Our method combines these three approaches to create a human affordance dataset automatically from human video streams. The labels generated during this process are distilled into a representation and used to improve downstream robotics task performance.

6.3 Preliminaries

6.3.1 Visual Representation Learning

Our goal is to learn a visual encoder network f_θ that takes an input image I and processes it into a low-dimensional vector $f_\theta(I) \in \mathcal{R}^d$. This resulting "embedding vector" would ideally encode important scene details for robotic policy learning – like the number and type of objects in a scene and their relationship to the robot end-effector. In this chapter, f_θ is a transformer network (specifically ViT-B [256], with patch size 16 and $d = 768$) parameterized with network weights θ. But to be clear, all our methods are network architecture agnostic.

Self-Supervised Learning The computer vision community has broadly adopted *self-supervised* representation learning algorithms that can pre-train network weights without using *any* task-specific supervision. This can be accomplished using a *generative learning objective* [58], which trains f_θ alongside a decoder network D that reconstructs the original input image input from the representation. Another common approach is *contrastive learning* [158, 96], which optimizes f_θ to maximize the mutual information between the encoding and the input image (i.e., place "similar" images closer in embedding space). In practice, these methods can learn highly useful features for downstream vision tasks [94, 96], but struggle in robotics settings (see Chapter 5). Our goal is to inject these features into an existing self-supervised network, with an affordance-driven fine-tuning stage.

6.3.2 Extracting Affordance Labels from Human Data

Before we can do any fine-tuning, we must first curate a suitable human affordance dataset $\mathcal{D}_H$. Thankfully this task can be done automatically using off-the-shelf vision modules, applied to a set of $150K$ human-object interaction videos from Ego4D (originally sampled by Nair et. al. [154]). These are subsets of larger videos (around 1.2K) videos, which were further broken down into shorter clips. Each clip contains a semantically meaningful action by the human. Each video clip V contains image frames $V = \{I_1, \dots, I_T\}$ that depict human hands performing tasks and moving around in the scene. From these images, we obtain **contact locations**, **future hand p-oses**, and **active object labels** (examples in Fig. 6.3) that capture various agent-centric properties (how to move and interact) and environment centric properties (where to interact) at multiple scales, i.e. contact-level and object-level. The following sections detail how each of these labels were generated.

Contact Locations To extract contact locations for an image I_t (with no object contact), we find the frame $I_j; j > t$ where contact with a given object will begin, using a hand-object interaction detection model [197]. Then, we use I_j to find the active object O_j and the hand mask M_j. The points intersecting M_j and O_j (acquired via skin segmentation) are our contact affordances (C_j). To account for motion between I_t and I_j, we compute the homography matrix between the frames and project those points forward. This is done using standard SIFT feature tracking [264]: $C_t = H_{j,t}C_j$. *In other words, the contact locations denote where in I_t the human will contact in the future.* Note that there could be a different number of points for each contact scenario, which is non-ideal for learning. Thus, we fit a Gaussian Mixture Model with $k = 5$ modes on C_t to make a uniform contact descriptor – defined as the means c_t of the mixture model.For more details on extraction, we refer to Appendix H.5.6.

Future Hand Poses This affordance label captures how the human moves next (e.g., to complete a task or reach an object), as the video V progresses. Given a current frame I_t, we detect the human hand's 2d wrist position (h_{t+k}) in a future frame I_{t+k}, where usually $k = 30$ (empirically determined). This is done using the Frank Mocap [176] hand detector. To correctly account for the human's motion, these wrist points are back-projected (again using the camera homography matrix) to I_t to create the final "future wrist label," $h_t = H_{t+k,t}h_{t+k}$.

Active Object Labels In a similar manner to the contact location extraction, we run a hand-object interaction detection model [197] on V to find the image where contact

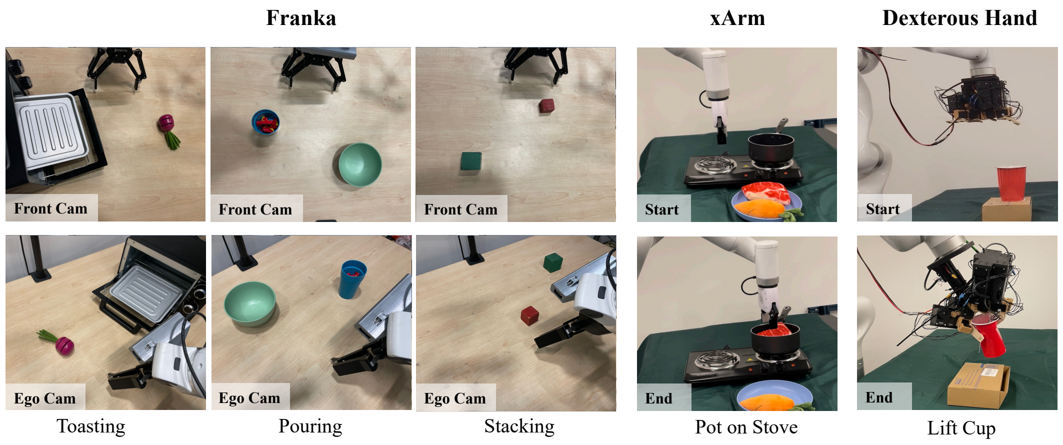


Figure 6.5: Our experiments consider 5 unique manipulation tasks, ranging from classic block-stacking to a multi-stage toasting scenario. These tasks are implemented on 3 unique robot setups, including a high Degree-of-Freedom dexterous hand (right). The 3 camera views shown – front, ego, and side views (for xArm/dexterous hand) – are the same views ingested by the policy during test-time. Note that 3 of the tasks consider 2 unique camera views in order to test for robustness!

began I_c. The same detector is used to find the four bounding box coordinates of the object that is being interacted with, which we refer to as the "active object." These coordinates b_c are then projected to every other frame I_t, using the homography matrix (see above). This results in an active bounding box b_t for each image in V.

6.4 Introducing HRP

A variety of visual pre-training tasks have been shown to help with downstream robotic performance– ranging from simple ImageNet classification [196] to self-supervised learning on human video [170, 154, 140, 141, 159]. These approaches treat human video and simple image frames, and do not explicitly model the rich hand-object contacts depicted by them. In contrast, we believe explicitly modeling the *affordances* [77] in this data could allow us to learn useful information about the agent's intents, goals and actions. Indeed, past work has shown that affordances can act as strong prior for manipulation [265, 110, 203, 253, 13, 28, 102, 19] in general. Moreover, this information can be represented in many different formats, such as physical attributes, geometric properties, interactions, object bounding boxes, or motion forecasting. We observe that most tasks of interests humans perform are with their hands. We thus focus on training our model to predict hand-object interactions and hand motion.

We present HRP, a simple and effective representation learning approach that injects hand-object interaction priors into a self-supervised network, f_θ, using an automatically generated human affordance dataset, $\mathcal{D}_H$ (see above for definitions and dataset mining approach). HRP is illustrated in Fig. 6.2, and the following sections describe its implementation in detail.

6.4.1 Training HRP

The initial network f_θ is fine-tuned using batches sampled from the human dataset: $(I_t, c_t, h_t, b_t) \sim \mathcal{D}_H$, where c_t, h_t, and b_t are contact, hand, and object affordances corresponding to image I_t (see Sec. 6.3.2 for definitions). Some frames may not include all 3 affordances, so we include 3 mask variables – $m_t^{(c)}, m_t^{(h)}, m_t^{(b)}$ – so the missing values can be ignored during training. We add 3 small affordance modules – p_c, p_h, p_b – on top of f_θ that are trained to regress the respective affordances for I_t. This results in the following three loss functions:

$$\mathcal{L}_{\text{ct}} = ||c_t - p_c(f_\theta(I_t))||_2 \tag{6.1}$$

$$\mathcal{L}_{\text{hand}} = ||h_t - p_h(f_\theta(I_t))||_2 \tag{6.2}$$

$$\mathcal{L}_{\text{obj}} = ||b_t - p_b(f_\theta(I_t))||_2 \tag{6.3}$$

The full loss is:

$$\mathcal{L} = m_t^{(c)} \lambda_{\text{ct}} \mathcal{L}_{\text{ct}} + m_t^{(h)} \lambda_{\text{hand}} \mathcal{L}_{\text{hand}} + m_t^{(b)} \lambda_{\text{obj}} \mathcal{L}_{\text{obj}} \tag{6.4}$$

Where the λs are hyper-parameters that control the relative weight of each affordance loss. We empirically found $\lambda_{\text{obj}} = 0.05, \lambda_{\text{ct}} = 0.005, \lambda_{\text{hand}} = 0.5$ to be optimal for downstream performance (see Appendix H.5.5).

6.4.2 Implementation Details

Our affordance dataset ($\mathcal{D}_H$) is at least an order of magnitude smaller than the pre-training image dataset initially used by the baseline representation (e.g., ImageNet has 1M frames v.s. our 150K). To preserve the useful features learned from the larger pre-training distribution, we keep most of the parameters in θ fixed during HRP fine-tuning. Specifically, we only fine-tune the baseline network's normalization layers and leave the rest fixed, which has been shown to be an effective approach [76, 259]. In the case of our ViT-B this amounts to fine-tuning only the LayerNorm parameters γ and β:

$$\text{LayerNorm}(x) = \frac{x - \mu}{\sigma}\gamma + \beta \tag{6.5}$$

These parameters are fine-tuned to minimize $\mathcal{L}$ using standard back-propagation and the ADAM [117] optimizer.

6.5 Experimental Details

Our contributions are validated using a simple empirical formula: first, HRP is applied to each baseline model (listed below). Then, (following standard practice [154, 141]) the resulting representation is fine-tuned into a manipulation policy using behavior cloning. Details for each stage are provided below, and the HRP is illustrated in Fig. 6.2.

Baseline Representations We chose 6 representative, SOTA baselines from both the vision and robotics communities:

1. **ImageNet MAE** was pre-trained by applying the Masked Auto-Encoders [94] (MAE) algorithm to the ImageNet-1M dataset [53]. It achieved SOTA performance across a suite of vision tasks, and is the first self-supervised representation to beat supervised pre-training. We use the standard Masked Auto Encoder training scheme for this, using hyperparmaeters from MAE [94].

2. **Ego4D MAE** was trained by applying the MAE algorithm to a set of 1M frames sampled from the Ego4D dataset [81]. For consistency with prior work, we use the same 1M frame-set sampled by the R3M authors [154]. We use the standard Masked Auto Encoder training scheme for this, using hyperparmaeters from MAE [94].

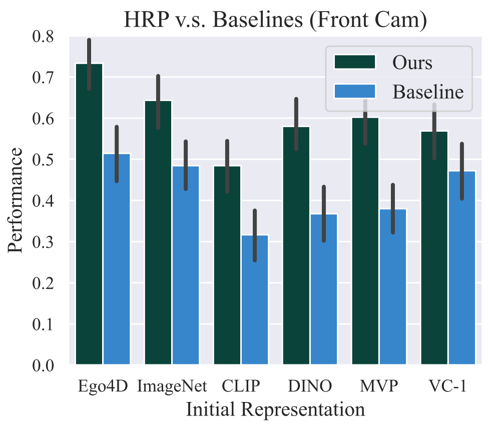
3. **CLIP** [168] is a SOTA representation for internet data. It was learned by applying contrastive learning [158] to a large set natural language - image pairs crawled from internet captions. We used publicly available model weights.

4. **DINO** [27] was trained using a self-distillation algorithm that encourages the network to learn local-to-global image correspondences. DINO's emergent segmentation capabilities could be well suited for robotics, and it has already shown SOTA performance in sim [25]. We used publicly available model weights.

5. **MVP** [170] was trained by applying MAEs to a mix of in-the-wild datasets (100 DoH [197], Ego4D [81], etc.). The authors showed strong performance on various manipulation tasks. We used publicly available model weights.

6. **VC-1** [141] was trained in a similar fashion to MVP, but used a larger dataset mix. It showed strong performance on visual navigation tasks. We used publicly available model weights.

Note that each baseline is parameterized with the same ViT-B encoder w/ patch size 16 (see Sec. 6.3.2), to ensure apples-to-apples comparisons.



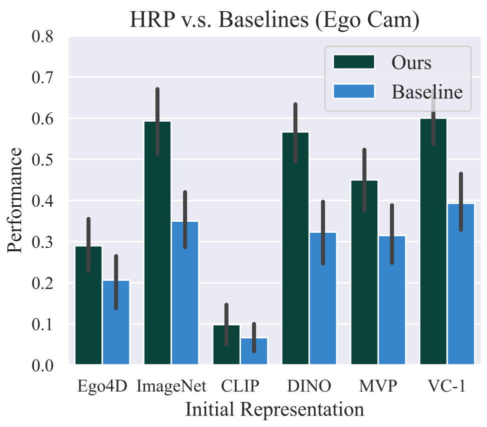


Figure 6.6: We apply HRP to 6 different baseline representations and plot how it affects performance on average across the *toasting*, *pouring*, and *stacking* tasks. This evaluation procedure is repeated using two distinct cameras (shown in Fig. 6.5) in order to test if HRP representation are robust to view shifts. We find that HRP representations consistently and substantially outperform their vanilla baselines, and that this effect holds across both the front (left) and ego (right) cameras. In fact, our strongest representation – ImageNet + HRP– delivers SOTA performance on both views!

Policy Learning Each representation is evaluated on downstream robotic manipulation tasks, by fine-tuning it into a policy (π) using Behavior Cloning [166, 187, 179]. Note that π must predict the expert action (a_t – robot motor command) given the observation (o_t – input image and robot state): $a_t \sim \pi(\cdot|o_t)$. And π is learned using a set of *50 expert demonstrations* $\mathcal{D} = \{\tau_1, \ldots, \tau_{50}\}$, where each demonstration $\tau_i = [(o_0, a_0), \ldots, (o_T, a_T)]$ is a trajectory of expert observation-action tuples. In our case, π is parameterized by a small 2-layer MLP (p) placed atop the pre-trained encoder $p(f(o_t))$ that predicts a Gaussian Mixture policy distribution w/ 5 modes. Both the policy network and visual encoder are optimized end-to-end (using ADAM [117] w/ $lr = 0.0001$ for 50K steps) to maximize the log-likelihood of expert actions: $\max_{p,f} log(\pi(a_t|p(f(o_t))))$. During test time actions are sampled from this distribution and executed on the robot: $a_t \sim \pi(\cdot|p(f(o_t)))$. This is a standard evaluation formula that closely follows best practice from prior robotic representation learning work [144].

Real World Tasks We fine-tune policies for each representation on the 5 diverse tasks listed below, which are implemented on 3 unique robotic setups, including a dexterous hand (illustrated in Fig. 6.5). 50 expert fine-tuning demonstrations were collected for each task via expert tele-operation. Note that the stacking, pouring, and toasting tasks were *evaluated twice using different camera views* to test robustness!

- **Stacking:** The stacking task requires the robot to pick up the red block and place it on the green block. During test time both blocks' starting positions are randomized to novel locations (not seen in training). A trial is marked as successful if the robot correctly picks and stacks the red block, and half successful if the red block is unstably placed on the green block. This task is implemented on a Franka robot, and used both an Ego and Front camera viewpoint.
- **Pouring:** The pouring task requires the robot to pick up the cup and pour the material (5 candies) into the target bowl. During test time we use novel cups and bowls and place each in new test locations. This task's success metric is the fraction of candies successfully poured (e.g., 2/5 candies poured $\rightarrow$ 0.4 success). This task was also implemented on the Franka using Ego and Front cameras.
- **Toasting:** The toasting task requires the robot to pick up a target object, place it in the toaster oven, and shut the toaster. This is a challenging, multi-stage task. During test time the object type, and object/toaster positions are both varied. A test trial is marked as successful if the whole task is completed, and 0.5 successful if the robot only successfully places the object. This is the final task implemented on Franka w/ Ego and Front camera views.
- **Pot on Stove:** The stove task requires picking up a piece of meat or carrot from a plate and placing it within a pot on a stove. During test time, novel "food" objects are used and the location is randomized. A trial is marked as successful if the food is correctly placed in the pot. This task is implemented on a xArm and uses the side camera view.
- **Hand Lift Cup** This task requires a dexterous hand to reach, grasp, and lift up a deformable red solo up. The hand's high dimensional action space ($\mathcal{R}^{20}$) makes this task especially challenging. A trial is marked successful if the cup is stably grasped and picked. This task is implemented on a custom dexterous hand using a side camera view.

6.6 Results

Our experiments are designed to answer the following:

1. **Can HRP improve the performance of the pre-trained baseline networks (listed above)**? Does the effect hold across different camera views and/or new robots? (see Sec. 6.6.1)
2. Our affordance labels are generated using off-the-shelf vision modules – **does distilling their affordance outputs into a representation (via HRP) work better than simply using those networks as encoders?** (see Sec. 6.6.2)

	Teacher ResNet		**HRP Models**	
Front Cam	100DoH [197]	w/ Ego4D	w/ ImageNet	w/ CLIP
Toasting	35% ± 15%	**83%** ± 9%	75% ± 10%	50% ± 11%
Pouring	34% ± 13%	**60%** ± 11%	48% ± 12%	39% ± 11%
Stacking	0%	**77%** ± 10%	70% ± 11%	57% ± 11%
Average	35% ± 10%	**73%** ± 6%	64% ± 7%	48% ± 6%

Table 6.1: This table compares 3 representations trained w/ `HRP` against the teacher ResNet [197] that generated our human affordance dataset (see Sec. 6.3.2). We find that the ResNet teacher under-performs even the worst `HRP` representation (fine-tuned from `CLIP`), *even after excluding the stacking task, which it failed on.*

3. How does HRP compare against alternate forms of supervision on the same human video dataset? (see Sec. 6.6.3)

4. How important are each of the three affordance losses for HRP's final performance? And is it really best to only fine-tune the LayerNorms and leave the other weights fixed? (see Sec. 6.6.4)

5. Can HRP handle scenarios with OOD distractor objects during test time? (see Sec. 6.6.5)

6. Can HRP representations work with different imitation learning pipelines, like diffusion policy [37]? (see Sec. 6.6.6)

Note that all experiments were conducted on real robot hardware, and the models were all tested back-to-back (i.e., using proper A/B evaluation) using 50+ trials per model to guarantee statistical significance. Note that all of our figures and tables report success rates (sometimes averaged across the toasting, stacking, and pouring tasks) alongside std. err. to quantify experimental uncertainty – i.e. success% ± std. err..

