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

Anonymous Author(s)
Affiliation
Address
email

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-anon.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 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 is to train the visual representations ‘in-domain’ on the demonstration data collected to solve the task [13, 4]. Prevalent approaches for using self-supervision in downstream control often make a bag-of-frames assumption, using contrastive methods [14, 15] 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 [16–23].

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,

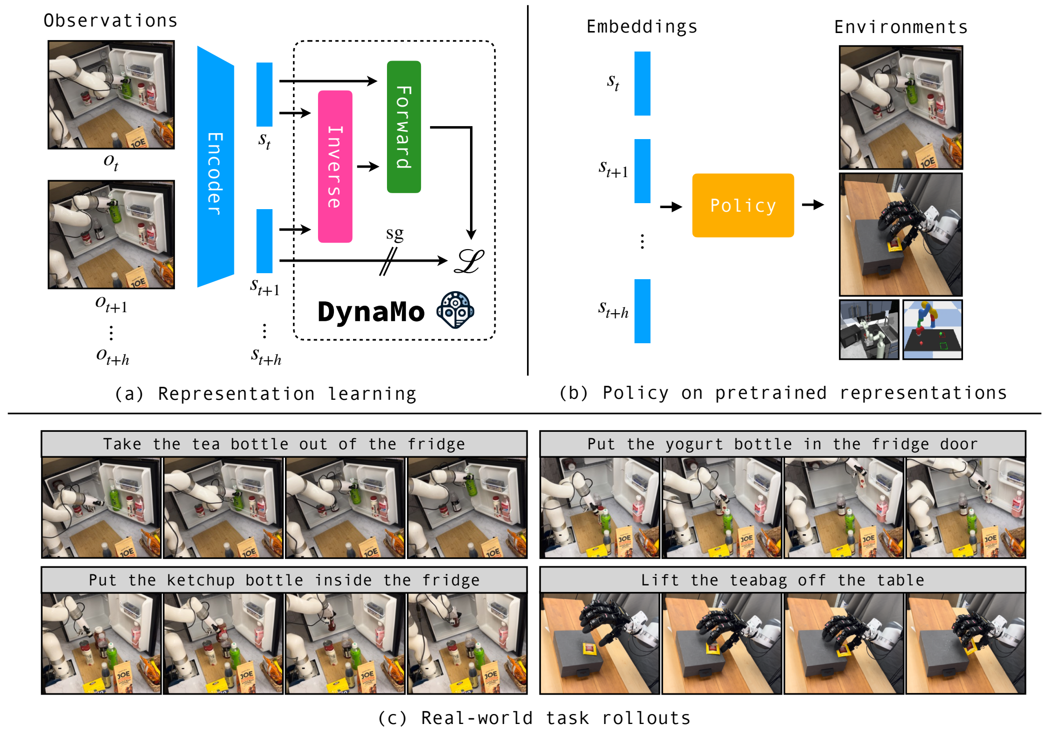

Observations
o_t
o_{t+1}
o_{t+h}
Encoder
s_t
s_{t+1}
s_{t+h}
Inverse
Forward
sg
$\mathscr{L}$
DynaMo
(a) Representation learning
Embeddings
s_t
s_{t+1}
s_{t+h}
Policy
Environments
(b) Policy on pretrained representations
Take the tea bottle out of the fridge
Put the yogurt bottle in the fridge door
Put the ketchup bottle inside the fridge
Lift the teabag off the table
(c) Real-world task rollouts

Figure 1: We present DynaMo, a new self-supervised method for learning visual representations for control.

without access to ground truth actions [24, 25]. 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-anon.github.io`.

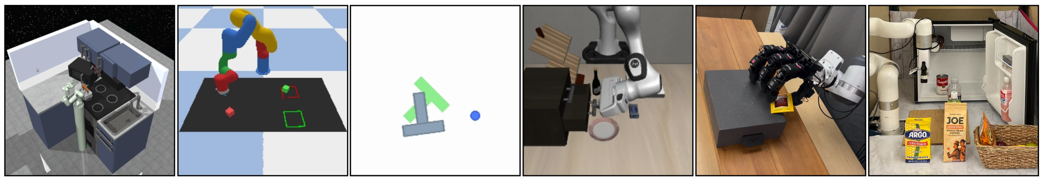
2 DynaMo

2.1 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 $p(\hat{s}_{1:h}|s_{:h-1}, z_{:h-1})$.

