DynaMo: In-Domain Dynamics Pretraining for Visuo-Motor Control

Zichen Jeff Cui* **Hengkai Pan** **Aadhithya Iyer** **Siddhant Haldar** **Lerrel Pinto**

New York University

Abstract

Imitation learning has proven to be a powerful tool for training complex visuo-motor policies. However, current methods often require hundreds to thousands of expert demonstrations to handle high-dimensional visual observations. A key reason for this poor data efficiency is that visual representations are predominantly either pretrained on out-of-domain data or trained directly through a behavior cloning objective. In this work, we present DynaMo, a new in-domain, self-supervised method for learning visual representations. Given a set of expert demonstrations, we jointly learn a latent inverse dynamics model and a forward dynamics model over a sequence of image embeddings, predicting the next frame in latent space, without augmentations, contrastive sampling, or access to ground truth actions. Importantly, DynaMo does not require any out-of-domain data such as Internet datasets or cross-embodied datasets. On a suite of six simulated and real environments, we show that representations learned with DynaMo significantly improve downstream imitation learning performance over prior self-supervised learning objectives, and pretrained representations. Gains from using DynaMo hold across policy classes such as Behavior Transformer, Diffusion Policy, MLP, and nearest neighbors. Finally, we ablate over key components of DynaMo and measure its impact on downstream policy performance. Robot videos are best viewed at https://dynamo-ssl.github.io.

1 Introduction

Learning visuo-motor policies from human demonstrations is an exciting approach for training difficult control tasks in the real world [1–5]. However, a key challenge in such a learning paradigm is to efficiently learn a policy with fewer expert demonstrations. To address this, prior works have focused on learning better visual representations, often by pretraining on large Internet-scale video datasets [6–11]. However, as shown in Dasari et al. [12], these out-of-domain representations may not transfer to downstream tasks with very different embodiments and viewpoints from the pretraining dataset.

An alternative to using Internet-pretrained models is to train the visual representations 'in-domain' on the demonstration data collected to solve the task [13, 4]. However, in-domain datasets are much smaller than Internet-scale data. This has resulted in the use of domain-specific augmentations [13] to induce representational invariances with self-supervision or to collect larger amounts of demonstrations [2, 14]. The reliance of existing methods on large datasets might suggest that in-domain self-supervised pretraining is ineffective for visuo-motor control, and we might be better with simply training end-to-end. In this work, we argue the contrary – in-domain self-supervision can be effective with a better training objective that extracts more information from small datasets.

*Corresponding author. Email: jeff.cui@nyu.edu

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

 https://doi.org/10.52202/079017-1069

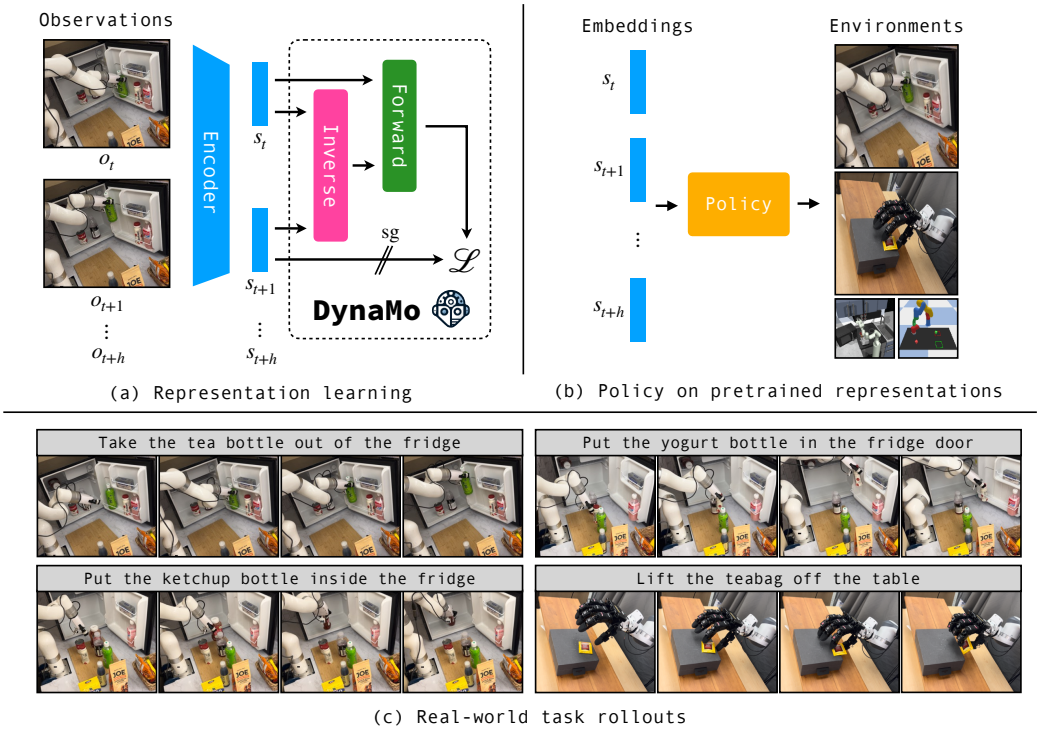


Figure 1: (a) We present DynaMo, a new self-supervised method for learning visual representations for visuomotor control. DynaMo exploits the causal structure in demonstrations by jointly learning the encoder with inverse and forward dynamics models. DynaMo requires no augmentations, contrastive sampling, or access to ground truth actions. This enables downstream policy learning using limited in-domain data across simulated and real-world robotics tasks. For each environment, we pretrain the visual representation in-domain with DynaMo and learn a policy on the pretrained embeddings. (b) We provide real-world rollouts of policies learned with DynaMo representation on our multi-task xArm Kitchen and Allegro Manipulation environments.

Prevalent approaches for using self-supervision in downstream control often make a bag-of-frames assumption, using contrastive methods [15, 16] or masked autoencoding [11, 8] on individual frames for self-supervision. Most of these approaches ignore a rich supervision signal: action-based causality. Future observations are dependent on past observations, and unobserved latent actions. Can we obtain a good visual representation for control by simply learning the dynamics? In fact, this idea is well-established in neuroscience: animals are thought to possess internal models of the motor apparatus and the environment that facilitate motor control and planning [17–24].

In this work, we present **Dyna**mics Pretraining for Visuo-**Mo**tor Control (**DynaMo**), a new self-supervised method for pretraining visual representations for visuomotor control from limited in-domain data. DynaMo jointly learns the encoder with inverse and forward dynamics models, without access to ground truth actions [25, 26].

To demonstrate the effectiveness of DynaMo, we evaluate our representation on four simulation suites - Franka Kitchen [27], Block Pushing [28], Push-T [3], and LIBERO [29], as well as eight robotic manipulation tasks on two real-world environments. Our main findings are summarized below:

1. DynaMo exhibits an overall 39% improvement in downstream policy performance over prior state-of-the-art pretrained and self-supervised representations, especially on the harder closed-loop control tasks in Block Pushing and Push-T (Table 1), and on real robot experiments (Table 2).
2. DynaMo is compatible with various policy classes, can be used to fine-tune pretrained weights, and works in the low-data regime with limited demonstrations on a real-world Allegro hand (Tables 4, 5, and 2 respectively).
3. Through an ablation analysis, we study the impact of each component in DynaMo on downstream policy performance (§4.6).

https://doi.org/10.52202/079017-1069

All of our datasets, and training and evaluation code will be made publicly available. Videos of our trained policies can be seen here: https://dynamo-ssl.github.io.

2 Background

2.1 Visual imitation learning

Our work follows the general framework for visual imitation learning. Given demonstration data $\mathcal{D} = \{(o_t, a_t)\}_t$, where o_t are raw visual observations and a_t are the corresponding ground-truth actions, we first employ a visual encoder $f_\theta : o_t \rightarrow s_t$ to map the raw visual inputs to lower-dimensional embeddings s_t. We then learn a policy $\pi(a_t|s_t)$ to predict the appropriate actions. For rollouts, we model the environment as a Markov Decision Process (MDP), where each subsequent observation o_{t+1} depends on the previous observation-action pair (o_t, a_t). We assume the action-conditioned transition distribution $p(o_{t+1}|o_t, a_t)$ to be unimodal for our manipulation tasks.

2.2 Visual pretraining for policy learning

Our goal is to pretrain the visual encoder f_θ using a dataset of sequential raw visual observations $\mathcal{D} = \{o_t\}_t$ to support downstream policy learning. During pretraining, we do not assume access to the ground-truth actions $\{a_t\}_t$.

Prior work has shown that pretraining encoders on large out-of-domain datasets can improve downstream policy performance [6–11]. However, such pretraining may not transfer well to tasks with different robot embodiments [12].