6.6.1 Improving Representations w/ HRP

To begin, we evaluate the 6 baseline representations (detailed in Sec. 6.5) on the *toasting*, *pouring*, and *stacking* tasks using the *front camera view*. Then, we apply `HRP` to each of these baselines, and evaluate those 6 new models on the same tasks. Average success rates across all 3 tasks are presented in Fig. 6.6 (left), and the full table is in the Appendix H.5.2. First, this experiment demonstrates that ImageNet MAE is still highly competitive on real-world manipulation tasks, when compared to other self-supervised representations from the vision [81, 27], machine learning [168], and robotics communities [239, 141]. Second, we show that HRP **uniformly boosts performance** on downstream robotics tasks – i.e., baseline + HRP > baseline for every baseline representation considered! Thus, we conclude that the affordance information injected by our method is highly useful for robot learning, and (for now) cannot be learned in a purely self-supervised manner.

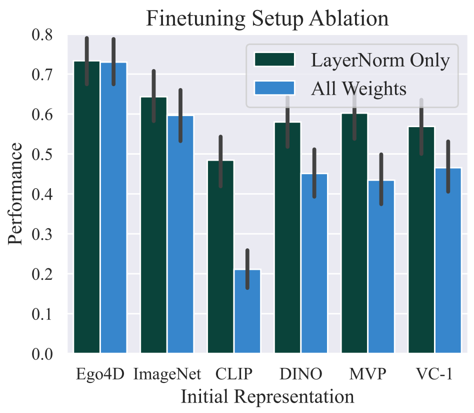


Figure 6.7: This chart applies an ablated HRP method (full fine-tuning) to the 6 baseline representations, and compares their average performance v.s. standard HRP representations on the *toasting*, *pouring*, and *stacking* tasks (front cam). We find that LayerNorm only fine-tuning is almost always superior.

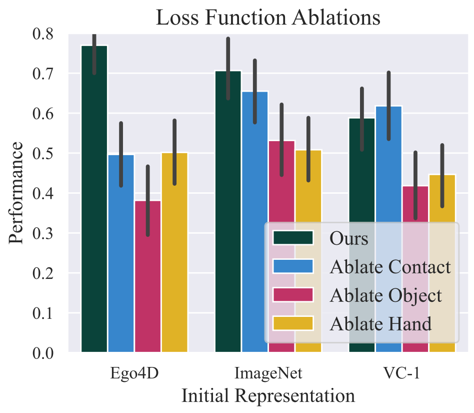


Figure 6.8: We drop each of the 3 losses in HRP, and compare the ablated method's average performance (across the *toasting*, *pouring*, *stacking* tasks) against full HRP representations. We find that the object and hand losses are critical for good performance, but the contact loss only makes a significant impact on the Ego4D base model.

Second Camera View A common critique is that robotic representations perform very differently when the camera view (even slightly) changes. To address this issue, we replicated the first experiment using a radically different *ego view*, where the camera is placed over the robot's shoulder (i.e., on its "head"). While perhaps a more realistic view, it is significantly more challenging due to the increased robot-object occlusion. Average success rates are presented in Fig. 6.6 (right), and a per-task breakdown is in Appendix H.5.3. Note that our findings replicate almost exactly from the front camera view. The ImageNet MAE representation is still competitive with the other baselines, and applying HRP uniformly improves the baseline performance. In addition, we find that **HRP injects a higher level of robustness to camera view shifts**, when compared to the baselines. For example, we find that ImageNet + HRP performs the same on the ego and front camera, even though the ImageNet baseline clearly prefers the front cam. This general effect holds (to varying degrees) across all six baselines!

	Ego4D		ImageNet	
	w/ HRP	Baseline	w/ HRP	Baseline
Pot on Stove	**50%** $\pm$ 17%	40% $\pm$ 16%	**60%** $\pm$ 16%	40% $\pm$ 16%
Hand Lift Cup	**50%** $\pm$ 17%	40% $\pm$ 16%	**50%** $\pm$ 17%	30% $\pm$ 15%

Table 6.2: We present results of `Ego4D + HRP` and `ImageNet + HRP`, as well as the respective baselines on the x-Arm (Pot on Stove) and a dexterous hand task (Lift Cup). We see that `HRP` can even boost performance in multiple morphologies, including a high-degree of freedom dexterous hand [201].

Scaling to More Robots Finally, we verify that `HRP` representations can provide benefits on other robotic hardware setups. Specifically, we compare `Ego4D + HRP` and `ImageNet + HRP` versus the respective baselines on the *Pot on Stove* (xARM) and *Hand Lift Cup* (dexterous hand) tasks. Results are presented in Table 6.2. Note that `HRP` representations provide a consistent and significant performance during policy learning on these radically different robot setups, which both also use a unique side camera view. This gives us further confidence in HRP's view robustness, and demonstrates that these representations are not tied to specific hardware setups, and can scale to complex morphologies like dexterous hands.

6.6.2 Distillation w/ HRP Improves Over Label Networks

It is clear that applying `HRP` to self-supervised representations results in a consistent boost. However, the hand, object, and contact affordance labels for `HRP` themselves come from neural networks (see Sec. 6.3.2) – specifically we use the ResNet-101 [97] detector from 100DoH [197] as a label generator for our active object and contact affordance. The hand affordance we use comes from FrankMocap [176], which uses 100DoH [197] as a base model. Thus, does distilling labels from this detector via `HRP` actually provide a benefit over simply using the 100DoH model itself as a pre-trained representation? To test this question, we fine-tune policies on the toasting, pouring, and stacking (front cam) tasks and compare them against `HRP` applied to `ImageNet`, `Ego4D`, and (the weakest model) `CLIP` (see Table 6.1). In all cases, our representation handily beats the 100DoH policy. So while the affordance labels can dramatically boost policy learning (via `HRP`), the source/teacher models are not at all competitive on robotics tasks.

6.6.3 Comparing Against Alternate Forms of Supervision

We now analyze if HRP's losses are betters suited for robotics tasks than an alternate supervision scheme. To be clear, the previous results already demonstrated that `HRP + Ego4D` out-performed the `Ego4D` baseline by up to 20% (see Fig. 6.6; left), despite being sourced from the same image data. However, it could be that the

Initialization	**w/ HRP**	**MAE Initialization**
Ego4D	**40%** $\pm$ 15%	15% $\pm$ 11%
ImageNet	**40%** $\pm$ 15%	**40%** $\pm$ 15%

Table 6.3: This table compares `Ego4D + HRP` and `ImageNet + HRP` representations against their respective baselines on a *stacking w/ distractors* task. Here the robot must successfully complete the usual stacking task, when extraneous objects (an orange carrot, and a green bowl) are added to the scene. We find that `Ego4D + HRP` improved over its baseline on this task, but `ImageNet + HRP` performed the same as its baseline.

	Ego4D		**ImageNet**		**CLIP**	
	+ HRP	+ Semantic	+ HRP	+ Semantic	+ HRP	+ Semantic
Toasting	**83%** $\pm$ 9%	25% $\pm$ 13%	**75%** $\pm$ 10%	40% $\pm$ 14%	**50%** $\pm$ 11%	20% $\pm$ 13%
Pouring	**60%** $\pm$ 11%	30% $\pm$ 13.4%	**48%** $\pm$ 12%	26% $\pm$ 11%	**39%** $\pm$ 11%	22% $\pm$ 10%
Stacking	**77%** $\pm$ 10%	30% $\pm$ 11%	**70%** $\pm$ 11%	40% $\pm$ 12%	**57%** $\pm$ 11%	30% $\pm$ 13%
Average	**73%** $\pm$ 6%	28% $\pm$ 7%	**64%** $\pm$ 7%	35% $\pm$ 7%	**48%** $\pm$ 6%	24% $\pm$ 7%

Table 6.4: We create `Semantic` representations by fine-tuning the `Ego4D`, `ImageNet`, and `CLIP` baselines using a classification loss, instead of HRP's affordance loss. Note that the exact same Ego4D clips (see Sec. 6.3.2) are used during semantic fine-tuning, thanks to object class labels generated automatically by Detic [266]. The sematic representations were evaluated (using the same BC pipeline) on the Toasting, Pouring, and Stacking tasks, and compared against their `HRP` counterparts. Success rates (and standard error) are reported above. We find that the affordance supervision provided by HRP is vastly superior to the semantic alternative.

additional fine-tuning step with the $100K$ filtered interaction clips is responsible, and the specific affordance losses are not key. To test this, we ran a modified version of HRP using a semantic classification loss, instead of our affordance hand-object losses. The ground-truth labels for each image were obtained using the Detic object detector [266]. We then similarly fine-tuned the `ImageNet`, `Ego4D`, and `CLIP` baseline representation using these labels, and compared them against the respective `HRP` models on the toasting, pouring, and stacking tasks. The results are presented in Table 6.4 We find that the `HRP` models perform significantly better on every task. Thus, we conclude that HRP's affordance losses play an important role in boosting performance (i.e., it's not just data or extra fine-tuning).

6.6.4 What Design Decisions are Important?

The following section ablates the key components of `HRP` to evaluate their relative importance. First, we apply `HRP` to each of the 6 baseline representations again, but this time none of the weights are kept fixed (see Sec. 6.4.2). These representations are fine-tuned on the toasting, stacking, and pouring tasks (front cam), and compared against the original HRP representations in Fig. 6.7. Note that fine-tuning all the layers results in a substantial performance hit on average, and this trend is consistent regardless of the base representation! Thus, we conclude fine-tuning only the layer norms when applying HRP is the correct decision.

Next, we ablate each of the affordance losses in Eq. 6.4, by applying HRP three times: once with $\lambda_{\text{ct}} = 0$, then with $\lambda_{\text{hand}} = 0$, and finally $\lambda_{\text{obj}} = 0$. This process is repeated using 3 different base models; ImageNet, Ego4D, and VC-1. This creates 9 ablated models (3 losses x 3 initializations) that are compared versus the full HRP models on the toasting, pouring, and stacking tasks. The average results are presented in Fig. 6.8, and the full, per-task breakdown is presented in the Appendix H.5.4. We find that removing the object (Eq. 6.3) and hand (Eq. 6.2) losses uniformly results in significant performance degradation. Meanwhile, the contact loss (Eq. 6.1) only provides a significant boost for the Ego4D base model but does not affect the others. Thus, we conclude that object and hand losses are critical for our method, while the contact loss is more marginal, most likely due to the fact that extraction of contacts is a relatively noisy process.

6.6.5 Novel Distractors During Test-Time

We evaluate the performance of HRP and baseline approaches in OOD settings, by adding extraneous "distractor" objects (an orange carrot and a light green bowl) in the stacking task. The robot must successfully ignore the distractor and complete the task. Results are presented in Table 6.3. We found that both ImageNet + HRP and ImageNet had the same level of robustness to distractors. Meanwhile, Ego4D's performance dropped substantially, while Ego4D + HRP remained robust. Our hypothesis is that human data by itself does not contain enough information to allow for OOD tasks. However, using HRP allows for more focus on task-relevant features, even when the representation is trained on less diverse data.

6.6.6 Evaluating w/ Diffusion Policy

Finally, we analyze if HRP representations offer improvements when using a radically different imitation learning framework, like diffusion policy [37]. Specifically, we adopt the original U-Net action prediction head and environment setup from Chi et. al. [37], but replace their ResNet visual encoder (inspired from RoboMimic [144]) with our HRP + ImageNet ViT-B model. Then we compare this HRP enhanced diffusion policy implementation, against (diffusion agents which use) both the original ResNet encoder and the baseline ImageNet ViT-B. Results for the (Franka) stacking, pouring, and toasting tasks are presented in Table 6.5. We find that HRP + ImageNet significantly improves over both alternatives (76% for HRP v.s., 56% for Chi et. al.'s implementation [37]), despite using a radically different imitation learning objective/setup! Thus, we conclude that HRP representations can boost performance across different setups.

	HRP + ImageNet	**ImageNet**	**ResNet [144, 37]**
Toasting	**80%** $\pm$ 13%	60% $\pm$ 16%	55% $\pm$ 16%
Pouring	**74%** $\pm$ 14%	48% $\pm$ 16%	38% $\pm$ 16%
Stacking	**75%** $\pm$ 13%	**70%** $\pm$ 15%	50% $\pm$ 17%
Average	**76%** $\pm$ 8%	59% $\pm$ 9%	48% $\pm$ 9%

Table 6.5: This table tests if HRP representations can boost performance when using a radically different imitation learning framework – namely Diffusion Policy [37]. We evaluate diffusion policies (following the U-Net + state action formula described by Chi et. al [37]) on the toasting, pouring, and stacking tasks using 3 different visual encoders: the default ResNet encoder from RoboMimic [144], the `ImageNet + MAE` baseline, and our `HRP + ImageNet` features. We find a clear improvement when using HRP weights, which suggests that HRP is applicable to different imitation learning frameworks!

6.7 Discussion

In this chapter, we investigate human affordances as a strong prior for training visual representations. Thus, we present HRP, a semi-supervised pipeline that extracts contact points, hand poses and activate objects from human videos, and use these affordances for fine-tuning representations. HRP improves base model performance drastically, for five different, downstream behavior cloning tasks, across three robot morphologies and three camera views. All components of our approach, including LayerNorm tuning, our three affordances, and our distillation process (from affordance labels to representations) are important for the model's success. One key limitation of this approach is that it has only been tested on imitation settings in this chapter. In the future, we hope to not only scale this approach to many more tasks and robot morphologies, but also incorporate HRP in other robot learning paradigms such as reinforcement learning or model based control.

Chapter 7

Conclusions and Future Work

This thesis explored 5 distinct strategies for acquiring data-driven robotic controllers from increasingly diverse data sources. First, Chapter 2 demonstrated how a simple behavior cloning loss can learn a complex, one-shot imitation policy, thanks to a high-capacity Transformer neural network architecture. This approach outperformed prior baselines that required significantly more complicated alogirithms and network architectures, but was prevented from scaling due to its reliance on expert demonstrations. To solve this issue, Chapter 3 presented a method for acquiring a *generalist* robotic controller by pre-training across data collected from different labs and institutions. This general controller could be fine-tuned to solve novel tasks in new environments in a more data efficient manner. Many modern robotics projects – like Open-X [40], Octo [156], and DROID [116] – were heavily inspired by this approach. However, robot lab data is still limited in terms of diversity. Thus, the third section of this thesis explored different mechanisms for pre-training effective robotic represtations from human data, and transferring them to the robotics regime. Chapter 4 found that self-supervised representations can strongly transfer to downstream robotics tasks, but the effect only materializes when given sufficiently diverse pre-training data (see Chapter 5). Finally, Chapter 6 proposed a semi-supervised fine-tuning method that could boost downstream task performance on 6+ representations, by injecting human affordance priors into the pre-trained representations. Altogether, we found that human data can boost downstream performance on robotic tasks by over 50%, especially in the low (robot) data regime.

7.1 Tips and Tricks for Better Behavior Cloning

In addition to scaling data, this thesis demonstrated how Behavior Cloning [188] (BC) – one of the simplest algorithms for robotic policy learning – can efficiently solve complex manipulation tasks (e.g., toasting in Chap. 5 and dexterous grasping in Chap. 6). These results were built on the back of critical engineering decisions that often go unmentioned in academic writing/research. Thus, this thesis concludes with an unofficial list of tips and tricks (below) for the benefit of future practitioners. Please note that these tricks are not rigorously proven, but are consistent trends from my experiments/experience. Refer to the code-base from Chap. 5 and Chap. 6, which provides a clean implementation w/ many of these tricks: `https://github.com/sudeepdasari/data4robotics`.

- **Use a High Capacity Network:** Parameterize the policy with a sufficiently large network. In particular, make sure that enough capacity is allocated to action prediction (not just image processing) – a small action prediction MLP head is often insufficient for multi-modal action prediction (e.g., via diffusion). This lesson holds even in low data regimes ($\leq$ 100 demos). For example, Chap. 5 trained successful BC policies by fine-tuning a 90M parameter ViT-B network with 50 demos, and Chap. 6 combined a ViT-B image encoder with a 50M parameter U-Net action head to learn a SOTA diffusion policy with 100 demos.

- **Process Cameras with Separate Encoders:** If the robot has multiple cameras (like a wrist + front camera), use separate encoders weights for each camera during image tokenization. For example, use two separate ResNet-18 (or larger if memory allows) networks to tokenize frames from each camera, before passing the combined features to a transformer for action prediction [260, 156, 37]. This allows the policy to learn separate features for each viewpoint.

- **Train w/ Large Batch Sizes:** Larger batch sizes result in more stable training and (often times) stronger policies. Try to use $B \geq 128$ at minimum, but scaling beyond that (e.g., $B \geq 512$) is advisable for final runs.

- **Predict Longer Action Chunks:** Given an observation o_t, predict a chunk of H actions $\{a_t, a_{t+1}, \ldots, a_{t+H}\}$, instead of just the next action a_{t+1}. This often results in more stable policy learning, since it allows the network to better model multi-modal action scenarios (like pauses). It also allows for trajectory level smoothing during test time, which can significantly improve performance [260]. If possible, make H large enough to fit 2 seconds of robot action predictions (so $H = 100$ for a 50Hz action space). Feel free to add a loss mask, to accommodate shorter trajectories during training.

- **Fine-Tune the Whole Network:** Avoid freezing parameters during behavior cloning, even when starting from pre-trained features. This is especially important for image encoders, since precise visual localization is required for most manipulation tasks. The effect is often quite significant: e.g., unfreezing parameters can boost performance by *over* 20%+!

- **Experiment with Velocity and State Actions:** It may be better to predict either velocity or state actions, depending on the exact choice of policy architecture, loss function, and robot platform. It's best to experiment with both options, and choose the best. Ideally, your demonstration collection code will automatically collect both types of action labels, to make this switch seamless.

- **Normalize the Action Space:** Be sure to normalize the action space (e.g., via Gaussian whitening, or max/min bounds normalization) based on the action statistics seen in the training demonstrations. This both stabilizes policy learning, and prevents dangerous scenarios during test time (e.g., policy can't erroneously command large actions).