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. [26] on a minibatch of observation embeddings S:

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



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

Figure 2: 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.

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. [26] 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 [27] 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 $f_{\bar{\theta}}$ [28, 15], $s_t^* := \bar{s}_t = f_{\bar{\theta}}(o_t)$, 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.

3 Experiments

We evaluate our dynamics-pretrained visual representation on four simulated and two real robot environments, depicted in Figure 2. Details of them are included in Appendix A. We compare DynaMo representations with pretrained representations for vision and control, as well as other self-supervised learning methods.

3.1 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 [29] 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 D. Table 1 provides comparisons of DynaMo pretrained representations with other self-supervised learning methods, and pretrained weights for vision and robotic manipulation. Detailed descriptions of the baselines can be found in Appendix B. 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.

3.2 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 [30] 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 [29] 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

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	3.32	0.07	0.07	0.80
Pretrained representations	ImageNet	3.01	0.12	0.41	0.93
	R3M	2.84	0.11	0.49	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**

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**

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. Additionally, in Appendix C, we also show that DynaMo is compatible with various policy architectures, can be used to fine-tune other pretrained weights like ImageNet initialization, and that each component in DynaMo is necessary through an extensive ablation study.

References

[1] S. Lee, Y. Wang, H. Etukuru, H. J. Kim, N. M. M. Shafiullah, and L. Pinto. Behavior generation with latent actions. *arXiv preprint arXiv:2403.03181*, 2024.

[2] T. Z. Zhao, V. Kumar, S. Levine, and C. Finn. Learning fine-grained bimanual manipulation with low-cost hardware. *arXiv preprint arXiv:2304.13705*, 2023.

[3] C. Chi, S. Feng, Y. Du, Z. Xu, E. Cousineau, B. Burchfiel, and S. Song. Diffusion policy: Visuomotor policy learning via action diffusion. *arXiv preprint arXiv:2303.04137*, 2023.

[4] Z. J. Cui, Y. Wang, N. M. M. Shafiullah, and L. Pinto. From play to policy: Conditional behavior generation from uncurated robot data. *arXiv preprint arXiv:2210.10047*, 2022.

[5] N. M. Shafiullah, Z. Cui, A. A. Altanzaya, and L. Pinto. Behavior transformers: Cloning k modes with one stone. *Advances in neural information processing systems*, 35:22955–22968, 2022.

[6] A. S. Chen, S. Nair, and C. Finn. Learning generalizable robotic reward functions from" in-the-wild" human videos. *arXiv preprint arXiv:2103.16817*, 2021.

[7] Y. J. Ma, S. Sodhani, D. Jayaraman, O. Bastani, V. Kumar, and A. Zhang. Vip: Towards universal visual reward and representation via value-implicit pre-training. *arXiv preprint arXiv:2210.00030*, 2022.

[8] T. Xiao, I. Radosavovic, T. Darrell, and J. Malik. Masked visual pre-training for motor control. *arXiv preprint arXiv:2203.06173*, 2022.

[9] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. *arXiv preprint arXiv:2203.12601*, 2022.

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

[11] A. Majumdar, K. Yadav, S. Arnaud, J. Ma, C. Chen, S. Silwal, A. Jain, V.-P. Berges, T. Wu, J. 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.

[12] S. Dasari, M. K. Srirama, U. Jain, and A. Gupta. An unbiased look at datasets for visuo-motor pre-training. In *Conference on Robot Learning*, pages 1183–1198. PMLR, 2023.

[13] S. P. Arunachalam, I. Güzey, S. Chintala, and L. Pinto. Holo-dex: Teaching dexterity with immersive mixed reality. In *2023 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5962–5969. IEEE, 2023.

[14] X. Chen, S. Xie, and K. 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.

[15] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. 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.

[16] D. M. Wolpert, Z. Ghahramani, and M. I. Jordan. An internal model for sensorimotor integration. *Science*, 269(5232):1880–1882, 1995.

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

[18] 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.

[19] S. Kitazawa, T. Kimura, and P.-B. Yin. Cerebellar complex spikes encode both destinations and errors in arm movements. *Nature*, 392(6675):494–497, 1998.

[20] R. C. Miall and D. M. Wolpert. Forward models for physiological motor control. *Neural networks*, 9(8):1265–1279, 1996.

