SpawnNet: Learning Generalizable Visuomotor Skills from Pre-trained Networks

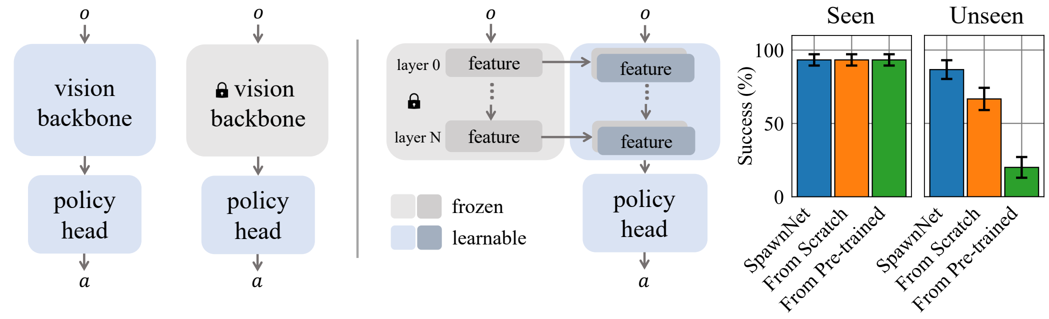
Xingyu Lin* **John So*** **Sashwat Mahalingam** **Fangchen Liu** **Pieter Abbeel**

UC Berkeley

*Equal Contribution

Abstract: The existing internet-scale image and video datasets cover a wide range of everyday objects and tasks, bringing the potential of learning policies that have broad generalization. Prior works have explored visual pre-training with different self-supervised objectives, but the generalization capabilities of the learned policies remain relatively unknown. In this work, we take the first step towards this challenge, focusing on how pre-trained representations can help the generalization of the learned policies. We first identify the key bottleneck in using a frozen pre-trained visual backbone for policy learning. We then propose SpawnNet, a novel two-stream architecture that learns to fuse pre-trained multi-layer representations into a separate network to learn a robust policy. Through extensive simulated and real experiments, we demonstrate significantly better categorical generalization compared to prior approaches in imitation learning settings. Videos can be found on our website: https://xingyu-lin.github.io/spawnnet/.

Keywords: Visual Pre-training, Generalizable Robotic Manipulation



(a) From scratch **(b) From pre-trained** **(c) SpawnNet** **(d) Imitation learning results**

Figure 1: Prior approaches for learning policies (a) from scratch, (b) from a pre-trained visual representation with a frozen backbone, and (c) the proposed two-stream architecture. The right figure (d) shows their performances on a real-world imitation learning task, evaluated on both seen and unseen instances in a category.

1 Introduction

To take steps towards deploying robots in our daily life, learning skills that can handle diverse situations is crucial for robots to subsist with us in the real world. For example, consider a scenario where we want our robot to help us wear hats. We can teach this skill by demonstrating with the hats we currently have. However, we don't want to recollect demonstrations once we buy a new hat or change the position of our coat rack. Thus, we hope to enable our robot with a smart hat-wearing policy that can handle object variations in color, pose and shape.

To this end, we need to equip manipulation skills with semantic knowledge of the world, which already lies in current internet-scale video and image datasets [1, 2, 3]. Given the success of visual representation learning [4, 5, 6, 7, 8, 9] on a wide range of computer vision tasks, a natural solution to our goal is to take the pre-trained representations out of the box, and train robust visuomotor skills on top of it. Prior works have explored this direction [10, 11, 12, 13], with promising results on improved sample efficiency and asymptotic performance. However, most evaluations in prior works are done on simple tasks with relatively small variations and thus are not sufficiently difficult. Indeed, recent work [14] has shown that a carefully done baseline that learns from scratch can often be very competitive with these prior works that use pre-trained networks.

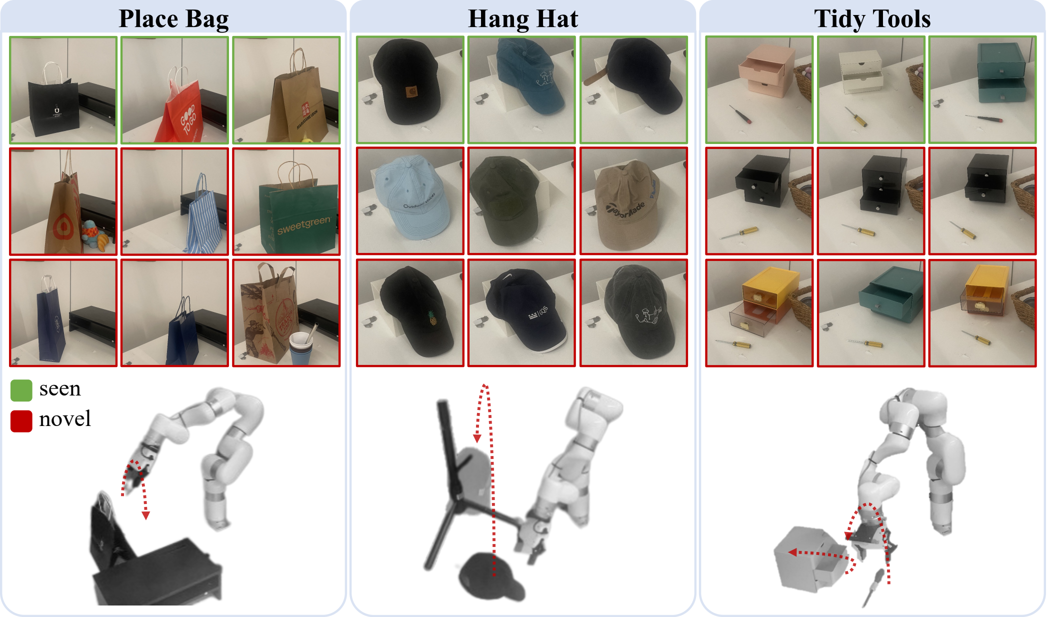


Figure 2: We consider three challenging categorical manipulation tasks in the real world. For each task, we train on three instances (green boxes) and test on held-out instances (red boxes), with additional variations in poses, articulation, visual distraction, and deformation.