- **Explicitly Handle Action Multi-Modality:** Explicitly model action multi-modality in your policy architecture – e.g., via action chunking [260] (see above), diffusion [37] (Chap. 6 uses DDIM w/ 100 train iters. and 10 evaluation iters.), and/or mixture model policies [144] (Chap. 5 predicts a Gaussian Mixture Model w/ 5 modes). Failing to do this can result in indecisive and ineffective policies.

- **Carefully Process Proprioception:** Action proprioception can greatly improve policy performance, but naively adding it to the network can result in heavy overfitting. We found that simply adding dropout (w/ $p = 0.2$) to the input signal can solve this issue (see Chap. 5).

- **Collect Diverse Demonstrations:** Repetitive demonstrations provide little value, so ensure that the training demonstrations cover the full range of task scenarios the policy may see during test time. One solution for this is marking each train position (e.g., w/ non-permanent pencil marks), and ensuring that they cover $\geq 80\%$ of the task workspace.

- **Collect Consistent Demonstrations:** Try to limit action multi-modality, by demonstrating behaviors in a consistent way: e.g., always pick up mug from the handle, grasp objects from the same angle/direction, and avoid pauses. This is especially important in the low-data regime ($\leq$ 100 demos), since multi-modal distributions are hard to fit with little data.

- **Create a Low-Data Sanity Test:** Use a simple task (like block stacking) for fast policy evaluation in the real world. This will allow for quick iteration and debugging on an actual robot.

- **Leverage Sim for Debugging Only:** Despite the large sim2real evaluation gap (see Chap. 5), it is still possible to detect major breaking bugs with simulators. Use it as a rough sanity check, but do not *evolve* the project based solely on signals from simulation.

7.2 Future Work

Despite the progress made in recent years (by the field and in this thesis), robotic manipulation still remains far from solved. These models still make frequent errors, especially when object and scene details change significantly from the training data distribution. The most straight-forward, scalable solution to this problem, is to continue broadening the training data distribution. This would entail both collecting, annotating, and curating more diverse manipulation datasets, and also enabling robots to learn more and more attributes (e.g., where to grasp, what trajectories to take, etc.) directly from human data. This thesis took initial steps in this direction, but it will require a large community-wide effort to fully land this vision.

However, scale is not the only possible solution – it is equally (if not more) important to make the learning process itself more efficient. This could be accomplished by fine-tuning a large model from another domain (e.g., large vision language model), which may have learned general world information and priors, to solve robotics tasks. Another reasonable idea is to inject physics and visual priors into the learning process. For example, the robot could learn 3D aware representations that help it grasp objects more precisely, even as camera views shift and change. Finally, improving the robotic hardware itself, could make learning far faster and more efficient in the long term. This could take the form of adding a second arm and/or utilizing dexterous hands, which could solve more tasks with simpler strategy. One could also integrate tactile signals (like RealSense [20]) into the learning process in order to make fundamental tasks like grasping easier. In the long term, these approaches could help robots solve more compelling tasks, with far less training data.

Chapter H

Appendix

H.1 Transformers for One-Shot Visual Imitation

H.1.1 Baseline Comparisons: Multi-Object Environments

While the prior experiments showed our model could successfully generalize to new task instances, can it also generalize to new tasks including unseen objects? To answer this question the baseline comparisons (described in Section 2.4.2) are repeated in environments with multiple objects. Importantly, the objects used during test time are unseen during training.

Environment Description: The *multi-object environment* is cloned from the base environment (presented in Section 2.4.1) and modified to include more objects with different shapes and textures. Note that while object appearance and shape is randomized, dynamical properties - like friction - are kept constant since they cannot be visually judged. The simulator has 30 unique objects, 26 of which are seen during training and 4 are only used during test time.

Data Collection Process: To collect train tasks, 4 objects are sampled from the 26 train objects, which results in an environment with 16 tasks. For each task, multiple task instances composed of expert demonstration videos (v_i) and imitator trajectories (t_i) are collected using the same methodology as before (refer to Section 2.4.2 and Section 2.4.1). In total, the train dataset is composed of 1200 tasks (2400 task instances total). Test tasks are also sampled in the same fashion as before, except using the 4 new objects. Our method is able to succeed at the object picking stage of the tasks $50 \pm 9.9\%$ of the time which is ~ 2x better than the best baseline (contextual-LSTM) which only picks $23 \pm 8.4\%$ of the time. Unfortunately, all methods (including ours) often place objects in the wrong bin resulting in final success rates of $23 \pm 8.4\%$ for our method and $22 \pm 8.3\%$ for the best baseline. In practice, this failure mode is easy to rectify since a hard coded policy will always place the object in the right bin. Encouragingly, our policy is best at grasping and picking unseen objects which is the *hardest* part of this task. Nonetheless,

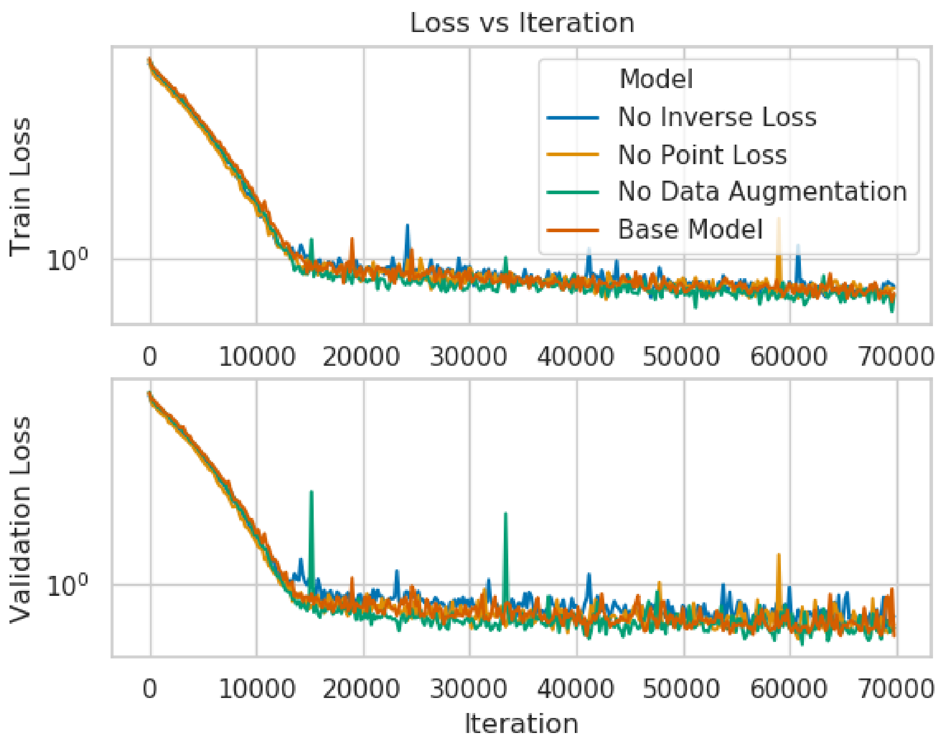


Figure H.1: One hypothesis is that the ablated models fail at test time because they cannot optimize the behavior cloning loss. Comparing train and val loss for models trained on the same data (N=1600) eliminates this possibility.

this failure mode shows more improvements are needed for this method to work in broader settings.

H.1.2 Regularization Effect on Behavior Cloning Loss

While the inverse model regularization term clearly changed test time performance for the better (shown in Section 2.4.4), can this be explained by positive transfer to the behavior cloning task? In other words, it is possible the inverse modelling loss merely prevents over-fitting in the behavior cloning loss, and thus some other regularization term could achieve the same effect.

To test this theory, we plot behavior cloning loss (both training and validation) vs train iteration for both the base model, and ablation models from Section 2.4.4. Note that behavior cloning train performance is nearly identical, whereas final success rates are dramatically different. We believe these facts in tandem confirm that self-supervised inverse modeling forces our representation to capture information which is useful for robust test performance, but not necessary to minimize the cloning loss.

H.1.3 Time-Step Ablation

Instead of using a context video from the demonstrator agent to infer the task, our model could just use the last frame from the demonstration video. After all, the last frame should uniquely specify which object should go in which bin, and prior work [70] has successfully used goal image conditioning. To test this, we train a version of our model which conditions just on the final frame from the context video, and compare its performance on the benchmarks from Section 2.4.2. This modified model achieves a final success rate of $61 \pm 9.7\%$ which is significantly less than the $88 \pm 5.0\%$ our model (which ingests more frames from context) can achieve. This effect holds even if the base model uses just one extra context frames (i.e. both beginning and end frame). We hypothesize that these frames, while not strictly necessary, help the infer which task it needs to perform, thus resulting in a performance boost.

H.2 RoboNet: Large-Scal Multi-Robot Learning

H.2.1 Action conditioned video-prediction model

Here we give a brief introduction into the visual foresight algorithm used in this chapter, see [67, 62, 60] for a more detailed treatment.

The core of visual foresight is the action conditioned video-prediction model, which is a deterministic variant of the model described in [128]. The model is illustrated in Figure H.2 and implemented as a recurrent neural network using actions $a_{0:T}$, and images $I_{0:T}$ as inputs and outputting future predicted images $\hat{I}_{1:T}$. Instead of using a context of 1 as shown in Figure H.2, a longer context can be used which increases the model's ability to adapt to environment variation such as held-out view-points. In all experiments in this chapter we used a context of 2 frames. Longer contexts can potentially help the model adapt to unseen conditions in the environment, however, this is left for future work. The RNN is unrolled according to the following equations:

$$[h_{t+1}, \hat{F}_{t+1\leftarrow t}] = g_\theta(a_t, h_t, I_t) \tag{H.1}$$

$$\hat{I}_{t+1} = \hat{F}_{t+1\leftarrow t} \diamond \hat{I}_t \tag{H.2}$$

Here $g_\theta(a_t, h_t, I_t)$ is a forward predictor parameterized by θ and $\hat{F}_{t+1\leftarrow t}$ is two-dimensional flow field with the same size as the image which is used to transform an image from one time-step into the next via bilinear-sampling.

The architecture of the RNN, which is illustrated in Figure H.3, uses a stack of convolutional LSTMs [243] interleaved with convolution layers, skip connection help the learning process.

Training details For pretraining all models are trained for 160k iterations using a batchsize of 16. For SGD we use the Adam optimizer. Finetuning is carried out for another 150k steps. The learning rate starts at 1e-3 and is annealed linearly to 0 after 200k steps until the end of training.

H.2.2 Sampling-based Planning

In visual foresight tasks are specified in terms of the motion of user-selected pixels. To predict where pixels move in response to a sequence of actions, we define a probability distribution P_0 over the locations of the designated pixel. At time step 0 this we use a one-hot-distribution with 1 a the user-selected pixel and 0 everywhere else. When then apply the same transformations to these distributions that we also apply to the images. This is summarized in the following equation:

$$\hat{P}_{t+1} = \hat{F}_{t+1\leftarrow t} \diamond \hat{P}_t \tag{H.3}$$

Here $\hat{P}_{t+1}$ denotes the predicted probability distribution of the designated pixel.

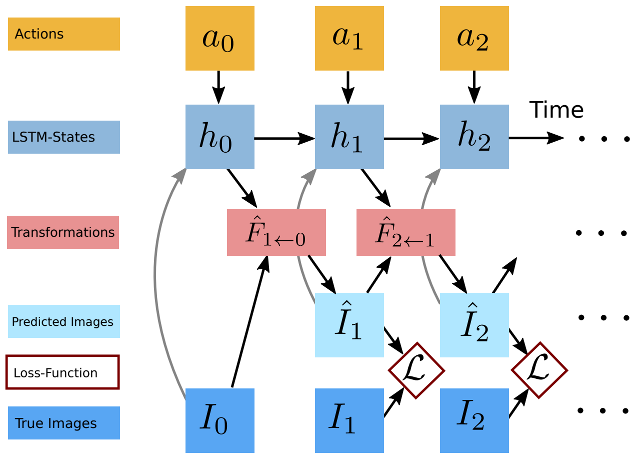


Figure H.2: Recurrent dynamics model for action-conditioned video-prediction based on flow transformations. (Used with permission from [61])

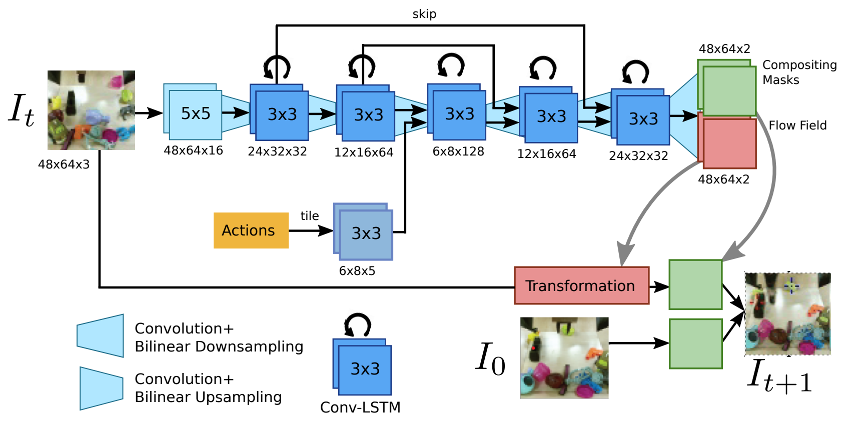


Figure H.3: Architecture of the recurrent video-prediction architecture. (Used with permission from [61])

The planning cost is computed as the expected distance to the goal pixel position d_g under the predicted distribution $\hat{P}_t$, averaged over time:

$$c = \sum_{t=1,\dots,T} c_t = \sum_{t=1,\dots,T} \mathbb{E}_{\hat{d}_t \sim P_t} \left[\|\hat{d}_t - d_g\|_2 \right] \tag{H.4}$$

To find the optimal action sequence we apply the model-predictive path intregral (MPPI) [238] algorithm, since this allows us to find good actions sequences more efficiently than random shooting. In the first iteration the actions are sampled from a unit Gaussian, in subsequent iterations the mean action is computed as an exponential weighted average as follows:

$$\mu_t = \frac{\sum_{k=0}^{N} e^{-\gamma c_k} a_{k,0:T}}{\sum_{k=0}^{N} e^{-\gamma c_k}} \tag{H.5}$$

Here N is the number of samples, chosen to be 600. The prediction horizon is 15 steps. We found that a number of 3 MPPI iterations works best in our settings. We apply temporal smoothing to the action samples using a low-pass filter to achieve smoother control and better exploration of the state space.

After finding an action sequence, the first action of this sequence is applied to the robot and the planner is queried again, thus operating in an MPC-like fashion. In order to perform re-planning, it is required to know the current position of the designated pixel. In this work we use a simple method for obtaining an estimate for the designated pixel by using the model's prediction, i.e. the flow maps from the previous time-step, we call this predictor propagation. While this position estimate is noisy and more complex alternatives, such as hand-engineered trackers or self-supervised registration [60] exist, we opt for the simple approach in this work.

H.2.3 Data Collection Details

State and Action Space Most of the robots in RoboNet (excluding Google R3 from [66]) employ the same Cartesian end-effector control action space with restricted rotation, and a gripper joint. At each time-step, the state is a $\mathbb{R}^5$ vector containing the grippers XYZ position, the gripper's yaw angle (rest of orientation is locked, with the gripper pointed downwards), and the gripper joint-angle value. The user must specify safety bounds per-robot, which constrain the end-effector to operate within a ”safe” region of space at all time-steps. Actions are specified as deltas between the current state and commanded state for the next time-step. Note that the gripper action is binarized to ”open” or ”close” for simplicity. Actions are blocking with a set time-out, so user defined policies only receive states and calculate actions once the robot has reached (or gotten as close as possible to) the commanded state. There are no ”real-time” constraints on the user policy. As a result, the robot must come to a complete stop at each step. While this scheme can easily generalize to new robots, it does impose constraints on the final robot behavior. We hope to relax these constraints in future work.

Exploration Policy During data collection, actions are either sampled from a simple diagonal Gaussian with one dimension per action-space dimension, or a more sophisticated distribution that biases the probability of grasping when the gripper is at the table height, increasing the chance that the robot will randomly grasp objects. This primitive is described further below. The variances in the diagonal Gaussians are hand-tuned per robot and differ between different action dimensions. The exact parameters are stored in inside the hdf5-files under the field `policy-description`.

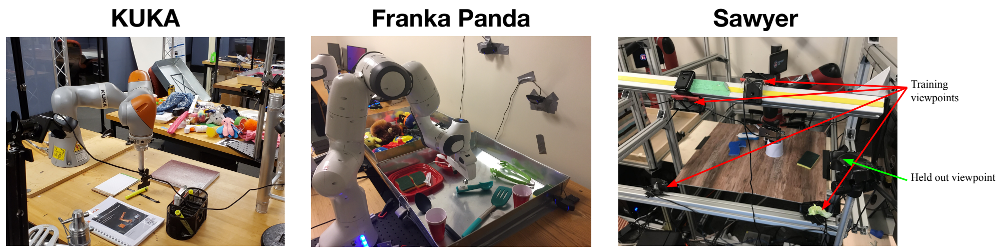


Figure H.4: Experimental setups for benchmarking tasks on the Kuka, Franka, and Sawyer robots.

While using a simple action distribution such as a diagonal Gaussian, the robot arm frequently pushes objects, however the arm quite rarely grasps objects. In order for a learning algorithm such as visual foresight to effectively model grasping, it must have seen a sufficient number of grasps in the dataset. By applying a grasping primitive, such as the one originally introduced in [60], the likelihood for these kinds of events can be increased. The grasping primitive is implemented as a hard-coded rule that closes the gripper when the z-level of the end-effector is less than a certain threshold, and opens the gripper if the arm is lifted above a threshold while not carrying an object.

There are, however, two robots in this dataset which employ significantly different exploration policies. The Google R3 robot samples random pushing motions instead of simply taking random Cartesian motions, and the Fetch robot data only contains random exploration in the x and y dimensions.