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

[22] J. R. Flanagan and A. 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.

[23] M. Haruno, D. M. Wolpert, and M. Kawato. Multiple paired forward-inverse models for human motor learning and control. *Advances in neural information processing systems*, 11, 1998.

[24] W. Whitney, R. Agarwal, K. Cho, and A. Gupta. Dynamics-aware embeddings. *arXiv preprint arXiv:1908.09357*, 2019.

[25] D. Brandfonbrener, O. Nachum, and J. Bruna. Inverse dynamics pretraining learns good representations for multitask imitation. *Advances in Neural Information Processing Systems*, 36, 2024.

[26] A. Bardes, J. Ponce, and Y. LeCun. Vicreg: Variance-invariance-covariance regularization for self-supervised learning. *arXiv preprint arXiv:2105.04906*, 2021.

[27] X. Chen and K. He. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 15750–15758, 2021.

[28] K. He, H. Fan, Y. Wu, S. Xie, and R. 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.

[29] S. Haldar, Z. Peng, and L. Pinto. Baku: An efficient transformer for multi-task policy learning. *arXiv preprint arXiv:2406.07539*, 2024.

[30] J. Pari, N. M. Shafiullah, S. P. Arunachalam, and L. Pinto. The surprising effectiveness of representation learning for visual imitation. *arXiv preprint arXiv:2112.01511*, 2021.

[31] A. Gupta, V. Kumar, C. Lynch, S. Levine, and K. Hausman. Relay policy learning: Solving long-horizon tasks via imitation and reinforcement learning. *arXiv preprint arXiv:1910.11956*, 2019.

[32] P. Florence, C. Lynch, A. Zeng, O. A. Ramirez, A. Wahid, L. Downs, A. Wong, J. Lee, I. Mordatch, and J. Tompson. Implicit behavioral cloning. In *Conference on Robot Learning*, pages 158–168. PMLR, 2022.

[33] B. Liu, Y. Zhu, C. Gao, Y. Feng, Q. Liu, Y. Zhu, and P. Stone. Libero: Benchmarking knowledge transfer for lifelong robot learning. *Advances in Neural Information Processing Systems*, 36, 2024.

[34] A. Iyer, Z. Peng, Y. Dai, I. Guzey, S. Haldar, S. Chintala, and L. Pinto. Open teach: A versatile teleoperation system for robotic manipulation, 2024.

[35] S. Young, J. Pari, P. Abbeel, and L. Pinto. Playful interactions for representation learning. In *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pages 992–999. IEEE, 2022.

[36] I. Radosavovic, B. Shi, L. Fu, K. Goldberg, T. Darrell, and J. Malik. Robot learning with sensorimotor pre-training. In *Conference on Robot Learning*, pages 683–693. PMLR, 2023.

[37] P. Sermanet, C. Lynch, Y. Chebotar, J. Hsu, E. Jang, S. Schaal, S. Levine, and G. Brain. Time-contrastive networks: Self-supervised learning from video. In *2018 IEEE international conference on robotics and automation (ICRA)*, pages 1134–1141. IEEE, 2018.

[38] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, and R. Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 16000–16009, 2022.

[39] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

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

[41] W. Xiang, H. Yang, D. Huang, and Y. Wang. Denoising diffusion autoencoders are unified self-supervised learners. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 15802–15812, 2023.

[42] V. Sterzentsenko, L. Saroglou, A. Chatzitofis, S. Thermos, N. Zioulis, A. Doumanoglou, D. Zarpalas, and P. Daras. Self-supervised deep depth denoising. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1242–1251, 2019.

[43] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[44] H. Bao, L. Dong, S. Piao, and F. Wei. Beit: Bert pre-training of image transformers. *arXiv preprint arXiv:2106.08254*, 2021.

[45] A. v. d. Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[46] M. Chen, A. Radford, R. Child, J. Wu, H. Jun, D. Luan, and I. Sutskever. Generative pretraining from pixels. In *International conference on machine learning*, pages 1691–1703. PMLR, 2020.

[47] A. Van Den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *International conference on machine learning*, pages 1747–1756. PMLR, 2016.

[48] T. H. Trinh, M.-T. Luong, and Q. V. Le. Selfie: Self-supervised pretraining for image embedding. *arXiv preprint arXiv:1906.02940*, 2019.