Alternatively, we can directly pretrain the encoder in-domain using self-supervised methods. One approach is contrastive learning with data augmentation priors, randomly augmenting an image twice and pushing their embeddings closer. Another approach is denoising methods, predicting the original image from a noise-degraded sample (e.g. by masking [11, 8, 30]). A third approach is contrastive learning with temporal proximity as supervision, pushing temporally close frames to have similar embeddings [31, 32].

3 DynaMo

Limitations of prior self-supervised techniques: Prior self-supervised techniques can learn to fixate on visually salient features and ignore fine-grained features important for control. We illustrate this limitation using the Block Pushing environment from Florence et al. [28]. In this task, the goal is to push a block into a target square. While the robot arm occupies much of the raw pixel space, the blocks are central to the task despite being smaller in the visual field. Figure 2 visualizes a random frame from the demonstration data and its 20 nearest neighbors in the embedding space learned by several self-supervised techniques.

We observe that prior self-supervised methods (details in §4.2) focus on the visually dominant robot, matching the whole robot arm extremely accurately. However, they fail to capture the block positions, which are important to the task despite being much less salient visually.

Can we learn a visual encoder that extracts task-specific features better? We know that the demonstrations are sequential: each observation is dependent on the previous observation, and an action (unobserved in this setting). Prior self-supervised methods ignore this sequential structure. Contrastive augmentations [16, 33] and autoencoding objectives [30, 8, 11] assume that the demonstration video is a bag of frames, discarding temporal information altogether. Temporal contrast [32, 31] uses temporal proximity but discards the sequential information in the observations: the contrastive objectives are usually symmetric in time, disregarding past/future order.

Instead of a contrastive or denoising objective, we propose a dynamics prediction objective that explicitly exploits the sequential structure of demonstration observations.

Overview of DynaMo: The key insight of our method is that we can learn a good visual representation for control by modeling the dynamics on demonstration observations, without requiring augmentations, contrastive sampling, or access to the ground truth actions. Given a sequence of raw

https://doi.org/10.52202/079017-1069

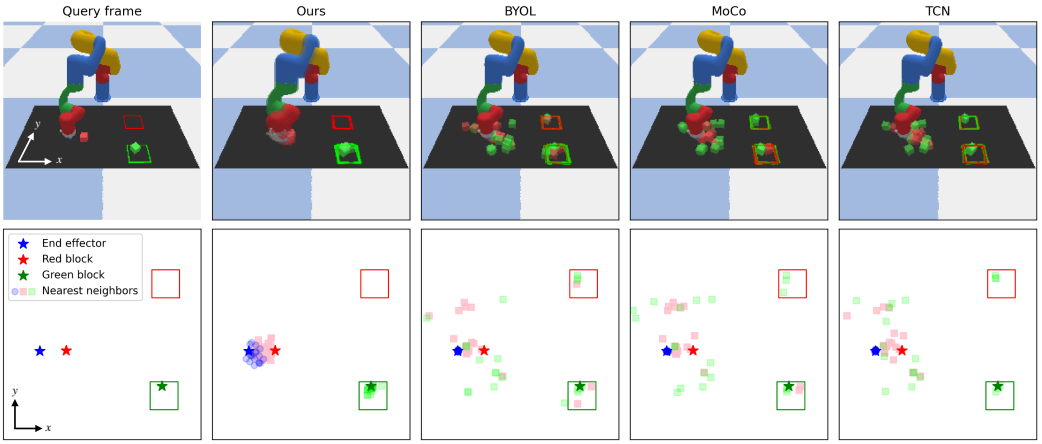


Figure 2: Embedding nearest neighbor matches for DynaMo, BYOL, MoCo, and TCN on the Block Pushing environment. **(Top)** The nearest neighbor matches visualized in pixel space. **(Bottom)** Matches visualized in a top-down view. We see that the DynaMo representation captures task-relevant features (end effector, block, and target locations in this case), whereas prior work fixates on the large robot arm.

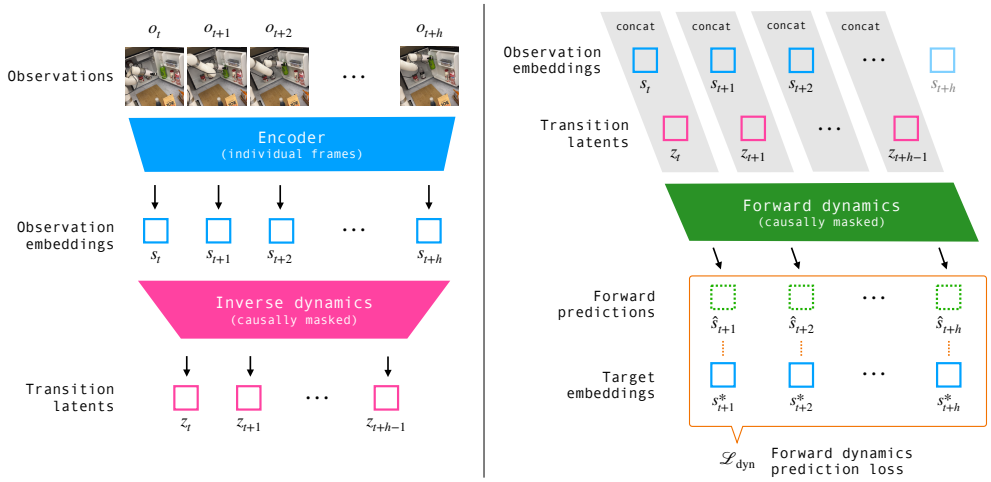


Figure 3: Architecture of DynaMo. DynaMo jointly learns an image encoder, an inverse dynamics model, and a forward dynamics model with a forward dynamics prediction loss.

visual observations $(o_1, \ldots, o_T)$, we jointly train the encoder $f_\theta : o_t \to s_t$, a latent inverse dynamics model $q(z_{t:t+h-1}|s_{t:t+h})$, and a forward dynamics model $p(\hat{s}_{t+1:t+h}|s_{t:t+h-1}, z_{t:t+h-1})$. We model the actions as unobserved latents, and train all models end-to-end with a consistency loss on the forward dynamics prediction. For our experiments, we use a ResNet18 [34] encoder, and causally masked transformer encoders [35] for the inverse and forward dynamics models. The architecture is illustrated in Figure 3.

3.1 Dynamics as a visual self-supervised learning objective

First, we sample an observation sequence $o_{t:t+h}$ of length h and compute its representation $s_{t:t+h} = f_\theta(o_{t:t+h})$. For convenience, we will write $s_{t:t+h}$ as $s_{:h}$, and $s_{t+1:t+h}$ as $s_{1:h}$ below. At any given step, the distribution of possible actions can be multimodal [5]. Therefore, the forward dynamics transition $p(s_{1:h}|s_{:h-1})$ can also have multiple modes. To address this, we first model the inverse dynamics $q(z_{:h-1}|s_{:h})$, where z_t is the latent transition between frames. We assume z_t to be well-determined and unimodal given consecutive frames $\{s_t, s_{t+1}\}$. We have $z \in \mathbb{R}^m, s \in \mathbb{R}^d, m \ll d$ such that the latent cannot trivially memorize the next frame embedding. Finally, we concatenate (s_t, z_t) and predict the one-step forward dynamics $p(\hat{s}_{1:h}|s_{:h-1}, z_{:h-1})$.

https://doi.org/10.52202/079017-1069

We compute a dynamics loss $\mathcal{L}_{\text{dyn}}(\hat{s}, s^*)$ on the one-step forward predictions $\hat{s}_{t+1:t+h}$, where $s^*_{t+1:t+h}$ are the target next-frame embeddings; and a covariance regularization loss $\mathcal{L}_{\text{cov}}$ from Bardes et al. [36] on a minibatch of observation embeddings S:

$$\begin{aligned} \mathcal{L}_{\text{dyn}}(\hat{s}_t, s^*_t) &= 1 - \frac{\langle \hat{s}_t, s^*_t \rangle}{\|\hat{s}_t\|_2 \cdot \|s^*_t\|_2} \\ \mathcal{L}_{\text{cov}}(S) &= \frac{1}{d} \sum_{i \neq j} [\text{Cov}(S)]^2_{i,j} \\ \mathcal{L} &= \mathcal{L}_{\text{dyn}} + \lambda \mathcal{L}_{\text{cov}} \end{aligned} \tag{1}$$

For environments with multiple views, we compute a loss over each view separately and take the mean. We choose $\lambda = 0.04$ following Bardes et al. [36] for the total loss $\mathcal{L}$. We find that covariance regularization slightly improves downstream task performance.