We hypothesize that the benefits of pre-training will be more pronounced when considering generalization performance. Hence, we create three challenging categorical manipulation tasks (depicted in Figure 2) and study the utility of pre-trained visual representations there. Our tasks contain a set of diverse objects with no overlapping instances between training and evaluation. Through these challenging tasks, we identify the bottleneck in the existing methods: frozen pre-trained networks give fixed representations to the policy and as such, may hinder policy learning without adapting the visual backbone. The need for adaptation may come from the differences between the pre-training and policy learning objectives, or from the distribution shift from the pre-training dataset to robotic demonstrations, including domains, tasks, or camera viewpoints [15].

Given this observation, we propose a simple two-stream architecture, SpawnNet, to learn a generalizable visuomotor policy from a pre-trained neural network. Instead of directly using frozen representations, we adopt a separate network that learns to fuse multi-layer pre-trained representations (Figure 1) and generates actions. The learnable stream incorporates domain-specific features from the raw observations to help handle distribution shifts. Meanwhile, it can take advantage of the pre-trained features at different layers for faster learning and better generalization. Through extensive experiments, we demonstrate that SpawnNet significantly outperforms other pre-training methods and a learning-from-scratch baseline when tested on held-out novel objects. Below we highlight the contribution of this paper:

1. We propose SpawnNet, a novel, effective, and flexible framework that can adapt any pre-trained model to a generalizable visuomotor policy on various downstream tasks.

2. We perform systematic evaluations of different methods utilizing pre-trained representation both in simulation and in the real world, showing significant improvement of our method in cross-instance generalization.
3. To the best of our knowledge, our work is the first demonstration of using a pre-trained visual representation for better categorical generalization in a visuomotor learning framework.

2 Related Works

Self-supervised Visual Representation Learning. Self-supervised learning is a scalable learning paradigm without requiring the ground-truth label of data. This enables learning representation from internet-scale data and thus achieves broad generalization [3, 16]. Prior works use different objectives to learn transferrable representations, including contrastive learning [7, 4, 16, 17], siamese similarity [9, 5, 8], and masked self-reconstruction [6, 18]. Instead of focusing on self-supervised training objectives, our method adopts a pre-trained representation out of the box and aims to leverage its generalization in policy learning.

Representation Learning for Control. Self-supervised learning objectives can be easily coupled with policy learning, serving as a plug-in tool for training visual control systems. Prior works combine self-supervised learning [19, 20, 21, 22, 23] jointly with policy training, which often yields better final performance and sample efficiency. Different from their approach, our method focuses more on leveraging the pre-trained representation from powerful large transformers and internet-scale datasets, without performing self-supervised learning from scratch.

Visual Pre-training for End-to-end Policy Learning. Recently there has been a surge of works that aim to pre-train a visual representation on image or video datasets, using ResNet [24, 25, 11] or Vision Transformer (ViT) [10, 26] as the backbone. After training, these works freeze the visual representation for downstream policy learning. It has been shown that pre-trained representation can achieve better sample efficiency or asymptotic performance. However, most evaluations in prior works are done in simple simulation environments or real-world tasks with relatively simple variations. As such, a recent study shows that strong learning-from-scratch baselines with data augmentations can achieve comparable performance to the best pre-trained representations [14]. In contrast, our work sheds light on the generalization capabilities of pre-trained representations on a rich set of assets and tasks in both simulation and real-world experiments through a novel neural architecture that better utilizes the pre-trained representation.

3 Method

Pre-trained visual representations are trained on large image and video datasets and have the potential to help learn generalizable visuomotor skills. We will first give a background on the vision transformer, the pre-trained vision backbone architecture we use in Section 3.1. Then in Section 3.2, we explain the issue with the current way of using the pre-trained features and present our architecture. Finally, we explain our policy learning frameworks for evaluation in Section 3.3.

3.1 Vision Transformer Preliminary

While our method can be combined with any pre-trained visual architecture, we will mainly discuss its instantiation with pre-trained vision transformers [27] as they are more scalable and commonly used for visual pre-training. The input RGB images are first split into patches. Each patch is encoded into a latent vector representation named a token. Additionally, a learned CLS token is concatenated to the other image tokens, and the sequence is passed through multiple transformer layers. Each transformer layer consists of alternating layers of multi-headed self-attention (MHSA) and MLPs, with residual layers between them. Within the MHSA block, an input X is first split, then projected onto learned key, query, and value bases before the attention mechanism: $MHSA(X) = softmax(QK^T/\sqrt{d})V$, where Q, K, V are linear projections of X.

Prior works in using the pre-trained representation for policy learning usually take the CLS token at the last layer as the image representation. On the other hand, people have found the attention features in MHSA in intermediate layers of the vision transformer to encode semantic information at the object-level and part-level [28, 29]. As such, we follow Amir et al. [28] to extract the key K in MHSA from a pre-trained vision transformer DINO [17]. Since vision transformers usually use a large model and require large computation to fine-tune, throughout this paper we freeze the weights of the pre-trained network.