[49] M. Assran, Q. Duval, I. Misra, P. Bojanowski, P. Vincent, M. Rabbat, Y. LeCun, and N. 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.

[50] A. Bardes, Q. Garrido, J. Ponce, X. Chen, M. Rabbat, Y. LeCun, M. Assran, and N. Ballas. V-jepa: Latent video prediction for visual representation learning. 2023.

[51] N. Delson and H. 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.

[52] M. Kaiser and R. 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.

[53] S. Liu and H. 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.

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

[55] M. Reuss, M. Li, X. Jia, and R. Lioutikov. Goal-conditioned imitation learning using score-based diffusion policies. *arXiv preprint arXiv:2304.02532*, 2023.

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

[57] H. Von Helmholtz. *Handbuch der physiologischen Optik*, volume 9. Voss, 1867.

[58] A. M. Bastos, W. M. Usrey, R. A. Adams, G. R. Mangun, P. Fries, and K. J. Friston. Canonical microcircuits for predictive coding. *Neuron*, 76(4):695–711, 2012.

[59] L. F. Barrett and W. K. Simmons. Interoceptive predictions in the brain. *Nature reviews neuroscience*, 16(7):419–429, 2015.

[60] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[61] Y. Seo, K. Lee, S. L. James, and P. Abbeel. Reinforcement learning with action-free pre-training from videos. In *International Conference on Machine Learning*, pages 19561–19579. PMLR, 2022.

[62] K. Schmeckpeper, A. Xie, O. Rybkin, S. Tian, K. Daniilidis, S. Levine, and C. Finn. Learning predictive models from observation and interaction. In *European Conference on Computer Vision*, pages 708–725. Springer, 2020.

[63] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International conference on machine learning*, pages 2778–2787. PMLR, 2017.

[64] E. Shelhamer, P. Mahmoudieh, M. Argus, and T. Darrell. Loss is its own reward: Self-supervision for reinforcement learning. *arXiv preprint arXiv:1612.07307*, 2016.

[65] Z. Guo, S. Thakoor, M. Pîslar, B. Avila Pires, F. Altché, C. Tallec, A. Saade, D. Calandriello, J.-B. Grill, Y. Tang, et al. Byol-explore: Exploration by bootstrapped prediction. *Advances in neural information processing systems*, 35:31855–31870, 2022.

[66] J. Schrittwieser, I. Antonoglou, T. Hubert, K. Simonyan, L. Sifre, S. Schmitt, A. Guez, E. Lockhart, D. Hassabis, T. Graepel, et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839):604–609, 2020.

[67] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[68] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. Dream to control: Learning behaviors by latent imagination. *arXiv preprint arXiv:1912.01603*, 2019.

[69] D. Hafner, T. Lillicrap, M. Norouzi, and J. Ba. Mastering atari with discrete world models. *arXiv preprint arXiv:2010.02193*, 2020.

[70] R. Goroshin, J. Bruna, J. Tompson, D. Eigen, and Y. LeCun. Unsupervised learning of spatiotemporally coherent metrics. In *Proceedings of the IEEE international conference on computer vision*, pages 4086–4093, 2015.

[71] A. Bardes, Q. Garrido, J. Ponce, X. Chen, M. Rabbat, Y. LeCun, M. Assran, and N. Ballas. Revisiting feature prediction for learning visual representations from video. *arXiv preprint arXiv:2404.08471*, 2024.

[72] C. Feichtenhofer, H. Fan, B. Xiong, R. Girshick, and K. 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.

[73] X. Wang, A. Jabri, and A. 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.

[74] D. Dwibedi, Y. Aytar, J. Tompson, P. Sermanet, and A. Zisserman. Temporal cycle-consistency learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1801–1810, 2019.

[75] S. Pirk, M. Khansari, Y. Bai, C. Lynch, and P. Sermanet. Online object representations with contrastive learning. *arXiv preprint arXiv:1906.04312*, 2019.

[76] S. Bahl, A. Gupta, and D. Pathak. Human-to-robot imitation in the wild. *arXiv preprint arXiv:2207.09450*, 2022.

[77] P. Sharma, L. Mohan, L. Pinto, and A. Gupta. Multiple interactions made easy (mime): Large scale demonstrations data for imitation. In *Conference on robot learning*, pages 906–915. PMLR, 2018.