Naively, this objective admits a constant embedding solution. To prevent representation collapse, for $\mathcal{L}_{\text{dyn}}(\hat{s}, s^*)$, we follow SimSiam [37] and set the target embedding $s^*_t := \text{sg}(s_t)$, where sg is the stop gradient operator. Alternatively, our objective is also compatible with a target from a momentum encoder $f_{\bar{\theta}}$ [33, 16], $s^*_t := \bar{s}_t = f_{\bar{\theta}}(o_t)$, where $\bar{\theta}$ is an exponential moving average of θ.

We train all three models end-to-end with the objective in Eq. 1, and use the encoder for downstream control tasks.

4 Experiments

We evaluate our dynamics-pretrained visual representation on a suite of simulated and real benchmarks. We compare DynaMo representations with pretrained representations for vision and control, as well as other self-supervised learning methods. Our experiments are designed to answer the following questions: (a) Does DynaMo improve downstream policy performance? (b) Do representations trained with DynaMo work on real robotic tasks? (c) Is DynaMo compatible with different policy classes? (d) Can pretrained weights be fine-tuned in domain with DynaMo? (e) How important is each component in DynaMo?

4.1 Environments and datasets

We evaluate DynaMo on four simulated benchmarks and two real robot environments (depicted in Figure 4). We provide a brief description below with more details included in Appendix A.

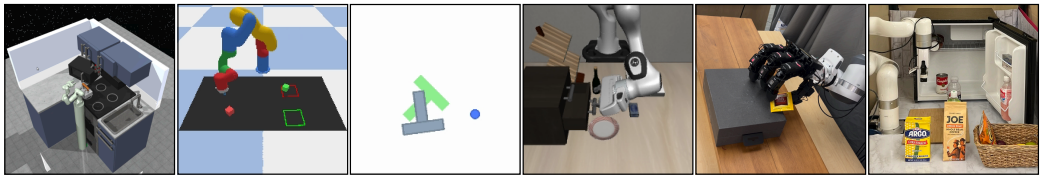
(a) **Franka Kitchen** [27]: The Franka Kitchen environment consists of seven simulated kitchen appliance manipulation tasks with a 9-dimensional action space Franka arm and gripper. The dataset has 566 demonstration trajectories, each completing three or four tasks. The observation space is RGB images of size $(224, 224)$ from a fixed viewpoint. We evaluate for 100 rollouts and report the mean number of completed tasks (maximum 4).

(b) **Block Pushing** [28]: The simulated Block Pushing environment has two blocks, two target areas, and a robot pusher with 2-dimensional action space (end-effector translation). Both the blocks and targets are colored red and green. The task is to push the blocks into either same- or opposite-colored targets. The dataset has 1 000 demonstration trajectories. The observation is RGB images of size $(224, 224)$ from two fixed viewpoints. We evaluate for 100 rollouts and report the mean number of blocks in targets (maximum 2).

(c) **Push-T** [3]: The environment consists of a pusher with 2-dimensional action space, a T-shaped rigid block, and a target area in green. The task is to push the block to cover the target area. The dataset has 206 demonstration trajectories. The observation space is a top-down view of the environment, rendered as RGB images of size $(224, 224)$. We evaluate for 100 rollouts and report the final coverage of the target area (maximum 1).

(d) **LIBERO Goal** [29]: The environment consists of 10 manipulation tasks with a 7-dimensional action space simulated Franka arm and gripper. The dataset has 500 demonstration trajectories in total, 50 per task goal. The observation space is RGB images of size $(224, 224)$ from a fixed external camera, and a wrist-mounted camera. We evaluate a

https://doi.org/10.52202/079017-1069



(a) Franka Kitchen (b) Block Pushing (c) Push-T (d) LIBERO Goal (e) Allegro Manipulation (f) xArm Kitchen

Figure 4: We evaluate DynaMo on four simulated benchmarks - Franka Kitchen, Block Pushing, Push-T, and LIBERO Goal, and two real-world environments - Allegro Manipulation, and xArm Kitchen.

goal-conditioned policy for 100 rollouts in total, 10 per task goal, and report the average success rate (maximum 1).

(e) **Allegro Manipulation**: A real-world environment with an Allegro Hand attached to a Franka arm. We evaluate on three tasks: picking up a sponge (6 demonstrations), picking up a teabag (7 demonstrations), and opening a microwave (6 demonstrations). The observation space is RGB images of size $(224, 224)$ from a fixed external camera. The action space is 23-dimensional, consisting of the Franka pose (7), and Allegro hand joint positions (16).

(f) **xArm Kitchen**: A real-world multi-task kitchen environment with an xArm robot arm and gripper. The environment consists of five manipulation tasks. The dataset includes 65 demonstrations across five tasks. The observation space is RGB images of size $(128, 128)$ from three fixed external cameras, and an egocentric camera attached to the gripper. The action space is 7-dimensional with the robot end effector pose and the gripper state.

4.2 Does DynaMo improve downstream policy performance?

We evaluate each representation by training an imitation policy head on the frozen embeddings, and reporting the downstream task performance on the simulated environments. We use Vector-Quantized Behavior Transformer (VQ-BeT) [1] for the policy head. For xArm Kitchen, we use a goal-conditioned BAKU [38] with a VQ-BeT action head. MAE-style baselines (VC-1, MVP, MAE) use a ViT-B backbone. All other baselines and DynaMo use a ResNet18 backbone.

For environments with multiple views, we concatenate the embeddings from all views for the downstream policy. Further training details are in Appendix B. Table 1 provides comparisons of DynaMo pretrained representations with other self-supervised learning methods, and pretrained weights for vision and robotic manipulation:

- **Random, ImageNet, R3M**: ResNet18 with random, ImageNet-1K, and R3M [9] weights.
- **VC-1**: Pretrained weights from Majumdar et al. [11].
- **MVP**: Pretrained weights from Xiao et al. [8].
- **BYOL**: BYOL [16] pretraining on demonstration data.
- **BYOL-T**: BYOL + temporal contrast [32]. Adjacent frames o_t, o_{t+1} are sampled as positive pairs, in addition to augmentations.
- **MoCo-v3**: MoCo [33] pretraining on demonstration data.
- **RPT**: RPT [39] trained on observation tokens.
- **TCN**: Time-contrastive network [31] pretraining on demonstrations. MV: multi-view objective; SV: single view objective.
- **MAE**: Masked autoencoder [30] pretraining on demonstrations.
- **DynaMo**: DynaMo pretraining on demonstrations.

The best pretrained representation is underlined and the best self-supervised representation is **bolded**. We find that our method matches prior state-of-the-art visual representations on Franka Kitchen, and outperforms all other visual representations on Block Pushing, Push-T, and LIBERO Goal.

Table 1: Downstream policy performance on frozen visual representation on four simulated benchmarks - Franka Kitchen, Blocking Pushing, Push-T, and LIBERO Goal. We observe that DynaMo matches or significantly outperforms prior work on all simulated tasks.

	Method	Franka Kitchen (·/4)	Block Pushing (·/2)	Push-T (·/1)	LIBERO Goal (·/1)
	Random	<u>3.32</u>	0.07	0.07	0.80
Pretrained representations	ImageNet	3.01	<u>0.12</u>	0.41	<u>0.93</u>
	R3M	2.84	0.11	<u>0.49</u>	0.89
	VC-1	2.63	0.05	0.38	0.91
	MVP	2.31	0.00	0.20	0.88
Self-supervised methods	BYOL	**3.75**	0.09	0.23	0.28
	BYOL-T	3.33	0.16	0.34	0.28
	MoCo-v3	3.28	0.03	0.57	0.70
	RPT	3.54	0.52	0.56	0.17
	TCN-MV	—	0.07	—	0.69
	TCN-SV	2.41	0.07	0.07	0.76
	MAE	2.70	0.00	0.07	0.59
	DynaMo	3.64	**0.65**	**0.66**	**0.93**

Table 2: We evaluate DynaMo on eight tasks across two real-world environments: Allegro Manipulation, and xArm Kitchen. Results are presented as (successes/total). We observe that DynaMo significantly outperforms prior representation learning methods on real tasks.