H.2.4 Database Implementation Details

The database stores every trajectory as a separate entity with a set of attributes that can be filtered. We provide code infrastructure that allows a user to filter certain subsets of attributes for training and testing. The database can be accessed using the Pandas python-API, a popular library for structuring large amounts of data. Data is stored in the widely adopted `hdf5`-format, and videos are encoded via MP4 for efficiency reasons. New trajectory attributes can be added easily.

H.2.5 Description of Benchmarking Tasks

For all control benchmarks we used object relocation tasks from a set of fixed initial positions towards a set of fixed goal positions marked on a table. The experimental setups for each robot are depicted in Figure H.4. After executing the action sequences computed by the algorithm the remaining distance to the goal is measured using a tape, and success is determined by human judges.

H.2.6 Experimental evaluation of adaptation to unseen gripper

We evaluate on a domain where a Sawyer robot is equipped with a new gripper that was not seen in the dataset. We collected 300 new trajectories with a Robotiq 2-finger gripper, which differs significantly in visual appearance and dimensions from the Weiss Robotics gripper used in all other Sawyer trajectories (see Figure 3.2), and used this data to evaluate four different models: zero-shot generalization for a model trained on RoboNet, a model trained only on the new data, a model pre-trained on only the Sawyer data in RoboNet and then finetuned with the new data, and a model pre-trained on all of RoboNet and finetuned with the new data. The results qualitative results of this evaluation are shown in Figure H.5 and the quantitative results are in Table H.1, averaging over 10 trajectories each. The best-performing model in this case is the one that is pretrained on only the Sawyer data, and it attains performance that is comparable to in-domain generalization (see, e.g., the seen viewpoint result in Table 3.2 for a comparison). The model pre-trained on the more diverse RoboNet dataset actually performs worse, likely due to the limited capacity and underfitting issues discussed in Section 3.5.4. However, these results do demonstrate that visual foresight models can adapt to moderate morphological changes using a modest amount of data.

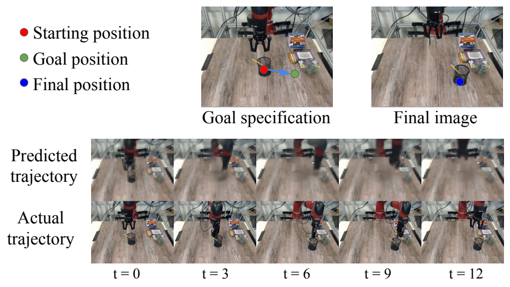


Figure H.5: Example task of pushing an object with an unseen gripper, in this case the Robotiq gripper.

	Avg. distance (cm)
zero-shot	15.5 ± 2.6
without pretraining	17 ± 1.8
pretraining on Sawyer-only	**9.8 ± 2.1**
pretraining on all of RoboNet	14.7 ± 2.1

Table H.1: Evaluation results for adaptation to Robotiq gripper with the Sawyer arm. The model trained on only Sawyer data performs the best when fine-tuned on 300 trajectories with a Robotiq gripper.

H.3 Manipulate By Seeing: Creating Manipulation Controllers from Pre-Trained Representations

H.3.1 Method Details

In this section, we present more details of our method.

Algorithm Algorithm 1 provides the psuedo-code of training our method. Algorithm 2 provides the psuedo-code of deploying our method on real robot during test time.

Data Collection Details Our Robot-Free Data Collection is built around the 19-inch RMS Handi Grip Reacher and Intel RealSense D435 camera to collect visual data. We attach a 3D printed mount above the stick to hold the camera in place. At the base of the reacher-grabber, there is a lever to control the opening and closing of the gripper fingers. To collect demonstrations, a human user uses the setup shown in Fig 4.4 (a) which allows the user to easily push, grab and interact with everyday objects in an intuitive manner. We also use an Intel RealSense T265 camera to track the end-effector position via visual inertial odometry. The demonstrations are represented as a sequence of images I_t with corresponding end-effector positions P_t.

Once we have the end-effector pose P_t for every image I_t,we extract the relative transformation $T_{t,t+1} = P_t^{-1} \times P_{t+1}$ between consecutive frames and use them as the action for training.

Training Details Our method is composed of two modules: 1) a pre-trained representation network, R, to encode observations, $i_t = R(I_t)$, and enables control via distance learning. 2) a dynamics function F, to predict the future state for a possible action a_t.

Here we encode the image I_t via a ResNet18 [97] and use a 1-layer projection head to get the visual embedding $i_t \in R^{128}$. For the dynamics function $F(i_t, a_t)$, we use a 3-layer MLP (128 + action dimension to 128 to 128) with ReLU activation. Both modules are trained jointly with $\mathcal{L} = \lambda_d \mathcal{L}_d + \lambda_F \mathcal{L}_F$ where $\lambda_d = \lambda_F = 1$. We use the Adam optimizer [117] for training the network with a batch size of 64 and a learning rate of 10^{-3}. We train the network for 500 epochs and report the performance.

For each current observation, we randomly sample 4096 actions from training set as negative logits and use the ground truth action as possitive logit. For rotation-free tasks, like pushing and stacking, we use only the translation of $T_{t,t+1}$ as the action, such that $a_t \in R^3$. For rotation-heavy tasks, like knob turning, we use both translation and rotation of $T_{t,t+1}$ as the action, such that $a_t \in R^{12}$ (first three rows in the $SE(3)$ homogeneous transformation).

```
##################Initialize###################
R: observation encoder; F: dynamics function
G: gripper action classifier
####################Input#####################
I_t: current images; I_t+1: current images;
I_g: goal images; a_t: current actions;
a_r: sampled random actions; g_t: gripper action
##############################################
# Learning Task-Centric Distances
for x in loader: # load a minibatch x with N samples
   i_t = R(x.I_t) # encode current images
   i_t+1 = R(x.I_t+1) # encode next images
   i_g = R(x.I_g) # encode goal images
   i_p = F(i_t, x.a_t) # predict next state
   i_h = F(i_t, x.a_r) # hallucinated next state
   l_pos = cosine_similarity(i_p, i_g)#positive: N*1
   l_neg = cosine_similarity(i_h, i_g)#negative: N*M
   logits = cat([l_pos, l_neg], dim=1)#logits: Nx(1+K)
   labels = zeros(N) # contrastive loss
   loss_dis = CrossEntropyLoss(logits, labels)
   loss_dyn = MSELoss(i_p, i_t+1) # dynamics loss
   loss = loss_dis + loss_dyn
   loss.backward()
   update(R.params, F.params) # Adam update
# Learning Binary Gripper Classifier
for x in loader: # load a minibatch x with N samples
   i_t = R(x.I_t) # encode current images
   g = G(i_t) # predict gripper action
   loss = BCELoss(g, x.g_t)
   loss.backward() # Adam update
   update(R.params, F.params)
```

Algorithm 1: LMLS (Train of Passive Videos)

```
##################Initialize###################
T_0: robot home position; I_0: initial observation
####################Input#####################
R: observation encoder; F: dynamics function
G: gripper action classifier; I_g: goal;
a_r: sampled random actions
##############################################
i_g = R(I_g)
While not reach_goal or t < max_step:
   i_h = F(R(I_t), a_r) # hallucinated next state
   distance = - cosine_similarity(i_h, i_g)
   # choose action leads to smallest distance-to-goal
   best_action_index = argmin(distance)
   a_t = a_r[best_action_index]
   g = G(I_t) # predict gripper action
   # Send command to robot and get new observation
   T_t+1, I_t+1 = Robot(T_t, a_t, g)
```

Algorithm 2: LMLS (Test on Robots)

To improve the performance of our networks with limited data, we use the following data augmentations in training: (a) color jittering: randomly adds up to $\pm 20\%$ random noise to the brightness, contrast and saturation of each observation. (b) gray scale: we randomly convert image to grayscale with a probability of 0.05. (c) crop: images are randomly cropped to 224×224 from an original image of size 240×240.

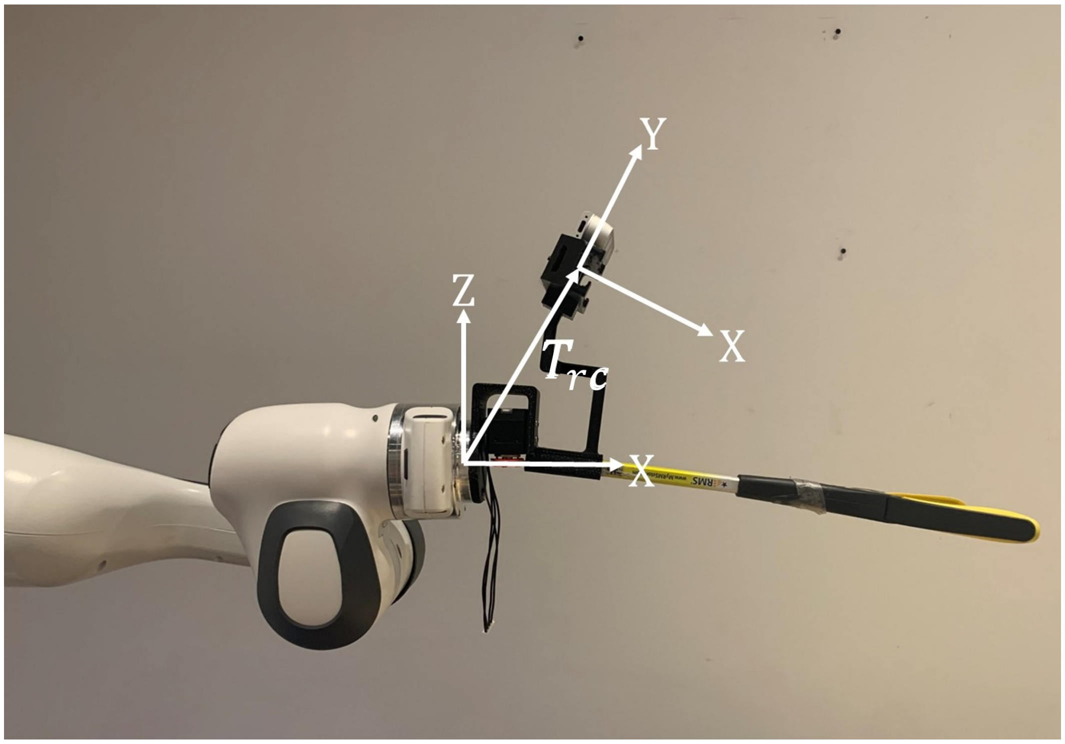


Figure H.6: Transform actions in camera frame to robot frame.

H.3.2 Experiment Details

Hardware Setup and Control Stack Our real-world experiments make use of a Franka Panda robot arm with all state logged at 50 Hz. Observations are recorded from an Intel RealSense D435 camera, using RGB-only images at 1280×720 resolution, logged at 30 Hz. On the robot's end, we use the same 19-inch RMS Handi Grip Reacher and attach it using metal studs to the robot end effector through a 3D-printed mount. To control the fingers of the tool, we remove the lever at the base of the grip reacher and replace it with a dynamixel XM430-W350-R servo motor.

The learned visual-feedback policy operates at 5 Hz. On a GTX 1080 Ti GPU. The learned action space is a 6 Dof homogeneous transformation, from the previous end-effector pose to the new one. We then calculate the new joint position using inverse kinematics through Mujoco [220]. The joint positions are linearly interpolated from their 5 Hz rate to be 100 Hz setpoints to our joint level controller. The joint positions are then sent to Facebook Polymetis [134] to control the Franka robot.

It worth noticing the learned action space is in camera frame instead of robot frame. Thus, we need to transform the predicted actions $T_{c^0c^1}$ to robot frame through a fixed homogeneous transformation T_{cr} (Fig H.6).

Using chain rule, we can easily calculate the motion in robot frame ($T_{r^0r^1}$) as:

$$T_{r^0r^1} = T_{rc} \times T_{c^0c^1} \times T_{cr} \tag{H.6}$$
$$= T_{cr}^{-1} \times T_{c^0c^1} \times T_{cr} \tag{H.7}$$

Baselines We compare our method against three SOTA baselines: behavior cloning, implicit behavior cloning, implicit Q-learning. To make the comparisons fair, we parameterize all neural networks with the same R3M representation backbone used by our method, and tune hyper-parameters for best possible performance.

- **Behavior Cloning [179, 246] (BC):** BC learns a policy (via regression) that directly predicts actions from image observations: $\min_\pi ||\pi(I_t, I_g) - a_t||_2$. This provides a strong comparison point for a whole class of LfD methods that focus on learning motor policies directly (i.e. learn policies that predict actions). Here we encode the image I_t via a ResNet18 [97] and use a 4-layer multi-layer perceptron [177] to regress the actions (512-256-128-action dimension). The predicted actions are supervised with ground-truth actions via MSELoss. We use the Adam optimizer [117] for training the network with a batch size of 64 and a learning rate of 10^{-3}. We train the network for 200 epochs and report the performance.

- **Implicit Behavior Cloning [69] (IBC):** IBC learns an energy based model that can predict actions during test time via optimization: $a_t = argmin_a E(a, I_t)$. This method is conceptually very similar to behavior cloning, but has the potential to better handle multi-modal action distributions and discontinuous actions. Similarly, we encode the image I_t via a ResNet18 [97] and use a 1-layer projection head to get the visual embedding $i_t \in R^{128}$. We also encode the actions with a 3-layer multi-layer perceptron (action dimension to 32 to 64 to 128). For each current observation, we randomly sample 4096 actions $\hat{a}_j$ from training set as negative logits and use the ground truth action a_t as possitive logit. Both visual encoder and action encoder are trained with NCE loss:

$$\mathcal{L} = \frac{exp(cos(i_t, a_t))}{exp(cos(i_t, a_t)) + \Sigma_j exp(cos(i_t, \hat{a}_j))}$$

  We use the Adam optimizer [117] for training the network with a batch size of 64 and a learning rate of 10^{-3}. We train the network for 500 epochs and report the performance.

- **Implicit Q-Learning [119] (IQL):** IQL is an offline-RL baseline that learns a Q function $Q(s, a) = Q((I_t, I_g), a_t)$, alongside a policy that maximizes it

$\pi(I_t, I_g) = argmax_a Q(s, a)$. Note that IQL's training process require us to annotate our offline trajectories $\mathcal{D}$ with a reward signal r_t for each time-step. Here we label the trajectories with sparse reward: +1 for end-effector reaching the target object, +2 for reaching the goal state, and +0 for all other states. We use d3rlpy [214] and trained the model for 500k steps.

H.4 An Unbiased Look at Datasets for Visuo-Motor Pre-Training

H.4.1 MAE Hyperparameters

We list key hyperparameters for the MAE training loop in Table H.2. Note that these parameters were employed directly from original MAE paper [94] and are actually shared by relevant robotics baselines [141, 239]. Consistent with the terminology in [94], the employed learning rate is the base learning rate scaled by (total batch size / 256). For a head-on comparison with prior work [94, 141], we train the ViT for iterations equivalent of 800 epochs over ImageNet dataset. This rigorous benchmarking took # GPUs $\times$ wall clock time $\times$ # data ablations $= 64 \times 1.5 \times 12 =$ 1152 GPU days.

H.4.2 BC Hyperparameters

The following section describes the hyperparameters used in our behavior cloning loop. As discussed in Sec. 5.3, the BC policy begins by taking in the image and passing it through the pre-trained encoder to get a representation, $E(i_t)$. That representation is then concatenated to the joint information to get a policy input, $x_t = [E(i_t), j_t]$. The policy input is fed through a 2-layer mlp network, with a batchnorm preceding the first layer, ReLU activations [121], and hidden dimensions of $[512, 512]$. Additionally, we add dropout [211] to the two mlp layers w/ probability $p = 0.2$ after the ReLU activations. The result of the top layer is then passed to 2 linear layers, that predict the mean (μ), mixing parameters (ϕ), and standard deviation (σ) of a Gaussian Mixture Model (GMM) distribution w/ m modes:

$$p(x) = \Sigma_{i=1}^{m} \phi_i N(x|\mu_i, \sigma_i)$$

The choice of GMM was based on prior work [144, 172] that showed it could dramatically improve performance. After some tuning, we used $m = 5$ on the RoboSuite tasks (note their benchmark [144] used $m = 5$) and the real world tasks, since it worked best. However, for Franka Kitchen and MetaWorld, we found no significant difference. As a result, we used $m = 1$ (i.e. standard Gaussian distribution) for those tasks to maximize comparability with prior benchmarks [141, 86].

The policy was optimized for 50000 iterations using the ADAM optimizer [117], with a learning rate of 0.0001 and a L2 weight decay of 0.0001. In addition, we applied data augmentation (random crops and random blur) to the input image i_t, before passing it E. This was based on recommendations for best practices from Hansen et. al. [92]. The full code for this setup is open-sourced on our website: https://data4robotics.github.io.

Hyperparameter	**Value**
MAE Pretraining	
optimizer	AdamW [117]
base learning rate	1e-4
weight decay	0.05
optimizer momentum	$\beta_1, \beta_2 = 0.9, 0.95$
batch size	4096
learning rate schedule	cosine decay
total batches or iterations	249600
warmup iterations	1/8 $\times$ total iterations
augmentation	RandomResizedCrop
#GPU	64 V100 (32 gb)
Wall-clock time	~36 hours
Encoder ViT Architecture	
#layers	12
#MHSA heads	12
hidden dim	768
class token	yes
positional encoding	sin cos
Decoder ViT Architecture	
#layers	8
#MHSA heads	16
hidden dim	512
class token used	yes
positional encoding	sin cos

Table H.2: Training and architectural hyperparameters for MAE pretraining.

H.4.3 Task Hyperparameters

This section describes the hyperparameters made while setting up both sim and real world tasks. All code (for robot/sim environments and BC training) is open sourced: https://data4robotics.github.io.