3.2 SpawnNet Architecture

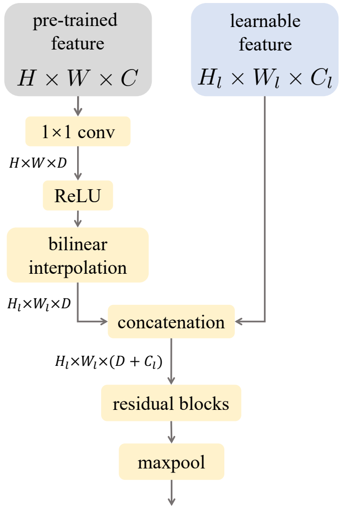


Figure 3: Adapter layers to fuse the pre-trained features with the learnable features.

Given these powerful pre-trained vision transformers, how do we utilize them for policy learning? Prior approaches extract the CLS token at the last layer as the image representation, as illustrated in Figure 1. The representation is typically frozen as fine-tuning vision transformers requires large computation. However, this architecture lacks flexibility in adapting the visual features. This potentially hinders downstream policy learning for two reasons. First, the self-supervised objectives in pre-training may not align with the policy learning objective. Second, datasets used for pre-training can have a different distribution from downstream tasks in domains, tasks, or camera viewpoints [15].

To learn more flexible task-specific representations while taking advantage of the pre-trained features, SpawnNet trains a separate stream of convolutional neural networks (CNN) from scratch, taking the raw observation as input (Figure 1). At the same time, we design adapter modules to fuse the new stream with the pre-trained features at different layers, taking advantage of the robust features as needed. We find that a shallow CNN for the new stream works well under the low-data regime during policy learning. Additionally, this architecture allows us to incorporate different modalities like depth in the separate stream, which is not in the pre-trained vision networks.

We first extract spatially dense descriptors from different layers of the pre-trained networks. At the l^{th} layer of the vision transformer, we extract the key feature in MHSA from each token, forming a feature grid of $\phi^l(I) \in \mathbb{R}^{H \times W \times C}$, where $H \times W$ is the number of image patches and C is the feature dimension. Extracting features from different layers allows us to extract both low-level and high-level visual features from the pre-trained network [28]. Note that vision transformers keep the same number of image patches at each layer; extracting features from each image patch allows us to maximally preserve spatial information.

Adapter Layers. We fuse the features from the two streams of networks at different layers as shown in Figure 3 using *adapter layers*. The adapter first maps the pre-trained feature into a latent embedding of size D through a convolutional kernel of size 1 and stride 1. It then resizes the feature from $H \times W$ to $H_l \times W_l$ with bi-linear interpolation and finally concatenates them in the feature dimension. It is then passed through two residual blocks to be processed in the next layers.

3.3 Visuomotor Policy Learning

We study the effect of visual representation in learning challenging visuomotor control tasks. For all our tasks, we consider large intra-category variations and hold out a portion of instances during training to study generalization.

Learning with Expert Guidance in Simulation. Prior works study the visual representations under the Reinforcement Learning (RL) or Behaviour Cloning (BC) framework. This couples the visual representation with challenges in exploration or covariate shift. For our simulation experiments, we first train RL policies on all training instances using PPO [30]. We treat the RL policies as experts and learn image-based policies using DAgger [31]: Within each iteration, we first roll out the current policy in the environments and then query the experts on agent's visited states to get the corresponding action labels for training. We then train the current policy with gradient descent updates to minimize the MSE loss between the agent's actions and the expert's actions.

Behavior Cloning with Teleoperation Demonstration in the Real World. While the simulated tasks allow us to iterate quickly, the rendered images are not photo-realistic, which creates an artificial gap to the real data used for pre-training. As such, we further train and evaluate visuomotor policies in the real world. For each task, we collect demonstrations from humans using a teleoperation setup. Since it is difficult to query humans in robot-visited states in the real world, we simply train different policies using behavior cloning.

4 Experiments

We design experiments in both simulation and in the real world to answer the following questions:

1. How does SpawnNet compare with other ways of using a pre-trained representation?
2. How do different pre-trained visual representations affect training and more importantly, the performance of generalization to novel instances?
3. How much do different components of SpawnNet help with the final performance?

We note that experiments in prior works Radosavovic et al. [10], Nair et al. [11], Hansen et al. [14] have only shown limited variations in poses or instances within a category. We aim to stress-test the generalization capabilities of the policies through significant in-category variations where a policy would break if not well adapted to each instance.

Baselines. We compare with the following baselines:

- **LfS+aug (learning from scratch) [14]** is a shallow ConvNet encoder followed by an MLP policy with data augmentation, which has been shown to be a strong baseline. Following Hansen et al. [14], we use random shift as our data augmentation.
- **MVP [10]** trains a masked auto-encoder from ego-centric video data and takes the frozen CLS token from the vision transformer as the representation.
- **R3M [11]** trains a language-aligned visual representation from videos using time contrastive learning and language-video alignment.

All methods with only a frozen pre-trained backbone (i.e. **MVP**, **R3M**) take RGB as input. Meanwhile, incorporating depth is straightforward for **SpawnNet** and **LfS** by adding a depth channel to the trainable encoder. We denote these depth-conditioned variants as **SpawnNet+d** and **LfS+aug+d**.