	Task	BYOL	BYOL-T	MoCo-v3	**DynaMo**
Allegro	Sponge	2/10	4/10	5/10	**7/10**
	Tea	1/10	0/10	2/10	**5/10**
	Microwave	2/10	3/10	1/10	**9/10**
xArm Kitchen	Put yogurt	4/5	4/5	2/5	**5/5**
	Get yogurt	0/5	4/5	4/5	**5/5**
	Put ketchup	**5/5**	3/5	**5/5**	4/5
	Get tea	2/5	2/5	3/5	**5/5**
	Get water	0/5	0/5	**3/5**	**3/5**

4.3 Do representations trained with DynaMo work on real robotic tasks?

We evaluate the representations pre-trained with DynaMo on two real-world robot environments: the Allegro Manipulation environment, and the multi-task xArm Kitchen environment. For the Allegro environment, we use a k-nearest neighbors policy [40] and initialize with ImageNet-1K features for all pretraining methods, as the dataset is relatively small with around 1 000 frames per task. In the xArm Kitchen environment, we use the BAKU [38] architecture for goal-conditioned rollouts across five tasks. For our real-robot evaluations, we compare DynaMo against the strongest performing baselines from our simulated experiments (see Table 1). The results are reported in Table 2. We observe that DynaMo outperforms the best baseline by 43% on the single-task Allegro hand and by 20% on the multi-task xArm Kitchen environment. Additionally, as shown in Table 3, DynaMo exceeds the performance of pretrained representations by 50% on the Allegro hand. These results demonstrate that DynaMo is capable of learning effective robot representations in both single-task and multi-task settings.

4.4 Is DynaMo compatible with different policy classes?

On the Push-T environment [3], we compare all pretrained representations across four policy classes: VQ-BeT [1], Diffusion Policy [3], MLP (with action chunking [2]), and k-nearest neighbors with locally weighted regression [40]. We present the results in Table 4. We find that DynaMo representa-

https://doi.org/10.52202/079017-1069

Table 4: We evaluate the compatibility of DynaMo with different policy classes for downstream policy learning on the Push-T simulated benchmark. We report the final target coverage achieved (maximum 1) and demonstrate that DynaMo significantly outperforms prior representation learning methods across all policy classes.

	Method	VQ-BeT	Diffusion	MLP (chunking)	kNN
	Random	0.07	0.04	0.07	0.01
Pretrained representations	ImageNet	0.41	**0.73**	0.24	0.09
	R3M	0.49	0.63	0.27	0.08
	VC-1	0.38	0.63	0.22	0.07
	MVP	0.20	0.49	0.11	0.08
Self-supervised methods	BYOL	0.23	0.40	0.11	0.04
	BYOL-T	0.34	0.50	0.16	0.04
	MoCo v3	0.57	0.67	0.30	0.07
	RPT	0.56	0.62	0.30	0.07
	TCN-SV	0.07	0.14	0.07	0.01
	MAE	0.07	0.06	0.07	0.02
	DynaMo	**0.66**	**0.73**	**0.35**	**0.12**

Table 5: We evaluate the ability of DynaMo to finetune an ImageNet-pretrained ResNet-18 encoder across 4 benchmarks. We demonstrate that using a pretrained encoder can further improve the performance of DynaMo.

Representation	Franka Kitchen ($\cdot/4$)	Block Pushing ($\cdot/2$)	Push-T ($\cdot/1$)	LIBERO Goal ($\cdot/1$)
ImageNet	3.01	0.12	0.41	**0.93**
DynaMo (random init)	3.64	0.65	**0.66**	**0.93**
DynaMo (ImageNet fine-tuned)	**3.82**	**0.67**	0.50	0.90

tions improve downstream policy performance across policy classes compared to prior state-of-the-art representations. We also note that our representation works on the robot hand in §4.3 with a nearest neighbor policy.

4.5 Can pretrained weights be fine-tuned in domain with DynaMo?

We fine-tune an ImageNet-1K-pretrained ResNet18 with DynaMo for each simulated environment, and evaluate with downstream policy performance on the frozen representation as described in §4.2. The results are shown in Table 5. We find that DynaMo is compatible with ImageNet initialization, and can be used to fine-tune out-of-domain pretrained weights to further improve in-domain task performance. We also note that our method works in the low-data regime with ImageNet initialization on the real Allegro hand in Table 2.

Table 3: Pretrained baselines on Allegro

Method	Sponge	Tea	Microwave
ImageNet	4/10	1/10	0/10
R3M	1/10	1/10	5/10
DynaMo	**7/10**	**5/10**	**9/10**

4.6 How important is each component in DynaMo?

In Table 6, we ablate each component in DynaMo and measure its impact on downstream policy performance on our simulated benchmarks.

Forward dynamics prediction: We replace the one-step forward prediction target $s^*_{1:h}$ with the same-step target $s^*_{:h-1}$. To prevent the model from trivially predicting s^*_t given s_t, we replace the forward dynamics input $(s_{:h-1}, z_{:h-1})$ with only $z_{:h-1}$. The ablated objective is essentially a variant of autoencoding s_t. We observe that removing forward dynamics prediction degrades performance across environments.

https://doi.org/10.52202/079017-1069

Table 6: Ablation analysis of downstream performance relative to the full architecture (100%)

Ablations	Kitchen	Block	Push-T	LIBERO
No forward	34%	8%	44%	33%
No inverse	72%	35%	97%	41%
No bottleneck	92%	22%	9%	75%
No cov. reg.	94%	62%	85%	59%
No stop grad.	1%	5%	9%	0%
Short context	100%	75%	88%	89%

Table 7: Variants with ground truth actions, downstream performance relative to the base model (100%)

Variants	Kitchen	Block	Push-T	LIBERO
Inverse dynamics only	100%	54%	70%	11%
DynaMo + action labels	97%	29%	94%	86%

Inverse dynamics to a transition latent: As described in §3.1, the forward dynamics loss assumes that the transition is unimodal and requires an inferred transition latent. We observed that removing the latent from the forward dynamics input results in a significant performance drop.

Bottleneck on the transition latent dimension: For the transition latent z and the observation embedding s, we find that having $\dim z \ll \dim s$ stabilizes training. Here we set $\dim z := \dim s$, and find that our model can still learn a reasonable representation in some environments, but training can destabilize, leading to a high variance in downstream performance.

Covariance regularization: We find that covariance regularization from Bardes et al. [36] improves performance across environments. Training still converges without it, but the downstream performance is slightly worse.

Stop gradient on target embeddings: We observe that removing techniques like momentum encoder [33, 16] and stop gradient [37] leads to representation collapse [41, 16, 36].

Observation context: The dynamics objective requires at least 2 frames of observation context. For Franka Kitchen, we find that a context of 2 frames works best. For the other environments, a longer observation context (5 frames) improves downstream policy performance. Details of hyperparameters used for DynaMo visual pretraining can be found in Appendix B.1.

4.7 Variants with access to ground truth actions

In Table 7, we compare with two variants of DynaMo where we assume access to ground truth action labels during visual encoder training.

Only inverse dynamics to ground truth actions: as proposed in Brandfonbrener et al. [26], we train the visual encoder by learning an inverse dynamics model to ground truth actions, with covariance regularization, and without forward dynamics.

Full model + inverse dynamics to ground truth actions: we train the full DynaMo model plus an MLP head to predict the ground truth actions given the transition latents inferred by the inverse dynamics model.

We observe that in both cases, having access to ground truth actions during visual pretraining does not seem to improve downstream policy performance. We hypothesize that this is because the downstream policy already has access to the same actions for imitation learning.

5 Related works

This work builds on a large body of research on self-supervised visual representations, learning from human demonstrations, neuroscientific basis for learning dynamics for control, predictive models for decision making, learning from videos for control, and visual pretraining for control.

https://doi.org/10.52202/079017-1069

Self-supervised visual representations: Self-supervised visual representations have been widely studied since the inception of deep learning. There are several common approaches to self-supervised visual representation learning. One approach is to recover the ground truth from noise-degraded samples using techniques like denoising autoencoders [42, 43] and masked modeling [44, 45, 30]. Another approach is contrastive learning, which leverages data augmentation priors [41, 16, 33, 36, 37] or temporal proximity [31, 46] to produce contrastive sample pairs. A third self-supervised method is generative modeling [47–49], which learns to sequentially generate the ground truth data. More recently, self-supervision in the latent space rather than the raw pixel space has proven effective, as seen in methods that predict representations in latent space [50, 51].

Learning from demonstrations: Learning from human demonstrations is a well-established idea in robotics [52–55]. With the advances in deep learning, recent works such as [3, 2, 5, 4, 1, 56] show that imitation learning from human demonstrations has become a viable approach for training robotic policies in simulated and real-world settings.

Neural basis for learning dynamics: It is widely believed that animals possess internal dynamics models that facilitate motor control. These models learn representations that are predictive of sensory inputs for decision making and motor control [57–60]. Early works such as [17–20] propose that there exists an internal model of the motor apparatus in the cerebellum for motor control and planning. [21, 22] propose that the central nervous system uses forward models that predict motor command outcomes and model the environment. Learning forward and inverse dynamics models also helps with generalization to diverse task conditions [23, 24].

Predictive models for decision making: Predictive model learning for decision making is well-established in machine learning. Learning generative models that can predict sequential inputs has achieved success across many domains, such as natural language processing [61], reinforcement learning [62–64], and representation learning [46, 65]. Incorporating the prediction of future states as an intrinsic reward has also been shown to improve reinforcement learning performance [66–68]. Moreover, recent work demonstrates that world models trained to predict environment dynamics can enable planning in complex tasks and environments [69–73].