[78] B. Chen, P. Abbeel, and D. Pathak. Unsupervised learning of visual 3d keypoints for control. In *International Conference on Machine Learning*, pages 1539–1549. PMLR, 2021.

[79] Y. Qin, Y.-H. Wu, S. Liu, H. Jiang, R. Yang, Y. Fu, and X. Wang. Dexmv: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision*, pages 570–587. Springer, 2022.

[80] A. Sivakumar, K. Shaw, and D. Pathak. Robotic telekinesis: Learning a robotic hand imitator by watching humans on youtube. *arXiv preprint arXiv:2202.10448*, 2022.

[81] I. Kostrikov, D. Yarats, and R. Fergus. Image augmentation is all you need: Regularizing deep reinforcement learning from pixels. *arXiv preprint arXiv:2004.13649*, 2020.

[82] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *Advances in neural information processing systems*, 33:19884–19895, 2020.

[83] I. Radosavovic, T. Xiao, S. James, P. Abbeel, J. Malik, and T. Darrell. Real-world robot learning with masked visual pre-training. In *Conference on Robot Learning*, pages 416–426. PMLR, 2023.

[84] A. Padalkar, A. Pooley, A. Jain, A. Bewley, A. Herzog, A. Irpan, A. Khazatsky, A. Rai, A. Singh, A. Brohan, et al. Open x-embodiment: Robotic learning datasets and rt-x models. *arXiv preprint arXiv:2310.08864*, 2023.

[85] N. M. M. Shafiullah, A. Rai, H. Etukuru, Y. Liu, I. Misra, S. Chintala, and L. Pinto. On bringing robots home. *arXiv preprint arXiv:2311.16098*, 2023.

[86] G. Zhou, V. Dean, M. K. Srirama, A. Rajeswaran, J. Pari, K. Hatch, A. Jain, T. Yu, P. Abbeel, L. 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.

[87] I. Guzey, B. Evans, S. Chintala, and L. Pinto. Dexterity from touch: Self-supervised pre-training of tactile representations with robotic play. *arXiv preprint arXiv:2303.12076*, 2023.

A Environment and dataset details

A.1 Franka Kitchen

The Franka Kitchen [31] 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).

A.2 Block Pushing

The simulated Block Pushing environment [32] 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). 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

The Push-T environment [3] 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). 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

The LIBERO Goal [33] 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 goal-conditioned policy for 100 rollouts in total, 10 per task goal, and report the average success rate (maximum 1).

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.

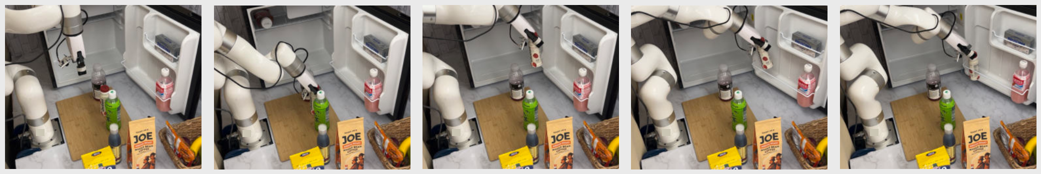
Sponge picking: This task requires the hand to reach to the position of the sponge, grasp the sponge, and lift the sponge from the table. We collect 6 demonstrations via OpenTeach [34] for the task, starting from different positions, with 543 frames in total. The task is considered successful if the robot hand can grasp the sponge from the table within 120 seconds.

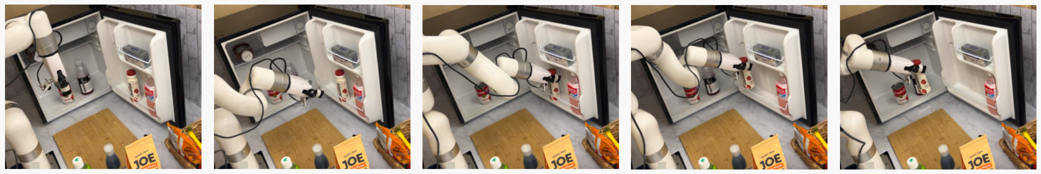
Teabag picking: This task is similar to the previous task, but more difficult with a smaller task object. We collect 7 demonstrations via OpenTeach with 1 034 frames in total. In this task, the robot needs reach the teabag, grasp the teabag with two fingers, then pick it up. The task is considered successful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