4.1 Simulation Experiments

Tasks. We conduct experiments on two tasks in simulation, opening different cabinet doors and drawers with a Franka arm, as shown in Figure 4. For each task, we hold out a subset of the objects during training and use them for evaluating the generalization of the learned policies on novel objects. Our tasks are taken from RLAfford [32], built on top of IsaacGym [33]. The agents receive image observations from three camera views (left, middle, right) and the proprioception as input, and output joint positions for the arm.

Training and Evaluation. We first take the pre-trained RL policies from Geng et al. [32] using PPO [30], which takes point clouds as input. We then train different policies with DAgger to imitate the RL experts. This helps us decouple the visual representation from other difficulties such as the covariant shift in behavior cloning. We alternate model training and agent rollouts and stop after a fixed number of trajectories. For all methods, we select 21 instances from the training set where the experts perform well and use them for training DAgger policies. We evaluate policies on 8 held-out instances for Open Door and 12 held-out instances for Open Drawer. We roll out the agent for 5 trajectories per asset to evaluate each model. For each method, we train three models with different seeds. We report the average performance and the standard error across training and novel instances.

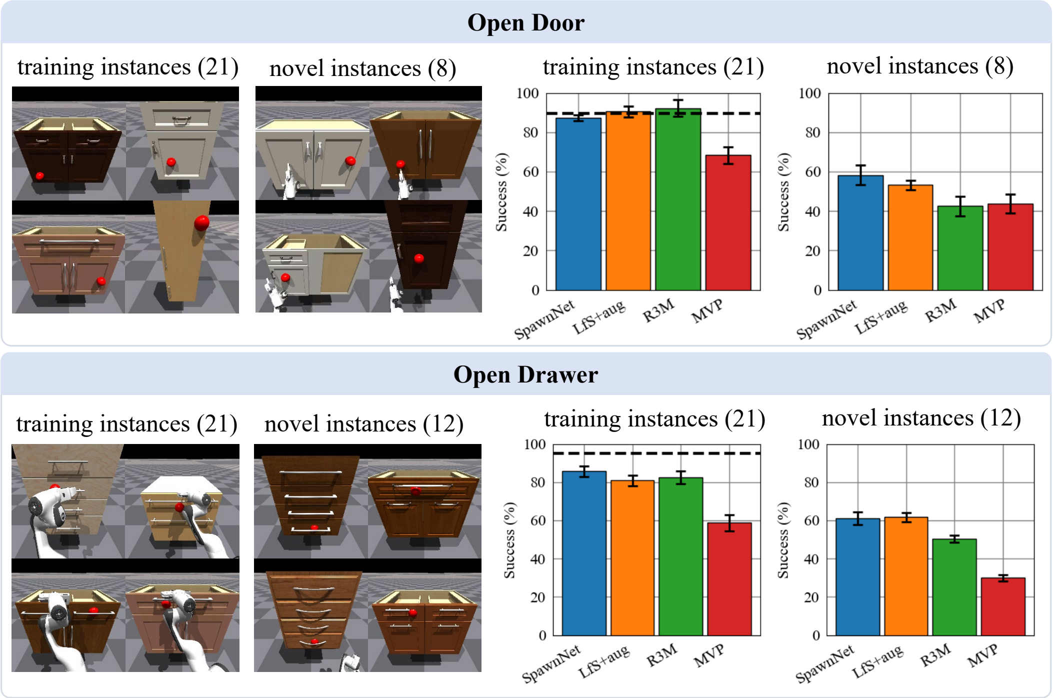


Figure 4: Simulation results on Open Door and Open Drawers. The left figures show the training and novel instances we use. The observations are rendered from the agent's middle camera. We add red spheres in the scene to specify the task of which door/drawer to open. The right shows success rates of different methods in both seen and unseen instances after a fixed number of agent rollouts. The dashed black line shows the RL expert's performance. The error bars show the standard error computed from three random seeds. Numbers in the brackets denote the number of instances.

The results are shown in Figure 4. We find that different approaches perform similarly on training instances, where the learning-from-scratch baseline can also perform competitively. Most methods are able to achieve expert performance. However, on novel instances, SpawnNet generalizes significantly better than pre-trained baselines, showing the benefit of our method in utilizing pre-trained features. LfS+aug proves to be a strong baseline in simulation. We next evaluate in a real world setting, where visual pre-training should comparatively benefit from realistic image observations and limited training instances.

4.2 Real World Experiments

Experimental Setup. We evaluate methods using a real-world robotics setup with a UFACTORY xArm 7. Humans provide demonstrations using a 3Dconnexion Spacemouse; actions are parameterized and collected as 6-DOF, delta end-effector control $(\Delta x, \Delta y, \Delta z, \Delta\alpha, \Delta\beta, \Delta\gamma)$ at 5Hz, plus gripper open/close. Both translation and rotation in the action space are defined in the wrist camera view to enable better generalization. Each demonstration collects observations as RGB and depth images from two RealSense cameras: one from third-person view, and one attached to the wrist of the robot arm. We stack the most recent four frames as the agents' observations. We do not give the agents access to proprioception information as we find that the agent tends to overfit to the proprioception and ignore the visual input, which hinders generalization.

For real-world comparison, we evaluate against **R3M** and **LfS+aug+d** as they perform well in simulation tasks.