Learning from video for control: Videos provide rich spatiotemporal information that can be leveraged for self-supervised representation learning [74–79]. These methods have been extended to decision-making through effective downstream policy learning [7–11, 6]. Further, recent work also enables learning robotic policies directly from in-domain human demonstration videos by incorporating some additional priors [80–84], as well as learning behavioral priors from actionless demonstration data [85–87].

Visual representation for control: Visual representation learning for control has been an active area of research. Prior work has shown that data augmentation improves the robustness of learned representations and policy performance in reinforcement learning domains [88, 89]. Additionally, pretraining visual representations on large out-of-domain datasets before fine-tuning for control tasks has been shown to outperform training policies from scratch [10, 12, 9, 11, 90, 8, 91]. More recent work has shown that in-domain self-supervised pretraining improves policy performance [92–95] and enables non-parametric downstream policies [40].

6 Discussion and Limitations

In this work, we have presented DynaMo, a self-supervised algorithm for robot representation learning that leverages the sequential nature of demonstration data. DynaMo incorporates predictive dynamics modeling to learn visual features that capture the sequential structure of demonstration observations. During pretraining, DynaMo jointly optimizes the visual encoder with dynamics models to extract task-specific features. These learned representations can then be used for downstream control tasks, leading to more efficient policy learning compared to prior approaches. We believe that training DynaMo on larger unlabeled datasets could potentially improve generalization. Additionally, while promising for control tasks, more research is needed to evaluate DynaMo's effectiveness on robotic manipulation outside of lab settings.

Acknowledgements

We would like to thank Ademi Adeniji, Alex Wang, Gaoyue Zhou, Haritheja Etukuru, Irmak Güzey, Mahi Shafiullah, Nikhil Bhattasali, Raunaq Bhirangi, Seungjae (Jay) Lee, and Ulyana Piterbarg for their valuable feedback and discussions. This work was supported by grants from Honda, Google, NSF award 2339096 and ONR awards N00014-21-1-2758 and N00014-22-1-2773. LP is supported by the Packard Fellowship.

References

[1] Seungjae Lee, Yibin Wang, Haritheja Etukuru, H Jin Kim, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024. 1, 6, 7, 10, 21

[2] Tony Z Zhao, Vikash Kumar, Sergey Levine, and Chelsea Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023. 1, 7, 10

[3] Cheng Chi, Siyuan Feng, Yilun Du, Zhenjia Xu, Eric Cousineau, Benjamin Burchfiel, and Shuran Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023. 2, 5, 7, 10, 17

[4] Zichen Jeff Cui, Yibin Wang, Nur Muhammad Mahi Shafiullah, and Lerrel Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022. 1, 10

[5] Nur Muhammad Shafiullah, Zichen Cui, Ariuntuya Arty Altanzaya, and Lerrel Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022. 1, 4, 10

[6] 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. 1, 3, 10

[7] 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. 10

[8] Tete Xiao, Ilija Radosavovic, Trevor Darrell, and Jitendra Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022. 2, 3, 6, 10

[9] 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. 6, 10

[10] Simone Parisi, Aravind Rajeswaran, Senthil Purushwalkam, and Abhinav Gupta. The unsurprising effectiveness of pre-trained vision models for control. In *international conference on machine learning*, pages 17359–17371. PMLR, 2022. 10

[11] Arjun Majumdar, Karmesh Yadav, Sergio Arnaud, Jason Ma, Claire Chen, Sneha Silwal, Aryan Jain, Vincent-Pierre Berges, Tingfan Wu, Jay Vakil, et al. Where are we in the search for an artificial visual cortex for embodied intelligence? *Advances in Neural Information Processing Systems*, 36, 2024. 1, 2, 3, 6, 10

[12] Sudeep Dasari, Mohan Kumar Srirama, Unnat Jain, and Abhinav Gupta. An unbiased look at datasets for visuo-motor pre-training. In *Conference on Robot Learning*, pages 1183–1198. PMLR, 2023. 1, 3, 10

[13] Sridhar Pandian Arunachalam, Irmak Güzey, Soumith Chintala, and Lerrel Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5962–5969. IEEE, 2023. 1

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

 https://doi.org/10.52202/079017-1069

[15] Xinlei Chen, Saining Xie, and Kaiming He. An empirical study of training self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 9640–9649, 2021. 2

[16] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. 2, 3, 5, 6, 9, 10

[17] Daniel M Wolpert, Zoubin Ghahramani, and Michael I Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995. 2, 10

[18] Daniel M Wolpert, R Chris Miall, and Mitsuo Kawato. Internal models in the cerebellum. *Trends in cognitive sciences*, 2(9):338–347, 1998.

[19] M Shidara, K Kawano, H Gomi, and M Kawato. Inverse-dynamics model eye movement control by purkinje cells in the cerebellum. *Nature*, 365(6441):50–52, 1993.

[20] Shigeru Kitazawa, Tatsuya Kimura, and Ping-Bo Yin. Cerebellar complex spikes encode both destinations and errors in arm movements. *Nature*, 392(6675):494–497, 1998. 10

[21] R Chris Miall and Daniel M Wolpert. Forward models for physiological motor control. *Neural networks*, 9(8):1265–1279, 1996. 10

[22] Michael I Jordan and David E Rumelhart. Forward models: Supervised learning with a distal teacher. *Cognitive Science*, 16(3):307–354, 1992. 10

[23] J Randall Flanagan and Alan M Wing. The role of internal models in motion planning and control: evidence from grip force adjustments during movements of hand-held loads. *Journal of Neuroscience*, 17(4):1519–1528, 1997. 10

[24] Masahiko Haruno, Daniel M Wolpert, and Mitsuo Kawato. Multiple paired forward-inverse models for human motor learning and control. *Advances in neural information processing systems*, 11, 1998. 2, 10

[25] William Whitney, Rajat Agarwal, Kyunghyun Cho, and Abhinav Gupta. Dynamics-aware embeddings. *arXiv preprint arXiv:1908.09357*, 2019. 2

[26] David Brandfonbrener, Ofir Nachum, and Joan Bruna. Inverse dynamics pretraining learns good representations for multitask imitation. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 9

[27] Abhishek Gupta, Vikash Kumar, Corey Lynch, Sergey Levine, and Karol Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019. 2, 5, 17

[28] Pete Florence, Corey Lynch, Andy Zeng, Oscar A Ramirez, Ayzaan Wahid, Laura Downs, Adrian Wong, Johnny Lee, Igor Mordatch, and Jonathan Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022. 2, 3, 5, 17

[29] Bo Liu, Yifeng Zhu, Chongkai Gao, Yihao Feng, Qiang Liu, Yuke Zhu, and Peter Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024. 2, 5, 17

[30] 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. 3, 6, 10

[31] 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. 3, 6, 10

[32] Sarah Young, Jyothish Pari, Pieter Abbeel, and Lerrel Pinto. Playful interactions for representation learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 992–999. IEEE, 2022. 3, 6

[33] 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. 3, 5, 6, 9, 10

[34] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[35] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 4

[36] Adrien Bardes, Jean Ponce, and Yann LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021. 5, 9, 10

[37] Xinlei Chen and Kaiming He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021. 5, 9, 10

[38] Siddhant Haldar, Zhuoran Peng, and Lerrel Pinto. Baku: An efficient transformer for multi-task policy learning. *arXiv preprint arXiv:2406.07539*, 2024. 6, 7

[39] Ilija Radosavovic, Baifeng Shi, Letian Fu, Ken Goldberg, Trevor Darrell, and Jitendra Malik. Robot learning with sensorimotor pre-training. In *Conference on Robot Learning*, pages 683–693. PMLR, 2023. 6

[40] Jyothish Pari, Nur Muhammad Shafiullah, Sridhar Pandian Arunachalam, and Lerrel Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021. 7, 10

[41] 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. 9, 10

[42] Weilai Xiang, Hongyu Yang, Di Huang, and Yunhong Wang. Denoising diffusion autoencoders are unified self-supervised learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15802–15812, 2023. 10

[43] Vladimiros Sterzentsenko, Leonidas Saroglou, Anargyros Chatzitofis, Spyridon Thermos, Nikolaos Zioulis, Alexandros Doumanoglou, Dimitrios Zarpalas, and Petros Daras. Self-supervised deep depth denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1242–1251, 2019. 10

[44] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018. 10

[45] Hangbo Bao, Li Dong, Songhao Piao, and Furu Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021. 10

[46] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018. 10

[47] Mark Chen, Alec Radford, Rewon Child, Jeffrey Wu, Heewoo Jun, David Luan, and Ilya Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR, 2020. 10

[48] Aäron Van Den Oord, Nal Kalchbrenner, and Koray Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

 https://doi.org/10.52202/079017-1069

[49] Trieu H Trinh, Minh-Thang Luong, and Quoc V Le. Selfie: Self-supervised pretraining for image embedding. *arXiv preprint arXiv:1906.02940*, 2019. 10

[50] Mahmoud Assran, Quentin Duval, Ishan Misra, Piotr Bojanowski, Pascal Vincent, Michael Rabbat, Yann LeCun, and Nicolas Ballas. Self-supervised learning from images with a joint-embedding predictive architecture. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15619–15629, 2023. 10

[51] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mido Assran, and Nicolas Ballas. V-jepa: Latent video prediction for visual representation learning. 2023. 10

[52] Nathan Delson and Harry West. Robot programming by human demonstration: Adaptation and inconsistency in constrained motion. In *Proceedings of IEEE International conference on Robotics and Automation*, volume 1, pages 30–36. IEEE, 1996. 10

[53] Michael Kaiser and Rüdiger Dillmann. Building elementary robot skills from human demonstration. In *Proceedings of IEEE International Conference on Robotics and Automation*, volume 3, pages 2700–2705. IEEE, 1996.