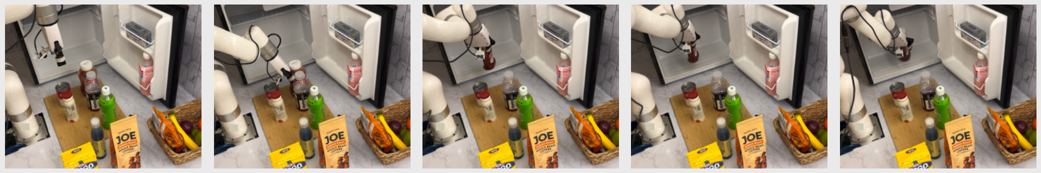
Microwave opening: This task requires the hand to reach the microwave door handle, grasp the handle, and pull down the door. We collect 6 demonstrations via OpenTeach with 735 frames in total. The task is considered successful if the robot hand can open the door within 240 seconds.

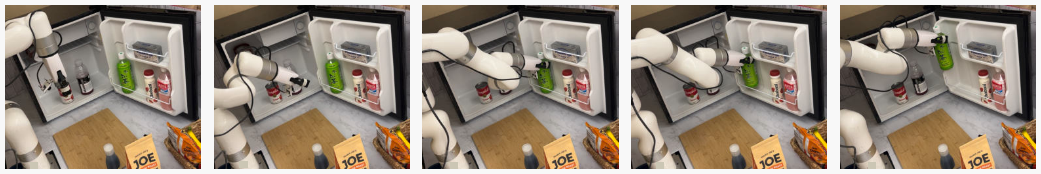
A.6 xArm Kitchen

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 comprises the robot end effector pose and the gripper state. We collect a total of 65 demonstrations across 5 tasks, depicted in Figure 3. The demonstrations were collected using OpenTeach [34] at 30Hz. The learned policies are deployed at 10Hz. Figure 3 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.

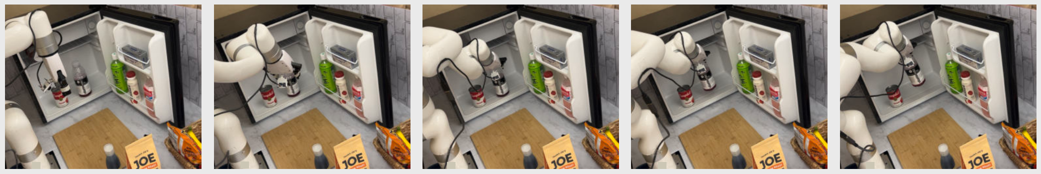

Fetch water bottle from fridge: Take the bottle of vitamin water out of the fridge.

Figure 3: xArm Kitchen environment tasks

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**

B Baseline details

- **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 [15] pretraining on demonstration data.
- **BYOL-T**: BYOL + temporal contrast [35]. Adjacent frames o_t, o_{t+1} are sampled as positive pairs, in addition to augmentations.
- **MoCo-v3**: MoCo [28] pretraining on demonstration data.
- **RPT**: RPT [36] trained on observation tokens.
- **TCN**: Time-contrastive network [37] pretraining on demonstrations. MV: multi-view objective; SV: single view objective.
- **MAE**: Masked autoencoder [38] pretraining on demonstrations.
- **DynaMo**: DynaMo pretraining on demonstrations.

C Additional experiments

C.1 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 [30]. We present the results in Table 4. We find that DynaMo representations 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 §3.2 with a nearest neighbor policy.

C.2 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 §3.1. 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

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 (·/4)	Block Pushing (·/2)	Push-T (·/1)	LIBERO Goal (·/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

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%

performance. We also note that our method works in the low-data regime with ImageNet initialization on the real Allegro hand in Table 2.

C.3 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.

Inverse dynamics to a transition latent: As described in §2.1, 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. [26] 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 [28, 15] and stop gradient [27] leads to representation collapse [39, 15, 26].

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 D.1.

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%

C.4 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. [25], 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.