Tasks. We evaluate methods with behavior cloning for three manipulation tasks. Each real-world task consists of around 90 demonstrations across 3 training instances with pose variations and a set of held-out objects. The tasks and variations are illustrated in Figure 2 and summarized below:

1. **Place Bag**: Lift up a bag by the strap, and place it on a table. The bag's pose is varied.
2. **Hang Hat**: Pick up a cap, and hang it on a rack. Caps' poses slightly vary across runs.
3. **Tidy Tools**: Pick up a handled tool, place it in an open drawer, and close the drawer. We provide different tools, different initial tool poses, different drawers, and vary which drawers to close.

The combination of geometrically and visually diverse instances variation makes both learning manipulation skills and generalizing to new instances challenging.

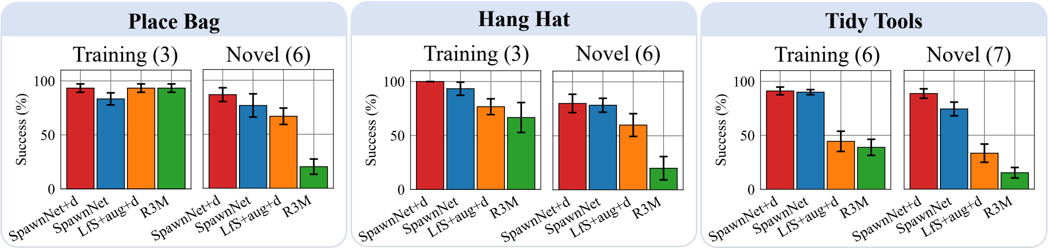


Figure 5: Real-world manipulation results on three tasks. All methods here use the same data augmentation. We evaluate each method on each instance over 5 trials (more than 30 trials on novel instances). We report the mean and standard error.

Results. The results on training and held-out instances are shown in Figure 5. Numeric values and more details can be found in the appendix. We observe that SpawnNet performs much better than baselines in both training and unseen instances; in unseen instances, the overall gap over the baselines becomes even larger than in simulation. We conjecture that this is because real-world images follow a closer distribution to the pre-training dataset compared to simulation. While LfS and R3M perform well on training occasionally, they are much worse at generalizing. For tasks with larger instance variations on training, such as Tidy Tools, both LfS and R3M fail to achieve 50% success rates even on training while SpawnNet achieves greater than 80% success. We hypothesize that this is because SpawnNet can better capture semantic variance across demonstrations on different instances through dense features.

Additionally, we show that by adding depth, SpawnNet+d improves by as much as 15% over SpawnNet on novel instances. This shows the flexibility of SpawnNet in incorporating different modalities.

4.3 What does SpawnNet learn from the pre-trained network?

We provide more insights into how SpawnNet's use of pre-trained features can help policy learning; to do so, we take a trained policy and visualize the norm of the features after the 1x1 convolutions and the nonlinear layers in the last layer of the adapter, as shown in Figure 6. We see that important parts of the bags are highlighted, such as the handles of the bags when picking and the bottom of the bags when placing. These highlighted regions are consistent across time and across instances. They also generalize to unseen instances. As such, we believe that fusing the pre-trained network's layers enables better learning and generalization. Please refer to Appendix D.2 for more visualizations.

4.4 Ablations

We ablate SpawnNet's architecture using two tasks, Open Door (sim) and Place Bag (real):

- **Remove pre-trained features (-Pre-trained)**: To study how much we gain from the pre-trained representation, we zero-mask all pre-trained features.
- **Last pre-trained layer only (-Multiple)**: To study the importance of features from multiple layers, we only adapt the last pre-trained layer's features.

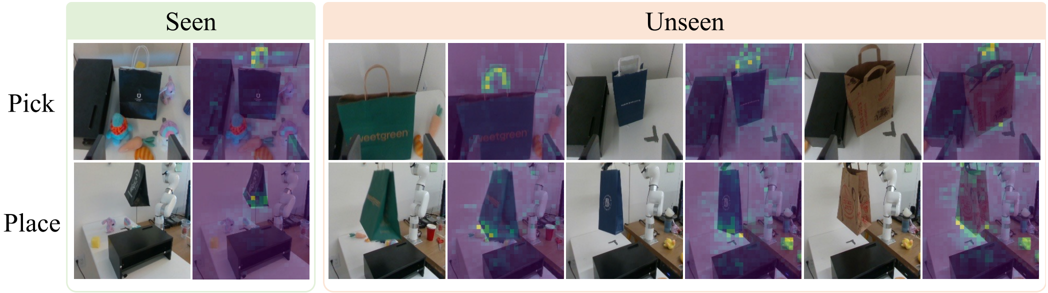


Figure 6: Visualization of the attention on the pre-trained features by the last adapter during the *Place Bag* task. The attention focuses on important regions that are consistent across even unseen instances: the handles of the bags are highlighted when picking the bag and the bottom of the bags are highlighted when placing.

- **CLS token only (-Dense)**: To study the importance of dense features, we replace pre-trained layers' dense features [HxWxC] with the CLS token [1xC] tiled to the same dimensions.

Results are shown in Table 1; **Full Method** denotes SpawnNet for Open Door, and SpawnNet+d for Place Bag. Notably, removing dense features impacts performance as much as removing pre-trained features entirely, suggesting the importance of dense features for generalization. Adapting only the last pre-trained layer results in a slight decrease in performance, suggesting that spatial information from multiple layers is also helpful.

Method	Open Door	Place Bag
Full Method	59.6	86.7
-Pre-trained	52.9 (-6.7)	-
-Multiple	58.3 (-1.3)	73.3 (-13.3)
-Dense	52.5 (-7.1)	61.7 (-24.9)

Table 1: Ablations on Open Door and Place Bag; numbers in red indicate performance decrease compared to the full method. Removing dense features has the most impact.