[54] Sheng Liu and Haruhiko Asada. Teaching and learning of deburring robots using neural networks. In *[1993] Proceedings IEEE International Conference on Robotics and Automation*, pages 339–345. IEEE, 1993.

[55] Haruhiko Asada and Boo-Ho Yang. Skill acquisition from human experts through pattern processing of teaching data. *Journal of The Robotics Society of Japan*, 8(1):17–24, 1990. 10

[56] Moritz Reuss, Maximilian Li, Xiaogang Jia, and Rudolf Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023. 10

[57] Richard S Sutton and Andrew G Barto. Toward a modern theory of adaptive networks: expectation and prediction. *Psychological review*, 88(2):135, 1981. 10

[58] Hermann Von Helmholtz. *Handbuch der physiologischen Optik*, volume 9. Voss, 1867.

[59] Andre M Bastos, W Martin Usrey, Rick A Adams, George R Mangun, Pascal Fries, and Karl J Friston. Canonical microcircuits for predictive coding. *Neuron*, 76(4):695–711, 2012.

[60] Lisa Feldman Barrett and W Kyle Simmons. Interoceptive predictions in the brain. *Nature reviews neuroscience*, 16(7):419–429, 2015. 10

[61] 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. 10

[62] Younggyo Seo, Kimin Lee, Stephen L James, and Pieter Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pages 19561–19579. PMLR, 2022. 10

[63] Max Schwarzer, Ankesh Anand, Rishab Goel, R Devon Hjelm, Aaron Courville, and Philip Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

[64] Max Schwarzer, Nitarshan Rajkumar, Michael Noukhovitch, Ankesh Anand, Laurent Charlin, R Devon Hjelm, Philip Bachman, and Aaron C Courville. Pretraining representations for data-efficient reinforcement learning. *Advances in Neural Information Processing Systems*, 34: 12686–12699, 2021. 10

[65] Karl Schmeckpeper, Annie Xie, Oleh Rybkin, Stephen Tian, Kostas Daniilidis, Sergey Levine, and Chelsea Finn. Learning predictive models from observation and interaction. In *European Conference on Computer Vision*, pages 708–725. Springer, 2020. 10

[66] Deepak Pathak, Pulkit Agrawal, Alexei A Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017. 10

[67] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

[68] Zhaohan Guo, Shantanu Thakoor, Miruna Pîslar, Bernardo Avila Pires, Florent Altché, Corentin Tallec, Alaa Saade, Daniele Calandriello, Jean-Bastien Grill, Yunhao Tang, et al. Byol-explore: Exploration by bootstrapped prediction. *Advances in neural information processing systems*, 35:31855–31870, 2022. 10

[69] 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. 10

[70] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[71] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[72] Danijar Hafner, Timothy Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[73] Jake Bruce, Michael D Dennis, Ashley Edwards, Jack Parker-Holder, Yuge Shi, Edward Hughes, Matthew Lai, Aditi Mavalankar, Richie Steigerwald, Chris Apps, et al. Genie: Generative interactive environments. In *Forty-first International Conference on Machine Learning*, 2024. 10

[74] Ross Goroshin, Joan Bruna, Jonathan Tompson, David Eigen, and Yann LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *Proceedings of the IEEE international conference on computer vision*, pages 4086–4093, 2015. 10

[75] Adrien Bardes, Quentin Garrido, Jean Ponce, Xinlei Chen, Michael Rabbat, Yann LeCun, Mahmoud Assran, and Nicolas Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

[76] Christoph Feichtenhofer, Haoqi Fan, Bo Xiong, Ross Girshick, and Kaiming He. A large-scale study on unsupervised spatiotemporal representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 3299–3309, 2021.

[77] Xiaolong Wang, Allan Jabri, and Alexei A Efros. Learning correspondence from the cycle-consistency of time. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2566–2576, 2019.

[78] Debidatta Dwibedi, Yusuf Aytar, Jonathan Tompson, Pierre Sermanet, and Andrew Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1801–1810, 2019.

[79] Sören Pirk, Mohi Khansari, Yunfei Bai, Corey Lynch, and Pierre Sermanet. Online object representations with contrastive learning. *arXiv preprint arXiv:1906.04312*, 2019. 10

[80] Shikhar Bahl, Abhinav Gupta, and Deepak Pathak. Human-to-robot imitation in the wild. *arXiv preprint arXiv:2207.09450*, 2022. 10

[81] Pratyusha Sharma, Lekha Mohan, Lerrel Pinto, and Abhinav Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*, pages 906–915. PMLR, 2018.

[82] Boyuan Chen, Pieter Abbeel, and Deepak Pathak. Unsupervised learning of visual 3d keypoints for control. In *International Conference on Machine Learning*, pages 1539–1549. PMLR, 2021.

[83] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision*, pages 570–587. Springer, 2022.

 https://doi.org/10.52202/079017-1069

[84] Aravind Sivakumar, Kenneth Shaw, and Deepak Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. *arXiv preprint arXiv:2202.10448*, 2022. 10

[85] Ashley Edwards, Himanshu Sahni, Yannick Schroecker, and Charles Isbell. Imitating latent policies from observation. In *International conference on machine learning*, pages 1755–1763. PMLR, 2019. 10

[86] Dominik Schmidt and Minqi Jiang. Learning to act without actions. *arXiv preprint arXiv:2312.10812*, 2023.

[87] Weirui Ye, Yunsheng Zhang, Pieter Abbeel, and Yang Gao. Become a proficient player with limited data through watching pure videos. In *The Eleventh International Conference on Learning Representations*, 2022. 10

[88] 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. 10

[89] Misha Laskin, Kimin Lee, Adam Stooke, Lerrel Pinto, Pieter Abbeel, and Aravind Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020. 10

[90] Ilija Radosavovic, Tete Xiao, Stephen James, Pieter Abbeel, Jitendra Malik, and Trevor Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, pages 416–426. PMLR, 2023. 10

[91] Abhishek Padalkar, Acorn Pooley, Ajinkya Jain, Alex Bewley, Alex Herzog, Alex Irpan, Alexander Khazatsky, Anant Rai, Anikait Singh, Anthony Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023. 10

[92] Nur Muhammad Mahi Shafiullah, Anant Rai, Haritheja Etukuru, Yiqian Liu, Ishan Misra, Soumith Chintala, and Lerrel Pinto. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023. 10

[93] 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. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 9197–9203. IEEE, 2023.

[94] Irmak Guzey, Ben Evans, Soumith Chintala, and Lerrel Pinto. Dexterity from touch: Self-supervised pre-training of tactile representations with robotic play. *arXiv preprint arXiv:2303.12076*, 2023.

[95] Ruijie Zheng, Yongyuan Liang, Xiyao Wang, Shuang Ma, Hal Daumé III, Huazhe Xu, John Langford, Praveen Palanisamy, Kalyan Shankar Basu, and Furong Huang. Premier-taco: Pretraining multitask representation via temporal action-driven contrastive loss. *arXiv preprint arXiv:2402.06187*, 2024. 10

[96] Aadhithya Iyer, Zhuoran Peng, Yinlong Dai, Irmak Guzey, Siddhant Haldar, Soumith Chintala, and Lerrel Pinto. Open teach: A versatile teleoperation system for robotic manipulation, 2024. 17, 18

[97] Andrej Karpathy. nanogpt. https://github.com/karpathy/nanoGPT, 2023. Accessed: 2024-05-20. 20

A Environment and dataset details

A.1 Franka Kitchen

The Franka Kitchen environment introduced by Gupta et al. [27] consists of a Franka arm with a 9-dimensional action space. This environment includes seven tasks and a dataset of 566 human-collected demonstrations. While the original environment is state-based, we created an image-based variant by rendering the states to 224×224 RGB images.