Simulation We evaluate on 5 tasks from *Metaworld* [250] (BinPick, ButtonPress, Hammering, Drawer Opening, and Assembly), 5 tasks from *Franka Kitchen* [86] (Knob Turning, Door Opening, Light Switch, Microwave, and Sliding Door), and 3 tasks from *RoboSuite* [269, 144] (Lift, Can, and Square). These environments are frequently used by the robot learning community, and the exact setups (e.g., camera positioning, object sets, demonstration trajectories, etc.) were directly taken from prior work [141, 154, 144]. As a result, our simulated results should be very accessible

	Task	Single Dataset Models (1M Images)					Baselines
		ImageNet	Ego4D	Kinetics	100 DoH	RoboNet	R3M [154]
Real	Block Stacking	60% ± 10%	52% ± 10%	60% ± 10%	76% ± 8.7%	56% ± 10%	4% ± 4%
	Pouring	25% ± 11%	13% ± 8.8%	22% ± 11%	25% ± 12%	6% ± 6%	19% ± 10%
	Toasting	10% ± 10%	10% ± 10%	10% ± 10%	30% ± 10%	0% ± 0%	0% ± 0%
	Average (Real)	**32**% ± 6.9%	25% ± 6.6%	**31**% ± 6.9%	**44**% ± 7.1%	21% ± 6.5%	8% ± 3.8%

Table H.3: This table analyzes if Table 5.1's conclusions apply to different pre-training schemes, or if they are limited to MAE [94]. Specifically, we apply a contrastive visual pre-training algorithm (SimCLR [35]) to 1M images from each of the target datasets. We also add an additional baseline – R3M [154] – that was trained using temporal contrastive learning on Ego4D clips. We evalaute these representations on our 3 real world tasks, and report results as success rates for each task w/ standard error (i.e., Success% ± Std. Err.%). This experiment reveals that the trends do generalize to different pre-training schemes (e.g., vision datasets still stronger than Ego4D), and that the MAE representations are stronger on average.

to the community.

The training demonstrations for these tasks were collected by previous work (CortexBench [141], Relay Policy Learning [86], RoboMimic [144] respectively). We fine-tune on $n = 25$ demos for MetaWorld/Franka Kitchen, and $n = 200$ demos on RoboSuite (again to stay consistent with older papers). Task success is measured by the environments themselves, and we get numbers by estimating success rates empirically using 50 test trajectories. Note that we only evaluate the policy at the end of training (unlike some prior work that evaluated multiple times over the course of training). This was done to ensure the sim evaluation setup matched the real world (i.e. we can't evaluate real policies multiple times during training).

Real World As discussed in Sec. 5.3, our real world tasks were built using a Franka Panda robot, and we collected 50 demonstrations for each task using a VR tele-op setup. We heavily encourage the reader to get a feel for the training data and tasks by viewing the supplemental video on our website: https://data4robotics.github.io.

The following section expands on our real world task descriptions from Sec. 5.3, and provides some additional details:

- **Block stacking** requires the robot to pick up the red block and place it on the green block. This is the simplest task, since the robot only has to adapt to new object configurations during test time, but it still requires the robot to precisely localize and grasp the (small) red block.

  We evaluated agents on this task using 25 test positions for the red/green block. These test positions were kept fixed for all policies to ensure maximum reproducibility.

- **Pouring** requires the robot to lift the cup and pour almonds in the target bowl. During test time the cup and target bowls are both novel objects (unseen during training), and are placed in random locations. Thus, this task forces the robot to generalize to new visual inputs.

  We evaluated 3 separate cup/target bowl pairs in 5 positions each (so 15 trials total). Note that none of these objects or positions were seen during test time. Again, the object and position combinations were kept fixed across every model tested.

- **Toasting** is the final task, and it requires the robot to pick up the object, place it in the toaster, and then shut the toaster. During test time, we use a novel object and randomize both the object's initial pose and the toaster's initial orientation. This is the most difficult task, since it requires the robot to execute a multi-stage manipulation strategy, while also generalizing to new visual scenarios.

  We evaluated 2 target objects pairs and randomized the toaster orientation into 5 separate poses (so 10 trials total). Note that none of these objects or toaster orientations were seen during test time. As before, all the test conditions were shared across all policies.

H.4.4 Replication with SimCLR

Our results from Sec. 5.4.1 raise questions about several key assumptions in the field. For example, we find that visuo-motor representations learned on the classic ImageNet [53] dataset are stronger than those learned on Ego4D [81] (in-the-wild data) and RoboNet [44] (random robot interactions). But are these trends fundamental to the data, or are they just a quirk of the specific pre-training algorithm/network?

To test this, we repeat the real world evaluation from Table 5.1 using the SimCLR [32] pre-training algorithm and ResNet-18 architecture [97]. As a refresher, SimCLR is a contrastive learning algorithm that optimizes a network R to "pull together" different views of the same image (i.e., two randomly augmented versions of the same image: $R(z_i), R(z_i^*)$) and "push apart" different images from each other $(R(z_i), R(z_j))$. This is accomplished with the following loss function, where $sim(x, y) = x^T y/(|x||y|)$:

$$L = -log \frac{exp(sim(R(z_i), R(z_i^*))/\tau)}{\sum_{i \neq j} exp(sim(R(z_i), R(z_j))/\tau)}$$

This SimCLR pre-training scheme is applied to each of the 1M images from our target datasets, using the same hyperparameters from the original paper [35].

We compare the newly trained representations alongside an additional ResNet-18 baseline, R3M [154], which was also trained using contrastive learning applied to Ego4D. The results for real world tasks are presented in Table H.3. Note that the trends we found in the ViT + MAE evaluations are replicated in these ResNet + SimCLR experiments: **the vision datasets – ImageNet/Kinetics/DoH – create stronger visuo-motor features w/ SimCLR compared to Ego4D/RoboNet!** We also find that despite additional tuning, which was not given to **any** other model (including trying the bigger R3M ResNet-34/50 architectures), the R3M baseline struggles heavily on our tasks (especially stacking). Finally, we note that the average performance of MAEs in Table 5.1 is stronger than the SimCLR performance (36% v.s. 31%), which further justifies our choice of setup. It is unclear if this is because of the pre-training scheme or architecture choice.

H.4.5 ImageNet Diversity Ablations

One potential hypothesis that would explain our results is that the dataset's *diversity* is critical for effective visuo-motor pre-training. This explanation is intuitive, since information compression is the basis of most self-supervised pre-training algorithms – e.g., MAEs [94] are based upon reconstructing a whole image from an encoding calculated from patches of the image. Thus, a cleaner and more diverse data distribution (like cureated ImageNet dataset) will make pre-text compression task "harder," which in turn could result in a stronger, more robust visuo-motor representation.

While this hypothesize is attractive, the main results in this chapter are not able to evaluate its veracity. Thus, we've added an additional experiment to try and shed some light on this theory. Specifically, we take two 500K subsets from ImageNet [53] that have varying levels of diversity. The first, **IN-500K-500C** consists of 500 classes each with 1000 images (500K frames total). The second, **IN-500K-1000C** uses all 1000 ImageNet classes with 500 images sampled from each (again 500K frames total). Note that these two datasets *are the same size, but the second dataset is more diverse (2x more classes)!* Thus, if diversity is critical, we should expect the 2nd dataset to perform better even though it's the same size.

Task	**IN-500K-500C**	**IN-500K-1000C**
Stacking	**70%**	**70%**
Pouring	16%	**32%**
Toasting	25%	**32%**
Average	37%	**46%**

Figure H.7: This table compares two representations trained on the same number of frames from ImageNet, but with different diversity levels (500 classes vs 1000). We find that the more diverse image set results in a marginally stronger representation.

We evaluate these two models on our real world tasks and present the success rates in Table. H.7. Note how the more diverse representation (IN-500K-1000C) performs better on the Pouring and Toasting tasks (w/ equal performance on Stacking), resulting in marginally better performance overall (46% vs 37%). In other words, **keeping all else equal a more diverse pre-training set results in a 7% performance boost!** While this result isn't fully definitive, it is an encouraging sign in favor of the diversity hypothesis.

H.5 HRP: Human Affordances for Robotic Pre-Training

H.5.1 Robot Controller Details

Franka: We use a 7-DOF Franka Emika Panda robot arm with a parallel gripper, operating in delta end-effector action space. We use a VR-based teleoperation system to collect expert demos on Franka.

xArm: We use a 6-DOF xArm robot arm with a parallel gripper, operating in absolute end-effector action space. We use an off-the-shelf hand tracking system to collect expert demos on xArm.

Dexterous Hand: We use a 6-DOF xArm robot arm with a custom dexterous hand, operating in absolute end-effector space.

For each task, the expert gets to practice for 30 to 60 mins before collecting the demonstrations. We collect 50 expert demonstrations for each of the tasks.

H.5.2 Front Cam: Full Task Performance Breakdown

Table H.4: Front Cam Performance Breakdown

Initial Representation	Method	Toasting	Pouring	Stacking	Avg. (Real)
Ego4D	Baseline	0.58	0.36	0.60	0.51
	Ours	**0.83**	**0.60**	**0.77**	**0.73**
ImageNet	Baseline	0.53	0.45	0.47	0.48
	Ours	**0.75**	**0.48**	**0.70**	**0.64**
CLIP	Baseline	0.28	0.33	0.33	0.32
	Ours	**0.50**	**0.39**	**0.57**	**0.48**
DINO	Baseline	0.38	0.32	0.40	0.37
	Ours	**0.67**	**0.57**	**0.50**	**0.58**
MVP	Baseline	0.27	0.41	0.47	0.38
	Ours	**0.73**	**0.44**	**0.63**	**0.60**
VC1	Baseline	0.52	0.33	0.57	0.47
	Ours	**0.83**	**0.34**	**0.53**	**0.57**

We observe that HRP (Ours) consistently boosts the performance across all three tasks for the front cam.

H.5.3 Ego Cam: Full Task Performance Breakdown

Table H.5: Ego Cam Performance Breakdown

Initial Representation	**Method**	**Toasting**	**Pouring**	**Stacking**	**Avg. (Real)**
Ego4D	Baseline	0.2	0.12	0.3	0.21
	Ours	0.2	**0.22**	**0.45**	**0.29**
ImageNet	Baseline	0.3	0.3	0.45	0.35
	Ours	**0.6**	**0.48**	**0.7**	**0.59**
CLIP	Baseline	0.2	0	0	0.07
	Ours	**0.275**	**0.02**	0	**0.1**
DINO	Baseline	0.35	0.32	0.3	0.32
	Ours	**0.45**	**0.7**	**0.55**	**0.57**
MVP	Baseline	0.175	0.32	0.45	0.32
	Ours	**0.3**	**0.4**	**0.65**	**0.45**
VC1	Baseline	0.5	0.28	0.4	0.39
	Ours	**0.55**	**0.6**	**0.65**	**0.6**

We also find that HRP (Ours) consistently boosts the performance across all three tasks for the ego camera.

H.5.4 Ablation Breakdown

Table H.6: Fine-Tuning Ablation Breakdown

Initial Representation	**Finetuning Scheme**	**Toasting**	**Pouring**	**Stacking**	**Avg. (Real)**
Ego4D	All Weights	**0.92**	0.51	0.77	0.73
	LayerNorm (Ours)	0.83	**0.60**	0.77	0.73
ImageNet	All Weights	**0.82**	0.34	0.63	0.60
	LayerNorm (Ours)	0.75	**0.48**	**0.70**	**0.64**
CLIP	All Weights	0.23	0.27	0.13	0.21
	LayerNorm (Ours)	**0.50**	**0.39**	**0.57**	**0.48**
DINO	All Weights	0.57	0.39	0.40	0.45
	LayerNorm (Ours)	**0.67**	**0.57**	**0.50**	**0.58**
MVP	All Weights	0.45	0.39	0.47	0.43
	LayerNorm (Ours)	**0.73**	**0.44**	**0.63**	**0.60**
VC1	All Weights	0.52	0.41	0.47	0.47
	LayerNorm (Ours)	**0.83**	**0.34**	**0.53**	**0.57**

Table H.7: Loss Ablation Performance Breakdown

Initial Representation	**Condition**	**Toasting**	**Pouring**	**Stacking**	**Avg. (Real)**
Ego4D	No Contact	0.65	0.34	0.5	0.50
	No Object	0.425	0.42	0.3	0.38
	No Hand	0.625	0.48	0.4	0.50
	Ours	**0.9**	**0.66**	**0.75**	**0.77**
Imagenet	No Contact	0.625	**0.64**	0.7	0.66
	No Object	0.525	0.52	0.55	0.53
	No Hand	0.525	0.3	**0.7**	0.51
	Ours	**0.8**	0.62	**0.7**	**0.71**
VC-1	No Contact	**0.625**	**0.48**	0.75	**0.62**
	No Object	0.225	0.38	0.65	0.42
	No Hand	0.5	0.44	0.4	0.45
	Ours	0.525	0.44	**0.8**	0.59

Note: do not compare numbers between Table H.7 and the other tables. The loss ablation experiments were run on a separate day, so all numbers were re-ran on that day. This was done to ensure a proper A/B comparison between the all methods in this table.

H.5.5 Loss Weighting Sweep

We swept through a range of weights for each of the losses to narrow down on a particular set of loss weights for HRP (presented in Table H.8). These were based relative orders of magnitude of the ground truth labels in the dataset. We empirically saw that increasing the loss weights more than 0.5 negatively affected performance and led to collapse.

Table H.8: We present the different affordance loss weights we ran sweeps on.

Exp	**Loss Weights**
HRP	$\lambda_{obj} = 0.05$, $\lambda_{ct} = 0.005$, $\lambda_{hand} = 0.5$
Drop Contact Only	$\lambda_{obj} = 0.05$, $\lambda_{ct} = 0$, $\lambda_{hand} = 0.5$
Drop Object Only	$\lambda_{obj} = 0$, $\lambda_{ct} = 0.005$, $\lambda_{hand} = 0.5$
Drop Hand Only	$\lambda_{obj} = 0.05$, $\lambda_{ct} = 0.005$, $\lambda_{hand} = 0$

H.5.6 Data Pipeline Description

To obtain human data, we first extract video clips from Ego4D [81]. Our dataset contains approximately 1200 videos. Each video is broken down semantically into smaller, by human annotators (as a part of the Ego4D). Our clips are between 1 and 30 seconds. For a given clip, we pass every frame through the 100 DOH model [197], which gives us hand object contact information. These are $\{h_l, h_r, o_l, o_r, c_l, c_r\}$. h are the hand bounding boxes, o are the object bounding boxes (which are in contact with the hand). c are contact variable (i.e. fixed, portable, self or no contact). We only look at contacts with fixed and portable. r or l represent left or right hand. Active object and hand trajectories used for our representations are directly used. For contact points, it is assumed that at the start of the clip there is no contact, from where we find the frame of first contact t. Since per-frame predictions are noisy, we run a filter [185] over the predictions. From the contact frame, we obtain the hand bounding box h and object bounding o. Contact points are computed in the intersection of h and o, and the exterior of the hand. This exterior is obtained via skin segmentation (similar to [13, 135]. These contacts can then be projected to previous frames in the clips by the homography matrix H_t obtained via SIFT features.

H.5.7 Behavior Cloning Hyper-Parameters

We list the hyper-paramaters that we used for policy training using behavior-cloning in this section. As shown in Figure 6.4, we pass an image through the learned HRP visual representation to obtain a 768 dimensional latent vector. This latent vector is passed through a two-layer MLP with (512, 512) hidden layer dimensions. To the output of the MLP we apply RELU activation along with dropout regularization with prob=0.2 to estimate the mean (μ), the mixing parameters (ϕ), and the standard deviation (σ) of a Gaussian Mixture Model (GMM) distribution with 5 modes.

We choose GMM model based on prior work [144] that showed its crucial role in increasing BC performance. We use ADAM optimizer [117] with the learning rate set to 1e-4, l2 weight decay also set to 1e-4. We train policy for 50K iterations. We also apply data augmentation (random crop and random blur) for the input images. We use the same set of hyper-parameters for both the real-world and the simulation tasks.

H.5.8 Simulation Results

Table H.9: Sim Performance

Initial Representation	**Method**	**MetaWorld Avg Performance**
Ego4D	Baseline	**0.656**
	Ours	0.580
ImageNet	Baseline	0.556
	Ours	**0.664**
CLIP	Baseline	**0.444**
	Ours	0.408
DINO	Baseline	0.660
	Ours	**0.664**
MVP	Baseline	0.592
	Ours	**0.640**
VC1	Baseline	0.576
	Ours	**0.648**

For simulation tasks, we choose 5 tasks from the *Metaworld* [250] benchmark namely: BinPick, ButtonPress, Hammering, Drawer Opening, and Assembly. This benchmark is extensively used by the robot learning community. We used the same camera viewpoint, object sets, and expert demonstrations as used by prior work [141]. We report the average performance on all 5 tasks in table H.9.

H.5.9 Evaluation

For each task, we run around 50 trials (per model), at various initial poses (for objects) and with different variations in objects. In every task, about half the trials are from the training distribution and half are from test. The differences in objects include: different colors, shapes, even semantic differences: for example in the toasting task, plush toys were tested on instead of the vegetables used to train. Cups or bowls tested, instead of mugs that were used to train the pouring task, etc.

Lighting is not controlled between train and test. We did try to run all baselines and methods as closely together as possible to avoid any confounding factors: i.e. for every trial, we ran all the baselines and our method together. Across trials, we allowed for variation in lighting conditions.

The results presented in this chapter are the average of the successes, on a scale from 0 to 1. We present the criteria for success in each task:

- **Stacking:** 1 if the robot correctly picks and stacks the red block, and 0.5 if the red block is unstably placed on the green block.
- **Pouring:** The fraction of candies, out of 5, successfully poured (e.g., 2/5 candies poured $\rightarrow$ 0.4 success).
- **Toasting:** 1 if the whole task is completed, and 0.5 successful if the robot only successfully places the object.
- **Pot on Stove:** 1 if the food is correctly placed in the pot.
- **Hand Lift Cup** 1 if the cup is stably grasped and picked.

We also compute the standard error for these trials and show that as our confidence in Tables 1-3, and as an error bar in Figures 5-7.