4.5 SpawnNet with other pre-trained networks

To determine whether our performance is because of the quality of DINO features, we additionally experiment with initializing SpawnNet from other pre-trained backbones (**SpawnDINO**, **SpawnMVP**, and **SpawnR3M**), and compare them to pre-trained networks without SpawnNet (**DINO**, **MVP**, and **R3M**). These models vary broadly in terms of dataset, training objectives, and architecture. Again, we evaluate using the simulated Open Door and real Place Bag tasks. The results, reported in Table 2, suggest that SpawnNet is a general architecture which can improve the performance of any pre-trained network.

Method	Open Door	Place Bag
DINO	44.6	11.7
R3M	42.5	20.0
MVP	43.8	-
SpawnDINO	59.6 (+15.0)	86.7 (+75.0)
SpawnR3M	49.2 (+6.7)	61.7 (+41.7)
SpawnMVP	49.2 (+16.2)	-

Table 2: Performance on the Open Door and Place Bag tasks with different pre-trained networks. The numbers in green show improvement over the pre-trained network. Using SpawnNet improves the performance of all pre-trained representations.

5 Limitations and Conclusions

Limitations Our experiments mainly study policy generalization under imitation learning settings, which requires the action distributions to be similar in training and evaluation tasks. For this reason, we only evaluate the policy on novel instances, without heavily extrapolating the pose of instances (see Appendix C for details), as an out-of-distribution pose can cause a drastic change of motion. However, this can be addressed in an interactive learning setting. We leave the combination of SpawnNet and reinforcement learning algorithms to future work.

Conclusions In this paper, we propose a novel and effective architecture to take advantage of pre-trained representations beyond simply freezing the representation. We show that SpawnNet's use of the pre-trained representation not only improves performance on training instances, but more importantly, offers better generalization to novel instances when compared to competitive learning-from-scratch and pre-trained baselines. Through systematic evaluation, we hope our paper can convince more people of the benefits of using pre-trained visual representations and boost further progress.

Acknowledgments

We would like to thank Youngwoon Lee, Xingyang Geng, Boyi Li, Qi Dou, and Jason Anderson for the fruitful discussions. We also thank Olivia Watkins, Yuqing Du and Oleh Rybkin for their feedback on the initial draft of the paper. This work was supported in part by NSF under the AI4OPT AI Center, the BAIR Industrial Consortium, and the InnoHK of the Government of the Hong Kong Special Administrative Region via the Hong Kong Centre for Logistics Robotics.

References

[1] D. Damen, H. Doughty, G. M. Farinella, S. Fidler, A. Furnari, E. Kazakos, D. Moltisanti, J. Munro, T. Perrett, W. Price, et al. Scaling egocentric vision: The epic-kitchens dataset. In *European Conference on Computer Vision (ECCV)*, 2018.

[2] K. Grauman, A. Westbury, E. Byrne, Z. Chavis, A. Furnari, R. Girdhar, J. Hamburger, H. Jiang, M. Liu, X. Liu, et al. Ego4d: Around the world in 3,000 hours of egocentric video. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[3] C. Schuhmann, R. Beaumont, R. Vencu, C. Gordon, R. Wightman, M. Cherti, T. Coombes, A. Katta, C. Mullis, M. Wortsman, et al. Laion-5b: An open large-scale dataset for training next generation image-text models. *arXiv preprint arXiv:2210.08402*, 2022.

[4] 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.

[5] 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.

[6] 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.

[7] 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.

[8] D. Chicco. Siamese neural networks: An overview. *Artificial neural networks*, pages 73–94, 2021.

[9] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, et al. Dinov2: Learning robust visual features without supervision. *arXiv preprint arXiv:2304.07193*, 2023.

[10] 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 (CoRL)*, 2022.

[11] S. Nair, A. Rajeswaran, V. Kumar, C. Finn, and A. Gupta. R3m: A universal visual representation for robot manipulation. In *Conference on Robot Learning (CoRL)*, 2022.

[12] 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. In *International Conference on Learning Representations (ICLR)*, 2023.

[13] S. Karamcheti, S. Nair, A. S. Chen, T. Kollar, C. Finn, D. Sadigh, and P. Liang. Language-driven representation learning for robotics. *arXiv preprint arXiv:2302.12766*, 2023.

[14] N. Hansen, Z. Yuan, Y. Ze, T. Mu, A. Rajeswaran, H. Su, H. Xu, and X. Wang. On pre-training for visuo-motor control: Revisiting a learning-from-scratch baseline. *ICML*, 2023.

[15] T. Zhao, S. Karamchetim, T. Kollar, C. Finn, and P. Liang. What makes representation learning from videos hard for control? In *RSS 2022 Workshop on Scaling Robot Learning*, 2022.

[16] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, et al. Learning transferable visual models from natural language supervision. In *International Conference on Machine Learning (ICML)*, 2021.

[17] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, and A. Joulin. Emerging properties in self-supervised vision transformers. In *IEEE International Conference on Computer Vision (ICCV)*, 2021.

[18] Z. Tong, Y. Song, J. Wang, and L. Wang. Videomae: Masked autoencoders are data-efficient learners for self-supervised video pre-training. *arXiv preprint arXiv:2203.12602*, 2022.

[19] M. Laskin, A. Srinivas, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. In *International Conference on Machine Learning*, pages 5639–5650. PMLR, 2020.