A.2 Block Pushing

In the Block Pushing environment introduced by Florence et al. [28], the objective is for the robot to push two colored blocks (red and green) into two target squares (also red and green). The training dataset consists of 1 000 trajectories, evenly distributed among the four possible combinations of block target and push order. These trajectories were collected by a scripted expert controller.

A.3 Push-T

In the Push-T environment introduced by Chi et al. [3], the goal is to push a T-shaped block to a designated target position on a table. The dataset for this environment contains 206 demonstrations collected by human operators. The action space in this environment is a two-dimensional end-effector position control. Similar to the Franka Kitchen environment, we have created an image-based variant by rendering demonstrations to 224×224 RGB images.

A.4 LIBERO Goal

In the LIBERO Goal environment introduced by Liu et al. [29], there are 10 manipulation tasks, each with 50 teleoperated demonstrations for goal-conditioned policy benchmarking. The environment has a 7-dimensional action space and an observation space of 224×224 RGB images from two cameras (fixed external view, and wrist-mounted egocentric view).

A.5 Allegro Manipulation

The environment consists of an Allegro hand attached to a Franka arm, and a fixed camera for image observations. The observation space is 224×224 RGB images. The action space is 23-dimensional, consisting of Cartesian position and orientation of the Franka robot arm (7 DoF), and 16 joint positions of the Allegro Robot Hand. The demonstrations are collected at 50Hz for Franka, and 60Hz for the Allegro hand. The learned policies are rolled out at 4Hz.

We evaluate on three contact-rich dexterous manipulation tasks that require precise multi-finger control and arm movement, described in detail below.

Sponge picking: This task requires the hand to reach to the position of the sponge, grasp the sponge, and lift the sponge from the table. We collect 6 demonstrations via OpenTeach [96] for the task, starting from different positions, with 543 frames in total. The task is considered successful if the robot hand can grasp the sponge from the table within 120 seconds.

Teabag picking: This task is similar to the previous task, but more difficult with a smaller task object. We collect 7 demonstrations via OpenTeach with 1 034 frames in total. In this task, the robot needs reach the teabag, grasp the teabag with two fingers, then pick it up. The task is considered successful if the robot hand can grasp the teabag from the table within 240 seconds.

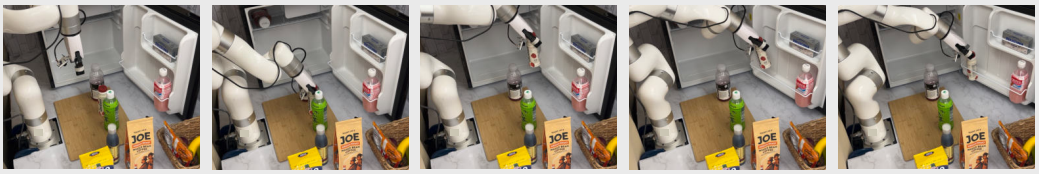
Microwave opening: This task requires the hand to reach the microwave door handle, grasp the handle, and pull down the door. We collect 6 demonstrations via OpenTeach with 735 frames in total. The task is considered successful if the robot hand can open the door within 240 seconds.

A.6 xArm Kitchen

This is a real-world multi-task kitchen environment comprising a Ufactory xArm 7 robot with an xArm Gripper. The policies are trained on RGB images of size 128×128 obtained from four different camera views, including an egocentric camera attached to the robot gripper. The action space

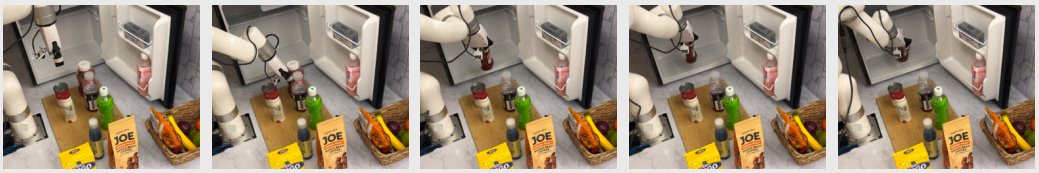
https://doi.org/10.52202/079017-1069

comprises the robot end effector pose and the gripper state. We collect a total of 65 demonstrations across 5 tasks, depicted in Figure 5. The demonstrations were collected using OpenTeach [96] at 30Hz. The learned policies are deployed at 10Hz. Figure 5 shows real-world task rollouts for the multitask policy learned for all 5 tasks.



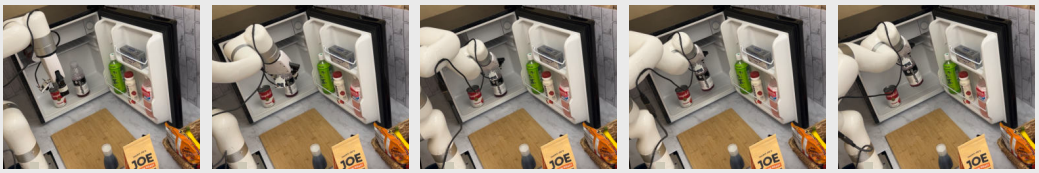
Put yogurt bottle in fridge door: Pick up the bottle of yogurt and place it in the door of the fridge.

Fetch yogurt bottle from fridge door: Take the bottle of yogurt out from the door of the fridge.



Put ketchup bottle inside fridge: Pick up the bottle of tomato ketchup and put it inside the fridge.

Fetch tea bottle from fridge door: Take the bottle of green tea out from the door of the fridge.



Fetch water bottle from fridge: Take the bottle of vitamin water out of the fridge.

Figure 5: xArm Kitchen environment tasks

B Hyperparameters and implementation details

B.1 Visual encoder training

We present the DynaMo hyperparameters below.

Table 8: Environment-dependent hyperparameters for DynaMo pretraining, random init

	Obs. context	EMA β	Forward dynamics dropout	Transition latent dim
Franka Kitchen	2	SimSiam	0	64
Block Pushing	5	0.99	0.3	16
Push-T	5	SimSiam	0	8
LIBERO Goal	5	SimSiam	0	32
xArm Kitchen	5	0.99	0	64

Table 9: Shared hyperparameters for DynaMo pretraining, random init

Name	Value
Optimizer	AdamW
Learning rate	10^{-4}
Weight decay	0.0
Betas	(0.9, 0.999)
Gradient clip norm	0.1
Covariance reg. coefficient	0.04
Epochs	40
Batch size	64

Table 10: Environment-dependent hyperparameters for DynaMo fine-tuning from ImageNet weights

	Obs. context	EMA β	Transition latent dim
Franka Kitchen	2	SimSiam	64
Block Pushing	5	0.99	16
Push-T	5	SimSiam	8
LIBERO Goal	5	0.99	32
Allegro	5	SimSiam	32

Table 11: Shared hyperparameters for DynaMo fine-tuning

Name	Value
Optimizer	AdamW
Learning rate	10^{-5}
Forward dynamics dropout	0.0
Weight decay	0.0
Betas	(0.9, 0.999)
Gradient clip norm	0.1
Covariance reg. coefficient	0.04
Epochs	40
Batch size	64

For Block Pushing and xArm kitchen, we use an EMA encoder with the beta schedule from the MoCo-v3 official repo. For DynaMo training, we use a constant learning rate schedule for LIBERO

https://doi.org/10.52202/079017-1069

Goal, and a cosine learning rate decay schedule with 5 warmup epochs on all other environments. For DynaMo fine-tuning, we use a cosine learning rate decay schedule with 5 warmup epochs on all environments.

We use the following official implementation repos:

- MoCo-v3: https://github.com/facebookresearch/moco-v3
- BYOL: https://github.com/lucidrains/byol-pytorch
- MAE: https://github.com/facebookresearch/mae
- R3M: https://github.com/facebookresearch/r3m/
- MVP: https://github.com/ir413/mvp
- VC-1: https://github.com/facebookresearch/eai-vc

We base our transformer encoder implementation on nanoGPT [97] at https://github.com/karpathy/nanoGPT.

For the Allegro Manipulation environment, we fine-tune MoCo and BYOL from ImageNet-1K weights for 1 000 epochs. For all other environments, we train MoCo and BYOL for 200 epochs, MAE for 400 epochs, all from random initialization. The hyperparameters used for training these models are detailed in Table 12.