Bibliography

[1] Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, page 1, 2004. 33

[2] Yazan Abu Farha, Alexander Richard, and Juergen Gall. When will you do what?-anticipating temporal occurrences of activities. In *CVPR*, 2018. 61

[3] Pulkit Agrawal, Ashvin V Nair, Pieter Abbeel, Jitendra Malik, and Sergey Levine. Learning to poke by poking: Experiential learning of intuitive physics. In *Advances in Neural Information Processing Systems*, 2016. 19, 20

[4] Ferran Alet, Tomás Lozano-Pérez, and Leslie P Kaelbling. Modular meta-learning. *arXiv:1806.10166*, 2018. 20

[5] Haitham Bou Ammar, Eric Eaton, Paul Ruvolo, and Matthew Taylor. Online multi-task learning for policy gradient methods. In *International Conference on Machine Learning*, pages 1206–1214, 2014. 20

[6] 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. *arXiv:1808.00177*, 2018. 18, 20, 33

[7] Daniel Angelov, Yordan Hristov, Michael Burke, and Subramanian Ramamoorthy. Composing diverse policies for temporally extended tasks. *IEEE Robotics and Automation Letters*, 5(2):2658–2665, 2020. 33

[8] Brenna D Argall, Sonia Chernova, Manuela Veloso, and Brett Browning. A survey of robot learning from demonstration. *Robotics and autonomous systems*, 57(5):469–483, 2009. 6, 33, 49

[9] Yusuf Aytar, Tobias Pfaff, David Budden, Thomas Paine, Ziyu Wang, and Nando De Freitas. Playing hard exploration games by watching youtube. *Advances in neural information processing systems*, 31, 2018. 33

[10] Roman Bachmann, David Mizrahi, Andrei Atanov, and Amir Zamir. Multimae: Multi-modal multi-task masked autoencoders. In *ECCV*, 2022. 48

[11] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 13

[12] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. *RSS*, 2022. 33

[13] Shikhar Bahl, Russell Mendonca, Lili Chen, Unnat Jain, and Deepak Pathak. Affordances from human videos as a versatile representation for robotics. 2023. 61, 64, 99

[14] Michael Bain and Claude Sammut. A framework for behavioural cloning. *Machine Intelligence*, 1995. 6

[15] Aayush Bansal, Bryan Russell, and Abhinav Gupta. Marr revisited: 2d-3d alignment via surface normal prediction. In *CVPR*, 2016. 61

[16] Mayank Bansal, Alex Krizhevsky, and Abhijit S. Ogale. Chauffeurnet: Learning to drive by imitating the best and synthesizing the worst. *CoRR*, abs/1812.03079, 2018. 20

[17] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. BEit: BERT pre-training of image transformers. In *ICLR*, 2022. 47, 48

[18] Marc Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Remi Munos. Unifying count-based exploration and intrinsic motivation. In *Advances in Neural Information Processing Systems*, 2016. 29

[19] Homanga Bharadhwaj, Abhinav Gupta, Vikash Kumar, and Shubham Tulsiani. Towards generalizable zero-shot manipulation via translating human interaction plans. *arXiv preprint arXiv:2312.00775*, 2023. 64

[20] Raunaq Bhirangi, Tess Hellebrekers, Carmel Majidi, and Abhinav Gupta. Reskin:versatile, replaceable, lasting tactile skins. In *CoRL*, 2021. 78

[21] Aude Billard, Sylvain Calinon, Ruediger Dillmann, and Stefan Schaal. Survey: Robot programming by demonstration. *Handbook of robotics*, 59(BOOK_CHAP), 2008. 6, 33, 49

[22] Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016. 7, 47, 60

[23] Anthony Brohan, Noah Brown, Justice Carbajal, Yevgen Chebotar, Joseph Dabis, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Tomas Jackson, Sally Jesmonth, Nikhil Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Isabel Leal, Kuang-Huei Lee, Sergey Levine, Yao Lu, Utsav Malla, Deeksha Manjunath, Igor Mordatch, Ofir Nachum, Carolina Parada, Jodilyn Peralta, Emily Perez, Karl Pertsch, Jornell Quiambao, Kanishka Rao, Michael Ryoo, Grecia Salazar, Pannag Sanketi, Kevin Sayed, Jaspiar Singh, Sumedh Sontakke, Austin Stone, Clayton Tan, Huong Tran, Vincent Vanhoucke, Steve Vega, Quan Vuong, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Tianhe Yu, and Brianna Zitkovich. Rt-1: Robotics transformer for real-world control at scale. In *arXiv preprint arXiv:2212.06817*, 2022. 47, 60, 61

[24] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018. 29

[25] Kaylee Burns, Zach Witzel, Jubayer Ibn Hamid, Tianhe Yu, Chelsea Finn, and Karol Hausman. What makes pre-trained visual representations successful for robust manipulation? *ArXiv*, 2023. VII, 58, 59, 61, 66

[26] Arunkumar Byravan and Dieter Fox. Se3-nets: Learning rigid body motion using deep neural networks. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 173–180. IEEE, 2017. 20

[27] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. *CVPR*, 2021. 60, 61, 66, 69

[28] Matthew Chang, Aditya Prakash, and Saurabh Gupta. Look ma, no hands! agent-environment factorization of egocentric videos. *Advances in Neural Information Processing Systems*, 36, 2024. 64

[29] Yevgen Chebotar, Karol Hausman, Zhe Su, Artem Molchanov, Oliver Kroemer, Gaurav Sukhatme, and Stefan Schaal. Bigs: Biotac grasp stability dataset. In *ICRA 2016 Workshop on Grasping and Manipulation Datasets*, 2016. 21

[30] Yevgen Chebotar, Karol Hausman, Marvin Zhang, Gaurav Sukhatme, Stefan Schaal, and Sergey Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. In *International Conference on Machine Learning*, 2017. 20

[31] Annie S Chen, Suraj Nair, and Chelsea Finn. Learning generalizable robotic reward functions from" in-the-wild" human videos. *arXiv preprint arXiv:2103.16817*, 2021. 33

[32] Boyuan Chen, Shuran Song, Hod Lipson, and Carl Vondrick. Visual hide and seek. In *Artificial Life Conference Proceedings*. MIT Press, 2020. 95

[33] Dian Chen, Brady Zhou, Vladlen Koltun, and Philipp Krähenbühl. Learning by cheating. In *CoRL*, 2020. 46, 47, 60

[34] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Misha Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *Advances in neural information processing systems*, 34:15084–15097, 2021. 33

[35] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020. XII, 7, 46, 47, 53, 94, 95

[36] Jianpeng Cheng, Li Dong, and Mirella Lapata. Long short-term memory-networks for machine reading. *arXiv preprint arXiv:1601.06733*, 2016. 13

[37] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. In *Proceedings of Robotics: Science and Systems (RSS)*, 2023. XII, 60, 69, 73, 74, 76, 77

[38] Paul F. Christiano, Zain Shah, Igor Mordatch, Jonas Schneider, Trevor Blackwell, Joshua Tobin, Pieter Abbeel, and Wojciech Zaremba. Transfer from simulation to real world through learning deep inverse dynamics model. *CoRR*, abs/1610.03518, 2016. 20

[39] Ignasi Clavera, Anusha Nagabandi, Ronald S. Fearing, Pieter Abbeel, Sergey Levine, and Chelsea Finn. Learning to adapt: Meta-learning for model-based control. *CoRR*, abs/1803.11347, 2018. 20

[40] Open X-Embodiment Collaboration, Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, Antonin Raffin, Ayzaan Wahid, Ben Burgess-Limerick, Beomjoon Kim, Bernhard Schölkopf, Brian Ichter, Cewu Lu, Charles Xu, Chelsea Finn, Chenfeng Xu, Cheng Chi, Chenguang Huang, Christine Chan, Chuer Pan, Chuyuan Fu, Coline Devin, Danny Driess, Deepak Pathak, Dhruv Shah, Dieter Büchler, Dmitry Kalashnikov, Dorsa Sadigh, Edward Johns, Federico Ceola, Fei Xia, Freek Stulp, Gaoyue Zhou, Gaurav S. Sukhatme, Gautam Salhotra, Ge Yan, Giulio Schiavi, Hao Su, Hao-Shu Fang, Haochen Shi, Heni Ben Amor, Henrik I Christensen, Hiroki Furuta, Homer Walke, Hongjie Fang, Igor Mordatch, Ilija Radosavovic, Isabel Leal, Jacky Liang, Jaehyung Kim, Jan Schneider, Jasmine Hsu, Jeannette Bohg, Jeffrey Bingham, Jiajun Wu, Jialin Wu, Jianlan Luo, Jiayuan Gu, Jie Tan, Jihoon Oh, Jitendra Malik, Jonathan Tompson, Jonathan Yang, Joseph J. Lim, João Silvério, Junhyek Han, Kanishka Rao, Karl Pertsch, Karol Hausman, Keegan Go, Keerthana Gopalakrishnan, Ken Goldberg, Kendra Byrne, Kenneth Oslund, Kento Kawaharazuka, Kevin Zhang, Keyvan Majd, Krishan Rana, Krishnan Srinivasan, Lawrence Yunliang Chen, Lerrel Pinto, Liam Tan, Lionel Ott, Lisa Lee, Masayoshi Tomizuka, Maximilian Du, Michael Ahn, Mingtong Zhang, Mingyu Ding, Mohan Kumar Srirama, Mohit Sharma, Moo Jin Kim, Naoaki Kanazawa, Nicklas Hansen, Nicolas Heess, Nikhil J Joshi, Niko Suenderhauf, Norman Di Palo, Nur Muhammad Mahi Shafiullah, Oier Mees, Oliver Kroemer, Pannag R Sanketi, Paul Wohlhart, Peng Xu, Pierre Sermanet, Priya Sundaresan, Quan Vuong, Rafael Rafailov, Ran Tian, Ria Doshi, Roberto Martín-Martín, Russell Mendonca, Rutav Shah, Ryan Hoque, Ryan Julian, Samuel Bustamante, Sean Kirmani, Sergey Levine, Sherry Moore, Shikhar Bahl, Shivin Dass, Shuran Song, Sichun Xu, Siddhant Haldar, Simeon Adebola, Simon Guist, Soroush Nasiriany, Stefan Schaal, Stefan Welker, Stephen Tian, Sudeep Dasari, Suneel Belkhale, Takayuki Osa, Tatsuya Harada, Tatsuya Matsushima, Ted Xiao, Tianhe Yu, Tianli Ding, Todor Davchev, Tony Z. Zhao, Travis Armstrong, Trevor Darrell, Vidhi Jain, Vincent Vanhoucke, Wei Zhan, Wenxuan Zhou, Wolfram Burgard, Xi Chen, Xiaolong Wang, Xinghao Zhu, Xuanlin Li, Yao Lu, Yevgen Chebotar, Yifan Zhou, Yifeng Zhu, Ying Xu, Yixuan Wang, Yonatan Bisk, Yoonyoung Cho, Youngwoon Lee, Yuchen Cui, Yueh hua Wu, Yujin Tang, Yuke Zhu, Yunzhu Li, Yusuke Iwasawa, Yutaka Matsuo, Zhuo Xu, and Zichen Jeff Cui. Open X-Embodiment: Robotic learning datasets and RT-X models, 2024. 61, 75

[41] 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. 61

[42] Ahmad Darkhalil, Dandan Shan, Bin Zhu, Jian Ma, Amlan Kar, Richard Higgins, Sanja Fidler, David Fouhey, and Dima Damen. Epic-kitchens visor benchmark: Video segmentations and object relations. *Advances in Neural Information Processing Systems*, 35:13745–13758, 2022. 61

[43] Pradipto Das, Chenliang Xu, Richard F Doell, and Jason J Corso. A thousand frames in just a few words: Lingual description of videos through latent topics and sparse object stitching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2634–2641, 2013. 61

[44] Sudeep Dasari, Frederik Ebert, Stephen Tian, Suraj Nair, Bernadette Bucher, Karl Schmeckpeper, Siddharth Singh, Sergey Levine, and Chelsea Finn. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019. VII, 48, 50, 53, 61, 95

[45] Sudeep Dasari and Abhinav Gupta. Transformers for one-shot imitation learning. In *CoRL 2020*, 2020. 33

[46] Sudeep Dasari and Abhinav Gupta. Transformers for one-shot visual imitation. In *Conference on Robot Learning*, pages 2071–2084. PMLR, 2021. 47, 60

[47] Sudeep Dasari, Abhinav Gupta, and Vikash Kumar. Learning dexterous manipulation from exemplar object trajectories and pre-grasps. In *IEEE International Conference on Robotics and Automation 2023*, 2023. 33

[48] Sudeep Dasari, Jianren Wang, Joyce Hong, Shikhar Bahl, Yixin Lin, Austin Wang, Abitha Thankaraj, Karanbir Chahal, Berk Calli, Saurabh Gupta, et al. Rb2: Robotic manipulation benchmarking with a twist. *arXiv preprint arXiv:2203.08098*, 2022. 33

[49] Sudeep Dasari, Jianren Wang, Joyce Hong, Shikhar Bahl, Yixin Lin, Austin S Wang, Abitha Thankaraj, Karanbir Singh Chahal, Berk Calli, Saurabh Gupta, et al. Rb2: Robotic manipulation benchmarking with a twist. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021. 57, 60

[50] Marc Deisenroth and Carl E Rasmussen. Pilco: A model-based and data-efficient approach to policy search. In *International Conference on machine learning (ICML)*, 2011. 18

[51] Marc Peter Deisenroth, Peter Englert, Jan Peters, and Dieter Fox. Multi-task policy search for robotics. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3876–3881. IEEE, 2014. 20

[52] Marc Peter Deisenroth, Dieter Fox, and Carl Edward Rasmussen. Gaussian processes for data-efficient learning in robotics and control. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):408–423, 2013. 18

[53] 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. VII, 45, 46, 48, 50, 53, 59, 60, 66, 95, 96

[54] 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. 7, 18, 21

[55] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *International Conference on Robotics and Automation (ICRA)*, 2017. 20

[56] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. 2018. 1

[57] Yiming Ding, Carlos Florensa, Mariano Phielipp, and Pieter Abbeel. Goal-conditioned imitation learning. *arXiv preprint arXiv:1906.05838*, 2019. 7

[58] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016. 62

[59] Yan Duan, Marcin Andrychowicz, Bradly Stadie, OpenAI Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. One-shot imitation learning. In *Advances in neural information processing systems*, pages 1087–1098, 2017. 5, 7, 20

[60] Frederik Ebert, Sudeep Dasari, Alex X Lee, Sergey Levine, and Chelsea Finn. Robustness via retrying: Closed-loop robotic manipulation with self-supervised learning. *arXiv:1810.03043*, 2018. 23, 82, 84, 85

[61] Frederik Ebert, Chelsea Finn, Sudeep Dasari, Annie Xie, Alex Lee, and Sergey Levine. Visual foresight: Model-based deep reinforcement learning for vision-based robotic control. *arXiv:1812.00568*, 2018. V, IX, 19, 20, 21, 22, 28, 83

[62] Frederik Ebert, Chelsea Finn, Alex X Lee, and Sergey Levine. Self-supervised visual planning with temporal skip connections. *arXiv:1710.05268*, 2017. 82

[63] D Eigen and R Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. corr, abs/1411.4734. *arXiv preprint arXiv:1411.4734*, 2014. 61

[64] Christoph Feichtenhofer, Yanghao Li, Kaiming He, et al. Masked autoencoders as spatiotemporal learners. *NeurIPS*, 2022. 48

[65] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *International Conference on Machine Learning*, pages 1126–1135. PMLR, 2017. 13, 20

[66] Chelsea Finn, Ian Goodfellow, and Sergey Levine. Unsupervised learning for physical interaction through video prediction. In *Advances in neural information processing systems*, pages 64–72, 2016. V, 19, 21, 23, 24, 84

[67] Chelsea Finn and Sergey Levine. Deep visual foresight for planning robot motion. In *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2786–2793. IEEE, 2017. 19, 20, 21, 82

[68] Chelsea Finn, Tianhe Yu, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot visual imitation learning via meta-learning. In *Conference on Robot Learning*, pages 357–368, 2017. 5, 8, 20

[69] Pete Florence, Corey Lynch, Andy Zeng, Oscar Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. *Conference on Robot Learning (CoRL)*, 2021. X, 40, 41, 90

[70] Justin Fu, Avi Singh, Dibya Ghosh, Larry Yang, and Sergey Levine. Variational inverse control with events: A general framework for data-driven reward definition. *arXiv preprint arXiv:1805.11686*, 2018. 7, 81

[71] Antonino Furnari, Sebastiano Battiato, Kristen Grauman, and Giovanni Maria Farinella. Next-active-object prediction from egocentric videos. *Journal of Visual Communication and Image Representation*, 2017. 61

[72] Antonino Furnari and Giovanni Maria Farinella. Rolling-unrolling lstms for action anticipation from first-person video. *TPAMI*, 2020. 61

[73] Samir Yitzhak Gadre, Mitchell Wortsman, Gabriel Ilharco, Ludwig Schmidt, and Shuran Song. Cows on pasture: Baselines and benchmarks for language-driven zero-shot object navigation. In *CVPR*, 2023. 46

[74] Jiyang Gao, Zhenheng Yang, and Ram Nevatia. Red: Reinforced encoder-decoder networks for action anticipation. *arXiv preprint arXiv:1707.04818*, 2017. 61

[75] Ali Ghadirzadeh, Atsuto Maki, Danica Kragic, and Mårten Björkman. Deep predictive policy training using reinforcement learning. In *International Conference on Intelligent Robots and Systems (IROS)*, 2017. 20

[76] Angeliki Giannou, Shashank Rajput, and Dimitris Papailiopoulos. The expressive power of tuning only the normalization layers. *arXiv preprint arXiv:2302.07937*, 2023. 65

[77] James Jerome Gibson. *The senses considered as perceptual systems*, volume 2. 61, 64

[78] JJ Gibson. The ecological approach to visual perception. *Houghton Mifflin Comp*, 1979. 61

[79] Rohit Girdhar and Kristen Grauman. Anticipative video transformer. In *ICCV*, 2021. 61

[80] Mohit Goyal, Sahil Modi, Rishabh Goyal, and Saurabh Gupta. Human hands as probes for interactive object understanding. In *CVPR*, 2022. 61

[81] Kristen Grauman, Andrew Westbury, Eugene Byrne, Zachary Chavis, Antonino Furnari, Rohit Girdhar, Jackson Hamburger, Hao Jiang, Miao Liu, Xingyu Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 18995–19012, 2022. VII, VIII, 31, 46, 47, 48, 50, 58, 59, 61, 66, 69, 95, 99