[20] W. Ye, S. Liu, T. Kurutach, P. Abbeel, and Y. Gao. Mastering atari games with limited data. *Advances in Neural Information Processing Systems*, 34:25476–25488, 2021.

[21] M. Schwarzer, A. Anand, R. Goel, R. D. Hjelm, A. Courville, and P. Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv preprint arXiv:2007.05929*, 2020.

[22] Y. Seo, D. Hafner, H. Liu, F. Liu, S. James, K. Lee, and P. Abbeel. Masked world models for visual control. In *Conference on Robot Learning*, pages 1332–1344. PMLR, 2023.

[23] X. Lin, H. Baweja, G. Kantor, and D. Held. Adaptive auxiliary task weighting for reinforcement learning. In *Neural Information Processing Systems (NeurIPS)*, 2019.

[24] R. Shah and V. Kumar. Rrl: Resnet as representation for reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2021.

[25] 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 (ICML)*, 2022.

[26] A. Khandelwal, L. Weihs, R. Mottaghi, and A. Kembhavi. Simple but effective: Clip embeddings for embodied ai. In *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[27] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *International Conference on Learning Representations (ICLR)*, 2021.

[28] S. Amir, Y. Gandelsman, S. Bagon, and T. Dekel. Deep vit features as dense visual descriptors. *ECCV Workshop on What is Motion For?*, 2022.

[29] D. Hadjivelichkov, S. Zwane, L. Agapito, M. P. Deisenroth, and D. Kanoulas. One-shot transfer of affordance regions? affcorrs! In *Conference on Robot Learning (CoRL)*, 2022.

[30] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[31] S. Ross, G. Gordon, and D. Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011.

[32] Y. Geng, B. An, H. Geng, Y. Chen, Y. Yang, and H. Dong. End-to-end affordance learning for robotic manipulation. *International Conference on Robotics and Automation (ICRA)*, 2023.

[33] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. Isaac gym: High performance gpu-based physics simulation for robot learning, 2021.

[34] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

Appendix

Table of Contents

A Additional Experiments

A.1 Real World: How much data is necessary?

Given SpawnNet's success in the real-world experiments with little data, we aim to see how well our method generalizes given differing amounts of demonstrations. To do so, we additionally evaluate SpawnNet+d with 30, 60, and all 90 demonstrations. For comparison, we additionally include LfS+aug+d with 90 demos; results are reported in Table 7.

SpawnNet+d performs surprisingly well with few demonstrations; it approaches the performance of the LfS+aug+d baseline with fewer than a third of the training demonstrations and exceeds it with two-thirds, demonstrating the effectiveness of dense features even with few demonstrations.

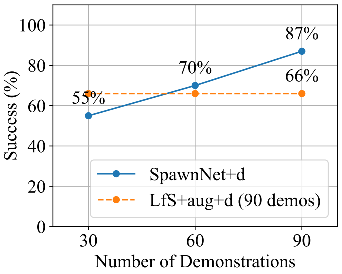


Figure 7: Comparing the number of training demos on *Place Bag*. With as few as 60 demos, SpawnNet exceeds LfS+aug+d with 90 demos.

B Architecture Design Choices

In this section, we describe specific choices in the architectures that we evaluate in our experiments.

Encoders. When comparing pretrained encoders, we control for similar parameter counts. MVP and DINO both use ViT-S (22M parameters), which has 12 layers. R3M uses ResNet-50 (23M parameters). For all encoders, we process each frame individually with the encoder, and concatenate representations across stacked frames and views before passing it into the MLP.

We further expand on our learned encoder architectures:

- **Learning-from-scratch architecture**: Our LfS architecture follows the deep convolutional encoder described in [34], with 128-channel 3x3 convolutions and 128-channel residual layers in each block. We detail this architecture in Figure 8.
- **SpawnNet**: For SpawnNet, we use ViT-S/8 with a stride of 8, resulting in spatial attention features of shape [384, 28, 28]. SpawnNet uses three adapters, taking pre-trained features from the 6^{th}, 9^{th}, and 12^{th} layers of the vision transformers respectively and mapping them to 64 channels (see Section 3.2) Each adapted 64-channel feature is then concatenated to the current learned 64-channel feature before a 128-channel residual block.

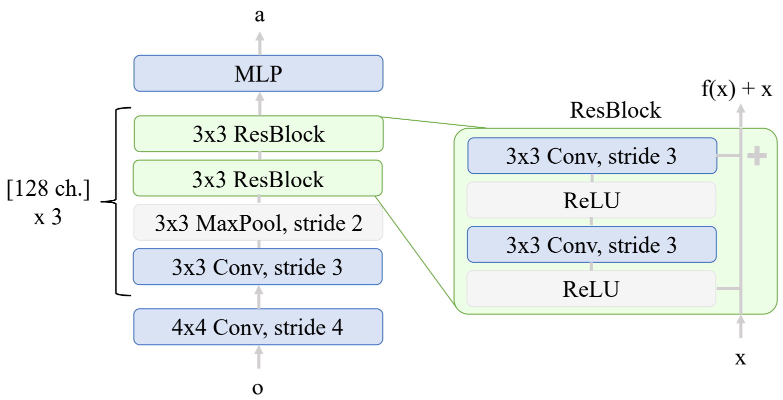


Figure 8: The convolutional encoder we consider for the LfS baseline. The initial 4x4 convolution transforms the input from its initial channel dimension to 128.