Compute used for training DynaMo:

- Franka Kitchen: 3 hours on 1x NVIDIA A100.
- Block Pushing: 7 hours on 1x NVIDIA A100.
- Push-T: 1 hour on 1x NVIDIA A100.
- LIBERO Goal: 2 hours on 1x NVIDIA H100.
- Allegro Manipulation: 3 minutes on 1x NVIDIA RTX A6000 for the sponge task, 4 minutes for the teabag task, and 3 minutes for the microwave task.
- xArm kitchen: 4 hours on 1x NVIDIA RTX A6000.

Table 12: SSL Hyperparameters

(a) MoCo Hyperparameters

Name	Value
Optimizer	LARS
Batch size	1024
Learning rate	0.6
Momentum	0.9
Weight decay	10^{-6}

(b) BYOL Hyperparameters

Name	Value
Optimizer	LARS
Batch size	512
Learning rate	0.2
Momentum	0.9
Weight decay	1.5×10^{-6}

(c) MAE Hyperparameters

Name	Value
Optimizer	AdamW
Batch size	64
Learning rate	2.5×10^{-5}
Weight decay	0.05

B.2 Downstream policy training

Table 13, 14 and 15 detail the downstream policy hyperparameters for VQ-BeT, Diffusion Policy and MLP training for the simulated environments.

For VQ-BeT, we use the implementation from the original paper [1] with the recommended hyperparameters. For Diffusion Policy, we use the implementation at https://github.com/real-stanford/diffusion_policy with a transformer-based noise prediction network with the recommended hyperparameters. We use AdamW as optimizer for the three policy heads.

Compute used for downstream policy training:

- Franka Kitchen VQ-BeT: 8.5 hours on 1x NVIDIA A4000.
- Block Pushing VQ-BeT: 4 hours on 1x NVIDIA A100.
- Push-T VQ-BeT: 7 hours on 1x NVIDIA A100.
- Push-T Diffusion Policy: 8 hours on 1x NVIDIA A100.
- Push-T MLP: 2 hours on 1x NVIDIA A100.
- LIBERO Goal VQ-BeT: 5 hours on 1x NVIDIA A4000.
- xArm Kitchen VQ-BeT: 6 hours on 1x NVIDIA A4000.

Table 13: Hyperparameters for VQ-BeT training

Parameter	Franka Kitchen	Block Pushing	Push-T	LIBERO Goal
Batch size	2048	64	512	64
Epochs	1000	300	5000	50
Window size	10	3	5	10
Prediction window size	1	1	5	1
Learning rate	5.5×10^{-5}	10^{-4}	5.5×10^{-5}	5.5×10^{-5}
Weight decay	2×10^{-4}	0	2×10^{-4}	2×10^{-4}

Table 14: Hyperparameters for Diffusion Policy Training

Parameter	Push-T
Batch size	256
Epochs	2000
Learning rate	10^{-4}
Weight decay	0
Observation horizon	2
Prediction horizon	10
Action horizon	8

Table 15: Hyperparameters for MLP Training

Parameter	Push-T
Batch size	256
Epochs	2000
Learning rate	10^{-4}
Weight decay	0
Hidden dim	256
Hidden depth	8
Observation context	5
Prediction context	5

https://doi.org/10.52202/079017-1069

C Real robot environment rollouts

Figure 6: Rollouts on Allegro Manipulation with our DynaMo-pretrained encoder.

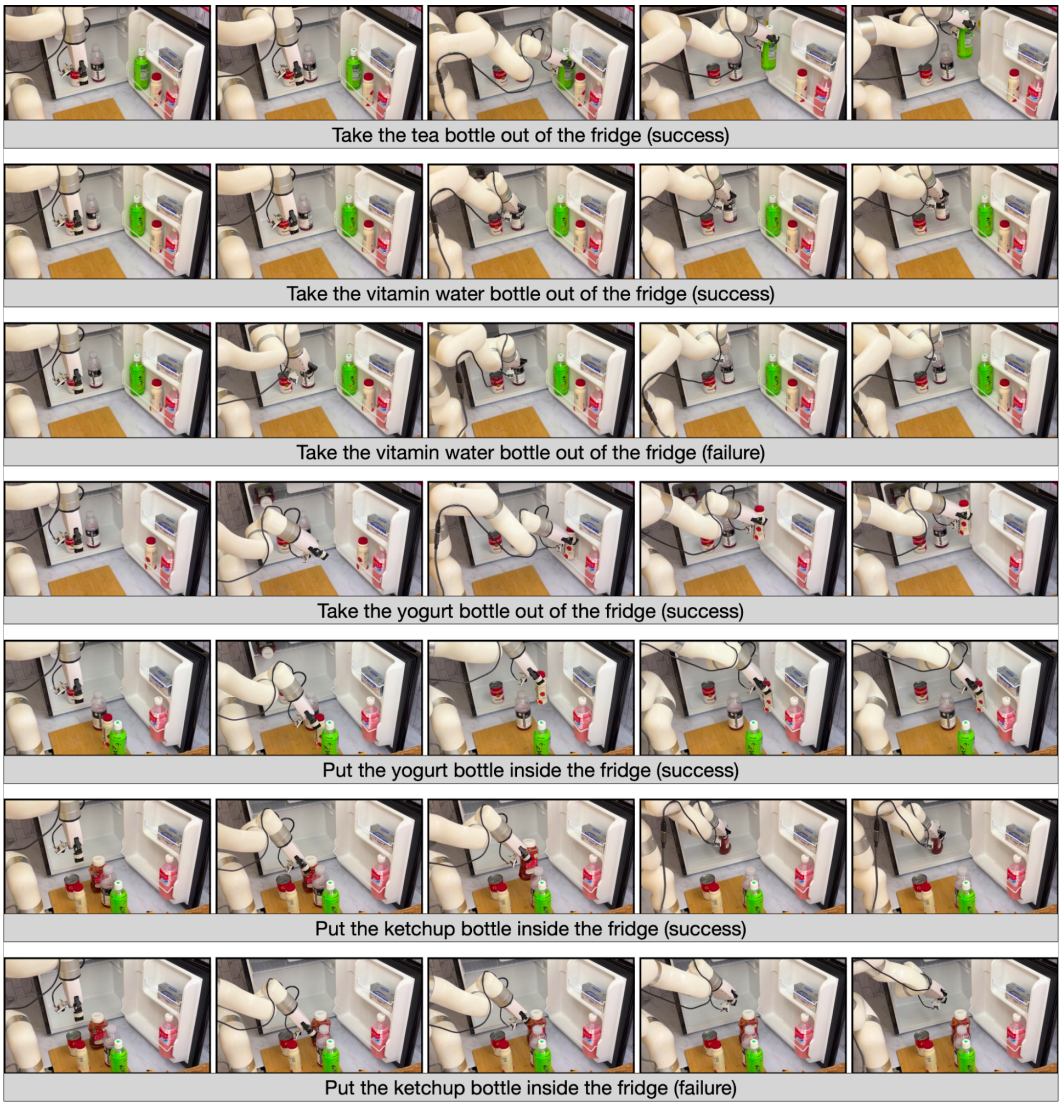


Figure 7: Rollouts on xArm Kitchen with our DynaMo-pretrained encoder.

D Broader Impacts

In this work, we present DynaMo, a new self-supervised learning method for pretraining in-domain visual representations for downstream policy learning. Our work takes an important step towards improves data efficiency by explicitly modeling the dynamics of the demonstration observations, improving upon prior state-of-the-art self-supervised methods, especially in the low-data regime, which can be valuable for robotics and visuomotor policy learning research with limited in-domain data.

https://doi.org/10.52202/079017-1069

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: We state the claims in the abstract and Section 1, and support them with experimental results in Section 4.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We discuss the limitations in Section 6.

   Guidelines:

   - The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory Assumptions and Proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

Justification: the paper does not include theoretical results.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental Result Reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: we will release code and data on the website for reproduction.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

https://doi.org/10.52202/079017-1069

Answer: [Yes]

Justification: we will release code and data on the website for reproduction.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://nips.cc/public/guides/CodeSubmissionPolicy) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental Setting/Details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: details can be found in Section 4, and Appendix A, B.2, and B.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment Statistical Significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [No]

   Justification: visual pretraining is compute-intensive, and we have limited compute.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.

- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments Compute Resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: details can be found in Appendix B.2, and B.1.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code Of Ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: we conform with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader Impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [Yes]

    Justification: we discuss broader impacts in Appendix D.

    Guidelines:

    - The answer NA means that there is no societal impact of the work performed.
    - If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to

https://doi.org/10.52202/079017-1069

generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: this work poses no such risks.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: we credit all authors in our paper and respect license and terms of use in our codebase.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New Assets**

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [Yes]

Justification: we will be releasing them on the website. All assets have been anonymized.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and Research with Human Subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: this paper does not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: this paper does not involve human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

https://doi.org/10.52202/079017-1069