D Hyperparameters and implementation details

D.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

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 [40] 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

D.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

E Real robot environment rollouts

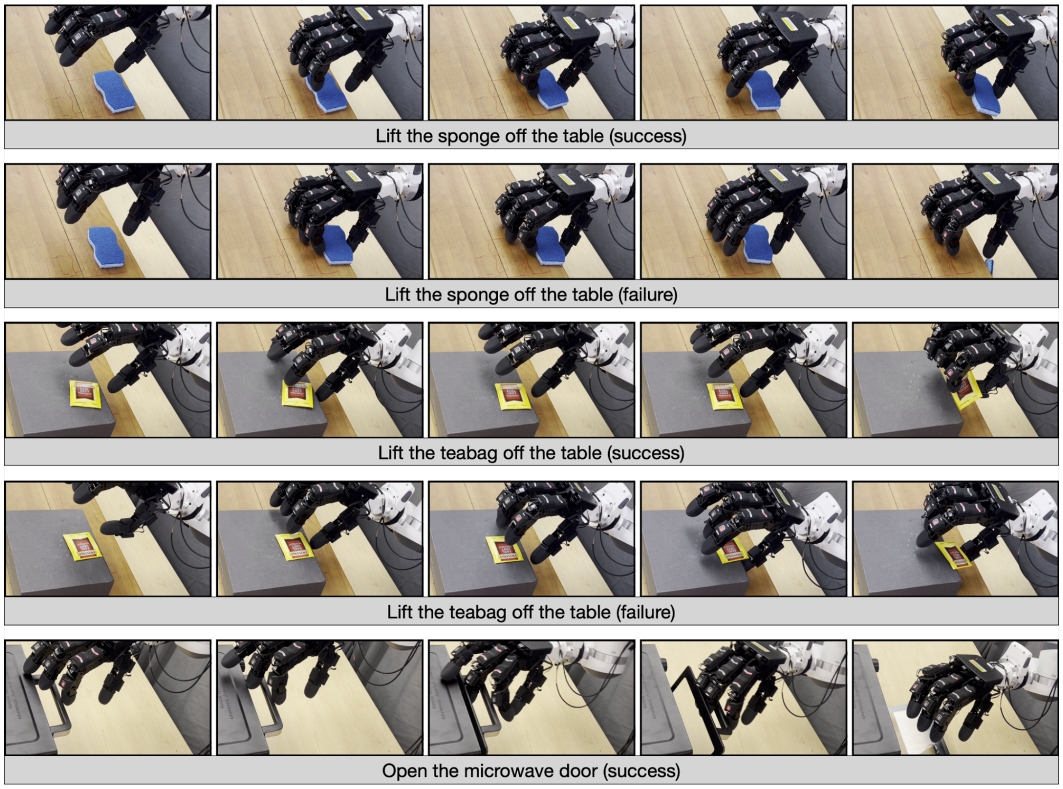

Lift the sponge off the table (success)
Lift the sponge off the table (failure)
Lift the teabag off the table (success)
Lift the teabag off the table (failure)
Open the microwave door (success)

Figure 4: Rollouts on Allegro Manipulation with our DynaMo-pretrained encoder.