[82] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020. 7

[83] Abhinav Gupta, Adithyavairavan Murali, Dhiraj Prakashchand Gandhi, and Lerrel Pinto. Robot learning in homes: Improving generalization and reducing dataset bias. *Advances in neural information processing systems*, 31, 2018. 20, 21, 47, 60

[84] Abhinav Gupta, Scott Satkin, Alexei A Efros, and Martial Hebert. From 3d scene geometry to human workspace. In *CVPR*, 2011. 61

[85] Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv:1703.02949*, 2017. 20

[86] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long horizon tasks via imitation and reinforcement learning. *Conference on Robot Learning (CoRL)*, 2019. 49, 51, 53, 56, 92, 93, 94

[87] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020. 60

[88] Meera Hahn, Devendra Singh Chaplot, Shubham Tulsiani, Mustafa Mukadam, James M Rehg, and Abhinav Gupta. No rl, no simulation: Learning to navigate without navigating. *Advances in Neural Information Processing Systems*, 34:26661–26673, 2021. 43

[89] Siddhant Haldar, Jyothish Pari, Anant Rai, and Lerrel Pinto. Teach a robot to fish: Versatile imitation from one minute of demonstrations. *arXiv preprint arXiv:2303.01497*, 2023. 45

[90] Ankur Handa, Arthur Allshire, Viktor Makoviychuk, Aleksei Petrenko, Ritvik Singh, Jingzhou Liu, Denys Makoviichuk, Karl Van Wyk, Alexander Zhurkevich, Balakumar Sundaralingam, Yashraj Narang, Jean-Francois Lafleche, Dieter Fox, and Gavriel State. Dextreme: Transfer of agile in-hand manipulation from simulation to reality. *arXiv*, 2022. 33

[91] Nicklas Hansen, Yu Sun, Pieter Abbeel, Alexei A Efros, Lerrel Pinto, and Xiaolong Wang. Self-supervised policy adaptation during deployment. *arXiv preprint arXiv:2007.04309*, 2020. 8

[92] Nicklas Hansen, Zhecheng Yuan, Yanjie Ze, Tongzhou Mu, Aravind Rajeswaran, Hao Su, Huazhe Xu, and Xiaolong Wang. On pre-training for visuo-motor control: Revisiting a learning-from-scratch baseline. In *International Conference on Machine Learning (ICML)*, 2023. 56, 92

[93] M Hassanin, S Khan, and M Tahtali. Visual affordance and function understanding: a survey. arxiv. *arXiv preprint arXiv:1807.06775*, 2018. 61

[94] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. VII, XII, 46, 47, 48, 49, 56, 59, 60, 61, 62, 66, 92, 94, 96

[95] 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. 7

[96] 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. 46, 47, 62

[97] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CVPR*, 2016. 8, 13, 14, 71, 87, 90, 95

[98] Todd Hester, Matej Vecerik, Olivier Pietquin, Marc Lanctot, Tom Schaul, Bilal Piot, Dan Horgan, John Quan, Andrew Sendonaris, Ian Osband, et al. Deep q-learning from demonstrations. In *AAAI*, 2018. 49

[99] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 13, 14

[100] De-An Huang and Kris M Kitani. Action-reaction: Forecasting the dynamics of human interaction. In *ECCV*, 2014. 61

[101] Po-Yao Huang, Vasu Sharma, Hu Xu, Chaitanya Ryali, Haoqi Fan, Yanghao Li, Shang-Wen Li, Gargi Ghosh, Jitendra Malik, and Christoph Feichtenhofer. Mavil: Masked audio-video learners. *arXiv preprint arXiv:2212.08071*, 2022. 48

[102] Wenlong Huang, Chen Wang, Ruohan Zhang, Yunzhu Li, Jiajun Wu, and Li Fei-Fei. Voxposer: Composable 3d value maps for robotic manipulation with language models. *arXiv preprint arXiv:2307.05973*, 2023. 64

[103] Brody Huval, Tao Wang, Sameep Tandon, Jeff Kiske, Will Song, Joel Pazhayampallil, Mykhaylo Andriluka, Pranav Rajpurkar, Toki Migimatsu, Royce Cheng-Yue, et al. An empirical evaluation of deep learning on highway driving. *arXiv:1504.01716*, 2015. 20

[104] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 10

[105] Ashesh Jain, Avi Singh, Hema S Koppula, Shane Soh, and Ashutosh Saxena. Recurrent neural networks for driver activity anticipation via sensory-fusion architecture. In *ICRA*, 2016. 61

[106] Unnat Jain, Iou-Jen Liu, Svetlana Lazebnik, Aniruddha Kembhavi, Luca Weihs, and Alexander G Schwing. Gridtopix: Training embodied agents with minimal supervision. In *ICCV*, 2021. 46

[107] Stephen James, Michael Bloesch, and Andrew J Davison. Task-embedded control networks for few-shot imitation learning. *arXiv preprint arXiv:1810.03237*, 2018. 20

[108] 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 *Computer Vision and Pattern Recognition*, 2019. 20

[109] 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 *Conference on Robot Learning*, pages 991–1002. PMLR, 2022. 33

[110] Yuanchen Ju, Kaizhe Hu, Guowei Zhang, Gu Zhang, Mingrun Jiang, and Huazhe Xu. Robo-abc: Affordance generalization beyond categories via semantic correspondence for robot manipulation. *arXiv preprint arXiv:2401.07487*, 2024. 64

[111] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021. 1

[112] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Qt-opt: Scalable deep reinforcement learning for vision-based robotic manipulation. *arXiv preprint arXiv:1806.10293*, 2018. 47, 60, 61

[113] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. *CoRR*, abs/1712.06584, 2017. 61

[114] Bingyi Kang, Zequn Jie, and Jiashi Feng. Policy optimization with demonstrations. In *ICML*. PMLR, 2018. 49

[115] Apoorv Khandelwal, Luca Weihs, Roozbeh Mottaghi, and Aniruddha Kembhavi. Simple but effective: Clip embeddings for embodied ai. *CVPR*, 2022. 46

[116] Alexander Khazatsky, Karl Pertsch, Suraj Nair, Ashwin Balakrishna, Sudeep Dasari, Siddharth Karamcheti, Soroush Nasiriany, Mohan Kumar Srirama, Lawrence Yunliang Chen, Kirsty Ellis, Peter David Fagan, Joey Hejna, Masha Itkina, Marion Lepert, Yecheng Jason Ma, Patrick Tree Miller, Jimmy Wu, Suneel Belkhale, Shivin Dass, Huy Ha, Arhan Jain, Abraham Lee, Youngwoon Lee, Marius Memmel, Sungjae Park, Ilija Radosavovic, Kaiyuan Wang, Albert Zhan, Kevin Black, Cheng Chi, Kyle Beltran Hatch, Shan Lin, Jingpei Lu, Jean Mercat, Abdul Rehman, Pannag R Sanketi, Archit Sharma, Cody Simpson, Quan Vuong, Homer Rich Walke, Blake Wulfe, Ted Xiao, Jonathan Heewon Yang, Arefeh Yavary, Tony Z. Zhao, Christopher Agia, Rohan Baijal, Mateo Guaman Castro, Daphne Chen, Qiuyu Chen, Trinity Chung, Jaimyn Drake, Ethan Paul Foster, Jensen Gao, David Antonio Herrera, Minho Heo, Kyle Hsu, Jiaheng Hu, Donovon Jackson, Charlotte Le, Yunshuang Li, Kevin Lin, Roy Lin, Zehan Ma, Abhiram Maddukuri, Suvir Mirchandani, Daniel Morton, Tony Nguyen, Abigail O'Neill, Rosario Scalise, Derick Seale, Victor Son, Stephen Tian, Emi Tran, Andrew E. Wang, Yilin Wu, Annie Xie, Jingyun Yang, Patrick Yin, Yunchu Zhang, Osbert Bastani, Glen Berseth, Jeannette Bohg, Ken Goldberg, Abhinav Gupta, Abhishek Gupta, Dinesh Jayaraman, Joseph J Lim, Jitendra Malik, Roberto Martín-Martín, Subramanian Ramamoorthy, Dorsa Sadigh, Shuran Song, Jiajun Wu, Michael C. Yip, Yuke Zhu, Thomas Kollar, Sergey Levine, and Chelsea Finn. Droid: A large-scale in-the-wild robot manipulation dataset. 2024. 75

[117] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 36, 49, 65, 67, 87, 90, 92, 93, 100

[118] Hema S Koppula and Ashutosh Saxena. Anticipating human activities using object affordances for reactive robotic response. *TPAMI*, 2015. 61

[119] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*, 2021. 32, 40, 41, 90

[120] Ilya Kostrikov, Denis Yarats, and Rob Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020. 8

[121] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90, 2017. 45, 92

[122] Ashish Kumar, Zipeng Fu, Deepak Pathak, and Jitendra Malik. Rma: Rapid motor adaptation for legged robots. *RSS*, 2021. 33

[123] Aviral Kumar, Rishabh Agarwal, Dibya Ghosh, and Sergey Levine. Implicit under-parameterization inhibits data-efficient deep reinforcement learning. *arXiv preprint arXiv:2010.14498*, 2020. 33

[124] Vikash Kumar and Emanuel Todorov. Mujoco haptix: A virtual reality system for hand manipulation. In *Humanoid Robots (Humanoids), 2015 IEEE-RAS 15th International Conference on*, pages 657–663. IEEE, 2015. 39

[125] Thanard Kurutach, Aviv Tamar, Ge Yang, Stuart J. Russell, and Pieter Abbeel. Learning plannable representations with causal infogan. *CoRR*, abs/1807.09341, 2018. 20, 29

[126] Tian Lan, Tsung-Chuan Chen, and Silvio Savarese. A hierarchical representation for future action prediction. In *ECCV*, 2014. 61

[127] Michael Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *arXiv preprint arXiv:2004.14990*, 2020. 8

[128] Alex X Lee, Richard Zhang, Frederik Ebert, Pieter Abbeel, Chelsea Finn, and Sergey Levine. Stochastic adversarial video prediction. *arXiv:1804.01523*, 2018. 21, 82

[129] Joonho Lee, Jemin Hwangbo, Lorenz Wellhausen, Vladlen Koltun, and Marco Hutter. Learning quadrupedal locomotion over challenging terrain. *Science robotics*, 5(47):eabc5986, 2020. 33

[130] Timothy E Lee, Jonathan Tremblay, Thang To, Jia Cheng, Terry Mosier, Oliver Kroemer, Dieter Fox, and Stan Birchfield. Camera-to-robot pose estimation from a single image. *arXiv preprint arXiv:1911.09231*, 2019. 7

[131] Sergey Levine, Chelsea Finn, Trevor Darrell, and Pieter Abbeel. End-to-end training of deep visuomotor policies. *The Journal of Machine Learning Research*, 17(1):1334–1373, 2016. 10, 13, 45, 47, 58, 60

[132] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 33

[133] Sergey Levine, Peter Pastor, Alex Krizhevsky, Julian Ibarz, and Deirdre Quillen. Learning hand-eye coordination for robotic grasping with deep learning and large-scale data collection. *The International journal of robotics research*, 37(4-5):421–436, 2018. 20, 47, 60, 61

[134] Yixin Lin and Austin Wang. Polymetis: a real-time pytorch controller manager. In *https://github.com/facebookresearch/polymetis*, 2021. 89

[135] Shaowei Liu, Subarna Tripathi, Somdeb Majumdar, and Xiaolong Wang. Joint hand motion and interaction hotspots prediction from egocentric videos. In *CVPR*, 2022. 61, 99

[136] Yunze Liu, Yun Liu, Che Jiang, Kangbo Lyu, Weikang Wan, Hao Shen, Boqiang Liang, Zhoujie Fu, He Wang, and Li Yi. Hoi4d: A 4d egocentric dataset for category-level human-object interaction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 21013–21022, 2022. 61

[137] Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, et al. Mediapipe: A framework for building perception pipelines. *arXiv preprint arXiv:1906.08172*, 2019. 61

[138] Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132. PMLR, 2020. 7, 10, 19, 20, 22, 27, 33

[139] Corey Lynch and Pierre Sermanet. Grounding language in play. *arXiv preprint arXiv:2005.07648*, 2020. 7

[140] Yecheng Jason Ma, Shagun Sodhani, Dinesh Jayaraman, Osbert Bastani, Vikash Kumar, and Amy Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022. 46, 47, 56, 64

[141] Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Yecheng Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Pieter Abbeel, Jitendra Malik, Dhruv Batra, Yixin Lin, Oleksandr Maksymets, Aravind Rajeswaran, and Franziska Meier. Where are we in the search for an artificial visual cortex for embodied intelligence? 2023. VII, XI, 46, 47, 48, 49, 51, 52, 56, 58, 60, 61, 64, 66, 69, 92, 93, 94, 100

[142] Priyanka Mandikal and Kristen Grauman. Dexvip: Learning dexterous grasping with human hand pose priors from video. In *Conference on Robot Learning (CoRL)*, 2021. 61

[143] Ajay Mandlekar, Fabio Ramos, Byron Boots, Li Fei-Fei, Animesh Garg, and Dieter Fox. Iris: Implicit reinforcement without interaction at scale for learning control from offline robot manipulation data. *arXiv preprint arXiv:1911.05321*, 2019. 7

[144] Ajay Mandlekar, Danfei Xu, Josiah Wong, Soroush Nasiriany, Chen Wang, Rohun Kulkarni, Li Fei-Fei, Silvio Savarese, Yuke Zhu, and Roberto Martín-Martín. What matters in learning from offline human demonstrations for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2021. XII, 49, 51, 53, 56, 67, 73, 74, 77, 92, 93, 94, 100

[145] Ajay Mandlekar, Yuke Zhu, Animesh Garg, Jonathan Booher, Max Spero, Albert Tung, Julian Gao, John Emmons, Anchit Gupta, Emre Orbay, et al. Roboturk: A crowdsourcing platform for robotic skill learning through imitation. *arXiv preprint arXiv:1811.02790*, 2018. V, 12, 21

[146] Esteve Valls Mascaro, Hyemin Ahn, and Dongheui Lee. Intention-conditioned long-term human egocentric action forecasting@ ego4d challenge 2022. *arXiv preprint arXiv:2207.12080*, 2022. 61

[147] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Wenjie Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Azade Nazi, et al. A graph placement methodology for fast chip design. *Nature*, 594(7862):207–212, 2021. 1

[148] Andriy Mnih and Koray Kavukcuoglu. Learning word embeddings efficiently with noise-contrastive estimation. *Advances in neural information processing systems*, 26, 2013. 34

[149] Katharina Mülling, Jens Kober, Oliver Kroemer, and Jan Peters. Learning to select and generalize striking movements in robot table tennis. *The International Journal of Robotics Research*, 32(3):263–279, 2013. 5

[150] Austin Myers, Ching L Teo, Cornelia Fermüller, and Yiannis Aloimonos. Affordance detection of tool parts from geometric features. In *ICRA)*, 2015. 61

[151] Tushar Nagarajan, Christoph Feichtenhofer, and Kristen Grauman. Grounded human-object interaction hotspots from video. In *ICCV*, 2019. 61

[152] Ashvin V Nair, Vitchyr Pong, Murtaza Dalal, Shikhar Bahl, Steven Lin, and Sergey Levine. Visual reinforcement learning with imagined goals. *Advances in neural information processing systems*, 31, 2018. 47, 60

[153] Suraj Nair, Mohammad Babaeizadeh, Chelsea Finn, Sergey Levine, and Vikash Kumar. Time reversal as self-supervision. *arXiv:1810.01128*, 2018. 20, 29

[154] 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. VII, XII, 31, 45, 46, 47, 49, 56, 58, 61, 63, 64, 66, 93, 94, 95

[155] Duy-Kien Nguyen, Vaibhav Aggarwal, Yanghao Li, Martin R Oswald, Alexander Kirillov, Cees GM Snoek, and Xinlei Chen. R-mae: Regions meet masked autoencoders. *arXiv preprint arXiv:2306.05411*, 2023. 48

[156] Octo Model Team, Dibya Ghosh, Homer Walke, Karl Pertsch, Kevin Black, Oier Mees, Sudeep Dasari, Joey Hejna, Charles Xu, Jianlan Luo, Tobias Kreiman, You Liang Tan, Dorsa Sadigh, Chelsea Finn, and Sergey Levine. Octo: An open-source generalist robot policy. https://octo-models.github.io, 2023. 75, 76

[157] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 34

[158] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 46, 47, 59, 62, 66

[159] Jyothish Pari, Nur Muhammad, Sridhar Pandian Arunachalam, Lerrel Pinto, et al. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021. 64

[160] Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control. *arXiv preprint arXiv:2203.03580*, 2022. 31

[161] Peter Pastor, Ludovic Righetti, Mrinal Kalakrishnan, and Stefan Schaal. Online movement adaptation based on previous sensor experiences. In *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 365–371. IEEE, 2011. 5

[162] Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. *arXiv preprint arXiv:1906.04161*, 2019. 29

[163] Deepak Pathak, Parsa Mahmoudieh, Guanghao Luo, Pulkit Agrawal, Dian Chen, Yide Shentu, Evan Shelhamer, Jitendra Malik, Alexei A Efros, and Trevor Darrell. Zero-shot visual imitation. In *Conference on Computer Vision and Pattern Recognition Workshops*, 2018. 20

[164] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. Sim-to-real transfer of robotic control with dynamics randomization. In *International Conference on Robotics and Automation (ICRA)*, 2018. 20

[165] Lerrel Pinto and Abhinav Gupta. Supersizing self-supervision: Learning to grasp from 50k tries and 700 robot hours. In *2016 IEEE international conference on robotics and automation (ICRA)*, pages 3406–3413. IEEE, 2016. 20, 47, 58, 60

[166] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. *Advances in neural information processing systems*, 1, 1988. 47, 60, 67

[167] Dean A Pomerleau. Alvinn: An autonomous land vehicle in a neural network. In *Advances in neural information processing systems*, pages 305–313, 1989. 5

[168] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. *CoRR*, abs/2103.00020, 2021. 66, 69

[169] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 1

[170] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. *CoRL*, 2022. VII, 46, 47, 58, 61, 64, 66

[171] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Aman Behal, and Ladislau Bölöni. From virtual demonstration to real-world manipulation using lstm and mdn. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018. 10

[172] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Aman Behal, and Ladislau Bölöni. From virtual demonstration to real-world manipulation using lstm and mdn. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 49, 92

[173] Rouhollah Rahmatizadeh, Pooya Abolghasemi, Ladislau Bölöni, and Sergey Levine. Vision-based multi-task manipulation for inexpensive robots using end-to-end learning from demonstration. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3758–3765. IEEE, 2018. 7, 10

[174] Aravind Rajeswaran, Vikash Kumar, Abhishek Gupta, Giulia Vezzani, John Schulman, Emanuel Todorov, and Sergey Levine. Learning complex dexterous manipulation with deep reinforcement learning and demonstrations. *arXiv preprint arXiv:1709.10087*, 2017. 33

[175] Nicholas Rhinehart and Kris M Kitani. Learning action maps of large environments via first-person vision. In *CVPR*, 2016. 61

[176] Yu Rong, Takaaki Shiratori, and Hanbyul Joo. Frankmocap: A monocular 3d whole-body pose estimation system via regression and integration. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, pages 1749–1759, October 2021. 60, 61, 63, 71

[177] Frank Rosenblatt. Principles of neurodynamics. perceptrons and the theory of brain mechanisms. Technical report, Cornell Aeronautical Lab Inc Buffalo NY, 1961. 90

[178] Stéphane Ross and Drew Bagnell. Efficient reductions for imitation learning. In *AISTATS*, 2010. 45

[179] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 627–635. JMLR Workshop and Conference Proceedings, 2011. X, 6, 31, 32, 33, 40, 41, 67, 90

[180] Constantin A Rothkopf, Dana H Ballard, and Mary M Hayhoe. Task and context determine where you look. *Journal of vision*, 7(14):16–16, 2007. 6

[181] Anirban Roy and Sinisa Todorovic. A multi-scale cnn for affordance segmentation in rgb images. In *ECCV*, 2016. 61

[182] Fereshteh Sadeghi and Sergey Levine. Cad2rl: Real single-image flight without a single real image. *arXiv:1611.04201*, 2016. 18, 20

[183] Fereshteh Sadeghi, Alexander Toshev, Eric Jang, and Sergey Levine. Sim2real viewpoint invariant visual servoing by recurrent control. In *Conference on Computer Vision and Pattern Recognition*, 2018. 20

[184] Tim Salimans, Andrej Karpathy, Xi Chen, and Diederik P Kingma. Pixelcnn++: Improving the pixelcnn with discretized logistic mixture likelihood and other modifications. *arXiv preprint arXiv:1701.05517*, 2017. 10

[185] Abraham Savitzky and Marcel JE Golay. Smoothing and differentiation of data by simplified least squares procedures. *Analytical chemistry*, 36(8), 1964. 99

[186] Johann Sawatzky, Abhilash Srikantha, and Juergen Gall. Weakly supervised affordance detection. In *CVPR*, 2017. 61

[187] Stefan Schaal. Is imitation learning the route to humanoid robots? *Trends in cognitive sciences*, 3(6):233–242, 1999. 6, 33, 47, 49, 67

[188] Stefan Schaal et al. Learning from demonstration. *Advances in neural information processing systems*, pages 1040–1046, 1997. 59, 75

[189] Karl Schmeckpeper, Oleh Rybkin, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Reinforcement learning with videos: Combining offline observations with interaction. In *Conference on Robot Learning*, pages 339–354. PMLR, 2021. 33

[190] Johannes Lutz Schönberger and Jan-Michael Frahm. Structure-from-motion revisited. In *Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. 39

[191] Julian Schrittwieser, Ioannis Antonoglou, Thomas Hubert, Karen Simonyan, Laurent Sifre, Simon Schmitt, Arthur Guez, Edward Lockhart, Demis Hassabis, Thore Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020. 1

[192] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1134–1141. IEEE, 2018. 8, 33

[193] Pierre Sermanet, Corey Lynch, Yevgen Chebotar, Jasmine Hsu, Eric Jang, Stefan Schaal, Sergey Levine, and Google Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018. 59

[194] Pierre Sermanet, Kelvin Xu, and Sergey Levine. Unsupervised perceptual rewards for imitation learning. *arXiv preprint arXiv:1612.06699*, 2016. 8, 33

[195] Nur Muhammad Mahi Shafiullah, Zichen Jeff Cui, Ariuntuya Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. In *Thirty-Sixth Conference on Neural Information Processing Systems*, 2022. 33

[196] Rutav M Shah and Vikash Kumar. Rrl: Resnet as representation for reinforcement learning. In *ICML*, 2021. 64

[197] Dandan Shan, Jiaqi Geng, Michelle Shu, and David Fouhey. Understanding human hands in contact at internet scale. In *CVPR*, 2020. VII, VIII, XI, 46, 48, 50, 59, 60, 61, 63, 66, 69, 71, 99

[198] 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)*, July 2020. 7

[199] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. *arXiv:1810.07121*, 2018. 21

[200] Pratyusha Sharma, Deepak Pathak, and Abhinav Gupta. Third-person visual imitation learning via decoupled hierarchical controller. *arXiv preprint arXiv:1911.09676*, 2019. 5

[201] Kenneth Shaw, Ananye Agarwal, and Deepak Pathak. Leap hand:low-cost, efficient, and anthropomorphic hand for robot learning. *RSS*, 2023. XI, 71

[202] Kenneth Shaw, Shikhar Bahl, and Deepak Pathak. Videodex: Learning dexterity from internet videos. In *CoRL*, 2022. 61

[203] Mohit Shridhar, Lucas Manuelli, and Dieter Fox. Cliport: What and where pathways for robotic manipulation. In *CoRL*, 2022. 64

[204] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. Mastering the game of go without human knowledge. *nature*, 550(7676):354–359, 2017. 1

[205] Karen Simonyan and Andrew Zisserman. Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576, 2014. 7

[206] Mannat Singh, Quentin Duval, Kalyan Vasudev Alwala, Haoqi Fan, Vaibhav Aggarwal, Aaron Adcock, Armand Joulin, Piotr Dollár, Christoph Feichtenhofer, Ross Girshick, et al. The effectiveness of mae pre-pretraining for billion-scale pretraining. *arXiv preprint arXiv:2303.13496*, 2023. 48

[207] Lucas Smaira, João Carreira, Eric Noland, Ellen Clancy, Amy Wu, and Andrew Zisserman. A short note on the kinetics-700-2020 human action dataset. *arXiv preprint arXiv:2010.10864*, 2020. VII, 46, 48, 50

[208] Laura Smith, Nikita Dhawan, Marvin Zhang, Pieter Abbeel, and Sergey Levine. Avid: Learning multi-stage tasks via pixel-level translation of human videos. *arXiv preprint arXiv:1912.04443*, 2019. 5

[209] Shuran Song, Andy Zeng, Johnny Lee, and Thomas Funkhouser. Grasping in the wild: Learning 6dof closed-loop grasping from low-cost demonstrations. *Robotics and Automation Letters*, 2020. 39, 60

[210] Aravind Srinivas, Michael Laskin, and Pieter Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv preprint arXiv:2004.04136*, 2020. 8

[211] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958, 2014. VII, 10, 46, 55, 56, 92

[212] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014. 13

[213] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018. 31, 33, 45, 47, 58

[214] Michita Imai Takuma Seno. d3rlpy: An offline deep reinforcement library. In *NeurIPS 2021 Offline Reinforcement Learning Workshop*, December 2021. 91

[215] Matthew Tancik, Pratul P Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan T Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *arXiv preprint arXiv:2006.10739*, 2020. 9

[216] Sebastian Thrun. A lifelong learning perspective for mobile robot control. In *Intelligent Robots and Systems*, 1995. 20

[217] Sebastian Thrun and Tom M Mitchell. Lifelong robot learning. *Robotics and autonomous systems*, 1995. 20

[218] Stephen Tian, Frederik Ebert, Dinesh Jayaraman, Mayur Mudigonda, Chelsea Finn, Roberto Calandra, and Sergey Levine. Manipulation by feel: Touch-based control with deep predictive models. *arXiv:1903.04128*, 2019. 20, 29

[219] 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 *International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 2017. 20

[220] 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. 11, 47, 49, 89

[221] Zhan Tong, Yibing Song, Jue Wang, and Limin Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *NeurIPS*, 2022. 48

[222] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. *arXiv preprint arXiv:1805.01954*, 2018. 7

[223] Faraz Torabi, Garrett Warnell, and Peter Stone. Generative adversarial imitation from observation. *arXiv preprint arXiv:1807.06158*, 2018. 33

[224] Antonio Torralba, Rob Fergus, and William T Freeman. 80 million tiny images: A large data set for nonparametric object and scene recognition. *transactions on pattern analysis and machine intelligence*, 2008. 21

[225] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017. 6, 8, 9

[226] Ruben Villegas, Arkanath Pathak, Harini Kannan, Dumitru Erhan, Quoc V Le, and Honglak Lee. High fidelity video prediction with large neural nets. 27, 29

[227] Ruben Villegas, Jimei Yang, Seunghoon Hong, Xunyu Lin, and Honglak Lee. Decomposing motion and content for natural video sequence prediction. *arXiv preprint arXiv:1706.08033*, 2017. 61

[228] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. 1

[229] Stefan Vogt and Roland Thomaschke. From visuo-motor interactions to imitation learning: behavioural and brain imaging studies. *Journal of Sports Sciences*, 25(5):497–517, 2007. 5

[230] Carl Vondrick, Deniz Oktay, Hamed Pirsiavash, and Antonio Torralba. Predicting motivations of actions by leveraging text. In *CVPR*, 2016. 61

[231] Jiayi Wang, Franziska Mueller, Florian Bernard, Suzanne Sorli, Oleksandr Sotnychenko, Neng Qian, Miguel A Otaduy, Dan Casas, and Christian Theobalt. Rgb2hands: real-time tracking of 3d hand interactions from monocular rgb video. *ACM Transactions on Graphics (TOG)*, 39(6):1–16, 2020. 61

[232] Limin Wang, Bingkun Huang, Zhiyu Zhao, Zhan Tong, Yinan He, Yi Wang, Yali Wang, and Yu Qiao. Videomae v2: Scaling video masked autoencoders with dual masking. In *CVPR*, 2023. 48

[233] Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7794–7803, 2018. 6, 9

[234] Manuel Watter, Jost Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *neural information processing systems*, 2015. 20

[235] Luca Weihs, Unnat Jain, Iou-Jen Liu, Jordi Salvador, Svetlana Lazebnik, Aniruddha Kembhavi, and Alex Schwing. Bridging the imitation gap by adaptive insubordination. *NeurIPS*, 2021. 49

[236] Dirk Weissenborn, Oscar Täckström, and Jakob Uszkoreit. Scaling autoregressive video models. *arXiv preprint arXiv:1906.02634*, 2019. 29

[237] William Whitney, Rajat Agarwal, Kyunghyun Cho, and Abhinav Gupta. Dynamics-aware embeddings. *arXiv preprint arXiv:1908.09357*, 2019. 47, 60

[238] Grady Williams, Andrew Aldrich, and Evangelos Theodorou. Model predictive path integral control using covariance variable importance sampling. *CoRR*, abs/1509.01149, 2015. 83

[239] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022. XI, 31, 46, 47, 48, 49, 51, 52, 56, 69, 92

[240] Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *arXiv preprint arXiv:1904.05538*, 2019. 7

[241] Annie Xie, Frederik Ebert, Sergey Levine, and Chelsea Finn. Improvisation through physical understanding: Using novel objects as tools with visual foresight. *CoRR*, abs/1904.05538, 2019. 21, 29

[242] Annie Xie, Avi Singh, Sergey Levine, and Chelsea Finn. Few-shot goal inference for visuomotor learning and planning. In *Conference on Robot Learning*, pages 40–52. PMLR, 2018. 33

[243] SHI Xingjian, Zhourong Chen, Hao Wang, Dit-Yan Yeung, Wai-Kin Wong, and Wang-chun Woo. Convolutional lstm network: A machine learning approach for precipitation nowcasting. In *Advances in neural information processing systems*, pages 802–810, 2015. 82

[244] Chao Yang, Xiaojian Ma, Wenbing Huang, Fuchun Sun, Huaping Liu, Junzhou Huang, and Chuang Gan. Imitation learning from observations by minimizing inverse dynamics disagreement. In *Advances in Neural Information Processing Systems*, pages 239–249, 2019. 7

[245] Yufei Ye, Xueting Li, Abhinav Gupta, Shalini De Mello, Stan Birchfield, Jiaming Song, Shubham Tulsiani, and Sifei Liu. Affordance diffusion: Synthesizing hand-object interactions. In *CVPR*, 2023. 61

[246] Sarah Young, Dhiraj Gandhi, Shubham Tulsiani, Abhinav Gupta, Pieter Abbeel, and Lerrel Pinto. Visual imitation made easy. In *Conference on Robot Learning*, pages 1992–2005. PMLR, 2021. X, 39, 40, 41, 90

[247] Kuan-Ting Yu, Maria Bauza, Nima Fazeli, and Alberto Rodriguez. More than a million ways to be pushed. a high-fidelity experimental dataset of planar pushing. In *International Conference on Intelligent Robots and Systems (IROS)*, 2016. 21

[248] Lantao Yu, Tianhe Yu, Chelsea Finn, and Stefano Ermon. Meta-inverse reinforcement learning with probabilistic context variables. *Advances in Neural Information Processing Systems*, 32:11772–11783, 2019. 33

[249] Tianhe Yu, Chelsea Finn, Annie Xie, Sudeep Dasari, Tianhao Zhang, Pieter Abbeel, and Sergey Levine. One-shot imitation from observing humans via domain-adaptive meta-learning. *arXiv preprint arXiv:1802.01557*, 2018. 6, 7, 8, 13, 14

[250] Tianhe Yu, Deirdre Quillen, Zhanpeng He, R. Julian, Karol Hausman, Chelsea Finn, and S. Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *CoRL*, 2019. 49, 51, 53, 93, 100

[251] Tianhe Yu, Gleb Shevchuk, Dorsa Sadigh, and Chelsea Finn. Unsupervised visuomotor control through distributional planning networks. *arXiv:1902.05542*, 2019. V, 19, 20, 23, 24

[252] Wenhao Yu, C. Karen Liu, and Greg Turk. Preparing for the unknown: Learning a universal policy with online system identification. *CoRR*, abs/1702.02453, 2017. 20

[253] Andy Zeng, Pete Florence, Jonathan Tompson, Stefan Welker, Jonathan Chien, Maria Attarian, Travis Armstrong, Ivan Krasin, Dan Duong, Vikas Sindhwani, and Johnny Lee. Transporter networks: Rearranging the visual world for robotic manipulation. *CoRL*, 2020. 64

[254] Andy Zeng, Shuran Song, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Tossingbot: Learning to throw arbitrary objects with residual physics. *arXiv:1903.11239*, 2019. 20

[255] Andy Zeng, Shuran Song, Stefan Welker, Johnny Lee, Alberto Rodriguez, and Thomas Funkhouser. Learning synergies between pushing and grasping with self-supervised deep reinforcement learning. *arXiv:1803.09956*, 2018. 20

[256] Xiaohua Zhai, Alexander Kolesnikov, Neil Houlsby, and Lucas Beyer. Scaling vision transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12104–12113, 2022. 48, 62

[257] Ruohan Zhang, Zhuode Liu, Luxin Zhang, Jake A Whritner, Karl S Muller, Mary M Hayhoe, and Dana H Ballard. Agil: Learning attention from human for visuomotor tasks. In *Proceedings of the european conference on computer vision (eccv)*, pages 663–679, 2018. 6

[258] Tianhao Zhang, Zoe McCarthy, Owen Jow, Dennis Lee, Xi Chen, Ken Goldberg, and Pieter Abbeel. Deep imitation learning for complex manipulation tasks from virtual reality teleoperation. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE, 2018. 5, 39

[259] Bingchen Zhao, Haoqin Tu, Chen Wei, Jieru Mei, and Cihang Xie. Tuning layernorm in attention: Towards efficient multi-modal llm finetuning. *arXiv preprint arXiv:2312.11420*, 2023. 65

[260] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *Proceedings of Robotics: Science and Systems (RSS)*, 2023. 76, 77

[261] Yibiao Zhao and Song-Chun Zhu. Scene parsing by integrating function, geometry and appearance models. In *CVPR*, 2013. 61

[262] Allan Zhou, Eric Jang, Daniel Kappler, Alex Herzog, Mohi Khansari, Paul Wohlhart, Yunfei Bai, Mrinal Kalakrishnan, Sergey Levine, and Chelsea Finn. Watch, try, learn: Meta-learning from demonstrations and reward. *arXiv preprint arXiv:1906.03352*, 2019. 14

[263] Gaoyue Zhou, Victoria Dean, Mohan Kumar Srirama, Aravind Rajeswaran, Jyothish Pari, Kyle Hatch, Aryan Jain, Tianhe Yu, Pieter Abbeel, Lerrel Pinto, et al. Train offline, test online: A real robot learning benchmark. *arXiv preprint arXiv:2306.00942*, 2023. 57

[264] Huiyu Zhou, Yuan Yuan, and Chunmei Shi. Object tracking using sift features and mean shift. *Computer vision and image understanding*, 113(3):345–352, 2009. 63

[265] Wenxuan Zhou, Bowen Jiang, Fan Yang, Chris Paxton, and David Held. Hacman: Learning hybrid actor-critic maps for 6d non-prehensile manipulation. In *7th Annual Conference on Robot Learning*, 2023. 64

[266] Xingyi Zhou, Rohit Girdhar, Armand Joulin, Phillip Krähenbühl, and Ishan Misra. Detecting twenty-thousand classes using image-level supervision. *arXiv preprint arXiv:2201.02605*, 2022. XII, 72

[267] Yixin Zhu, Chenfanfu Jiang, Yibiao Zhao, Demetri Terzopoulos, and Song-Chun Zhu. Inferring forces and learning human utilities from videos. In *CVPR*, 2016. 61

[268] Yuke Zhu, Alireza Fathi, and Li Fei-Fei. Reasoning about object affordances in a knowledge base representation. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part II 13*, pages 408–424. Springer, 2014. 61

[269] 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. In *arXiv preprint arXiv:2009.12293*, 2020. 93