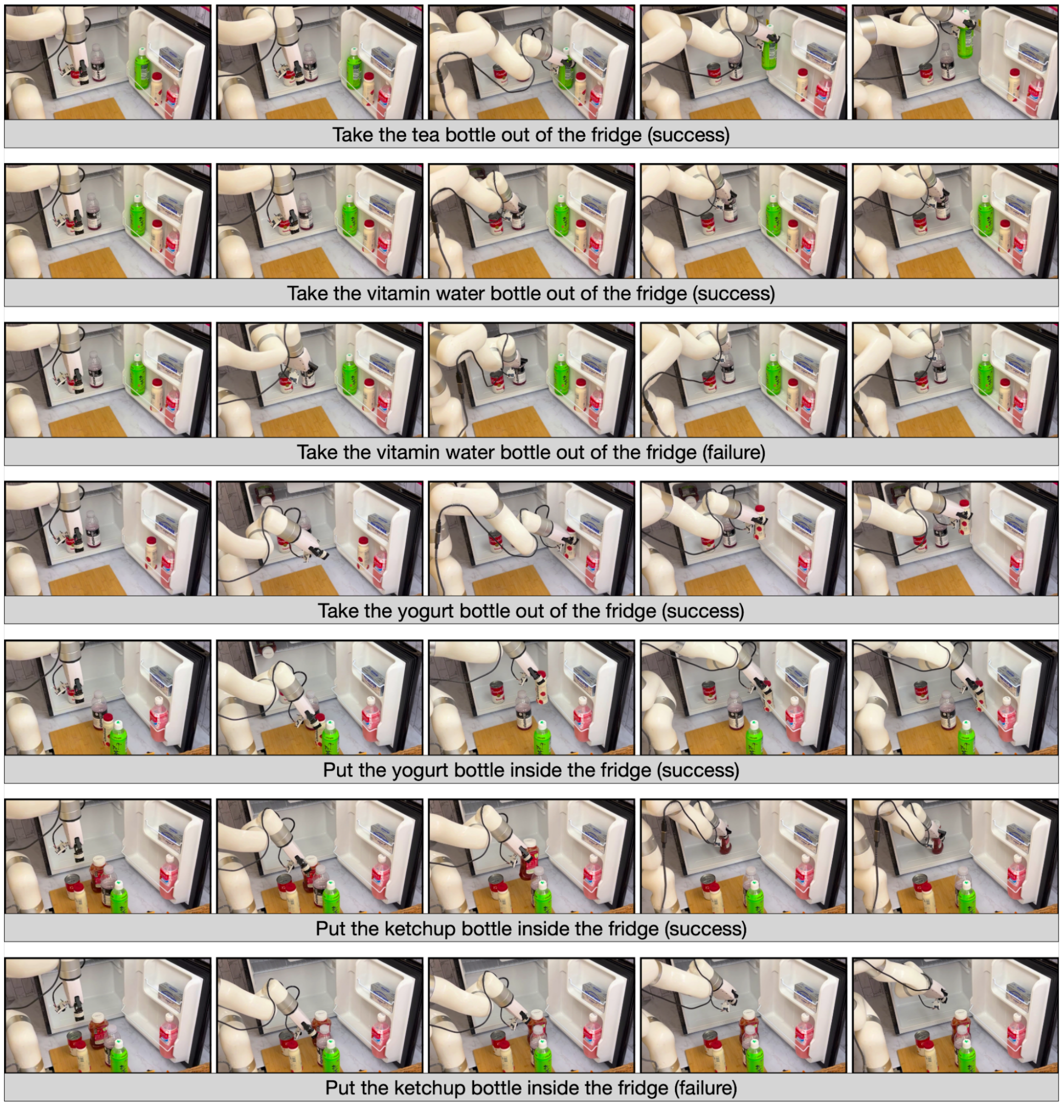

Take the tea bottle out of the fridge (success)
Take the vitamin water bottle out of the fridge (success)
Take the vitamin water bottle out of the fridge (failure)
Take the yogurt bottle out of the fridge (success)
Put the yogurt bottle inside the fridge (success)
Put the ketchup bottle inside the fridge (success)
Put the ketchup bottle inside the fridge (failure)

Figure 5: Rollouts on xArm Kitchen with our DynaMo-pretrained encoder.

F Background

F.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 $f_\theta : o_t \to s_t$ 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.

F.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, 38]). A third approach is contrastive learning with temporal proximity as supervision, pushing temporally close frames to have similar embeddings [37, 35].

G 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.

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 [41, 42] and masked modeling [43, 44, 38]. Another approach is contrastive learning, which leverages data augmentation priors [39, 15, 28, 26, 27] or temporal proximity [37, 45] to produce contrastive sample pairs. A third self-supervised method is generative modeling [46–48], 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 [49, 50].

Learning from demonstrations: Learning from human demonstrations is a well-established idea in robotics [51–54]. With the advances in deep learning, recent works such as [3, 2, 5, 4, 1, 55] 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 [56–59]. Early works such as [16–19] propose that there exists an internal model of the motor apparatus in the cerebellum for motor control and planning. [20, 21] 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 [22, 23].

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 [60], reinforcement learning [61], and representation learning [45, 62]. Incorporating the prediction of future states as an intrinsic reward has also been shown to improve reinforcement learning performance [63–65]. Moreover, recent work demonstrates that world models trained to predict environment dynamics can enable planning in complex tasks and environments [66–69].

Learning from video for control: Videos provide rich spatiotemporal information that can be leveraged for self-supervised representation learning [70–75]. 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 [76–80].

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 [81, 82]. 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, 83, 8, 84]. More recent work has shown that in-domain self-supervised pretraining improves policy performance [85–87] and enables non-parametric downstream policies [30].