MLP. We parameterize the MLP for all encoders using the same architecture, with hidden layers of size [256, 128]. The feature vector extracted from the encoder is flattened first before being passed in.

Model Sizes and Inference Time. We report the number of trainable parameters and inference time for models trained on xArm tasks in Table 3. We note that *Inference* is the real-time inference speed; *Cached Inference* is the time taken for a forward pass with pre-calculated features (i.e. for training). Our LfS baseline has similar numbers of trainable parameters and cached inference speed as SpawnNet. Additionally, a SpawnNet backbone has approximately the same inference speed as a frozen pre-trained backbone.

Model	**Trainable Params (M)**	**Inference (ms)**	**Cached Inference (ms)**
DINO	0.84	57.27	0.27
Spawn-DINO	14.86	59.17	2.91
R3M	4.25	11.64	0.16
Spawn-R3M	15.02	14.15	2.83
LfS	15.11	2.47	2.47

Table 3: Inference times for different models. The increase in parameters between pre-trained and SpawnNet is from the use of dense spatial features instead of the CLS token.

Data Augmentations: Following Hansen et al. [14], we consider random shift and random color jitter data augmentations. For simulation tasks, we only apply data augmentation to LfS+aug with $p_{aug} = 0.5$. For real tasks, we apply data augmentation to all methods with $p_{aug} = 0.5$. We provide psuedocode for our implementations below:

```
import torchvision.transforms as T

sim_aug = T.Compose([ # random shift
    T.Pad(5, padding_mode='edge'),
    T.RandomResizedCrop(size=224, scale=(0.7, 1.0))])

real_aug = T.Compose([ # random shift (no pad) and color jitter
    T.RandomResizedCrop(size=224, scale=(0.7, 1.0)),
    T.ColorJitter(brightness=0.3)])
```

C Details on Real World Tasks

We provide more details about the real-world tasks, including the total number of demos, the breakdown of demos per instance, and further details about the experimental setup.

Place Bag: 102 total demonstrations, with 34 demonstrations split across a red, black, and brown bag. Bags are placed in front of the robot, with variations in the x-y position (within a 1'x1' box) and the rotation (within a 90 degree range). The stand is kept fixed on the table. The table itself translates up and down within a 3" range, adding height variation as well.

Hang Hat: 95 total demonstrations, with 30 demonstrations on a teal hat, 33 demonstrations on a black hat, and 32 on a navy hat. Demonstrations grab above the bill of the hat, and hats are placed on a fixed stand with varying rotation within a 90 degree range. The table height remains fixed.

Tidy Tools: 90 total demonstrations, with 15 per drawer. We define a drawer as a level on the shelf, and leave some "levels" as novel instances; this tests the policy's ability to generalize learned features spatially. We additionally vary the tool being manipulated between two different handled tools, and split these with 45 total demonstrations per tool across 6 different drawers. Tools are placed with a rotation within a 90 degree range inside of a 5"x5" box. Different drawers are placed with a rotation within a 45 degree range inside of a 5"x5" box. The table height remains fixed.

Evaluation. We place the novel instances within the training instances' pose variations as described above. Following [13], we additionally award partial credit for tasks which consist of multiple manipulation skills; for example, in the *Hang Hat* task, if the policy grasps the hat but is unable to hang it, we count the grasp as a success and the hang as a failure for a score of 0.5. The success rates are reported as the average success rate per instance. We perform 5 rollouts for each instance.

D Experimental Results

We produce numeric tables for all experiments presented.

D.1 Numerical Results for Simulation Experiments

The results in Table 4 correspond to the analysis presented in Section 4.1.

Method	Open Door		Open Drawer	
	Train	Val	Train	Val
SpawnNet	87.3 ± 1.6	58.3 ± 5.1	85.7 ± 2.8	61.1 ± 3.2
LfS+aug	90.5 ± 2.8	53.3 ± 2.4	81.0 ± 2.8	61.7 ± 2.4
R3M	92.1 ± 4.2	42.5 ± 5.1	82.5 ± 3.2	50.3 ± 1.8
MVP	68.3 ± 4.2	43.8 ± 4.7	58.7 ± 4.2	30.0 ± 1.6
PointCloudRL (expert)	89.5 ± 0.0	14.4 ± 0.0	95.2 ± 0.0	65.8 ± 0.0

Table 4: Numerical results for the simulation experiments.

D.2 Real Experiments

The results in Table 5 correspond to the analysis presented in Section 4.2.

E Pretrained Feature Visualization

Following the results presented in Figure 6, we present more examples of the learned features from the adapter layers. More visualizations can also be found on our project website.

Method	Place Bag		Hang Hat		Tidy Tools	
	Train	Val	Train	Val	Train	Val
SpawnNet+d	93.3 ± 3.8	86.7 ± 6.4	1.00 ± 0.0	80.0 ± 8.5	91.1 ± 3.6	88.6 ± 4.4
SpawnNet	83.3 ± 5.6	76.7 ± 10.8	93.3 ± 6.1	78.3 ± 6.4	90.0 ± 2.4	74.3 ± 6.3
LfS+aug+d	93.3 ± 3.8	66.7 ± 10.8	76.7 ± 7.3	60.0 ± 10.5	44.4 ± 9.4	33.3 ± 8.4
R3M	93.3 ± 3.8	20.0 ± 7.1	66.7 ± 13.9	20.0 ± 10.8	38.9 ± 7.4	15.2 ± 4.8

Table 5: Numerical performance on the real world tasks. We report the average of the total success rate for each instance. The bar denotes standard error.

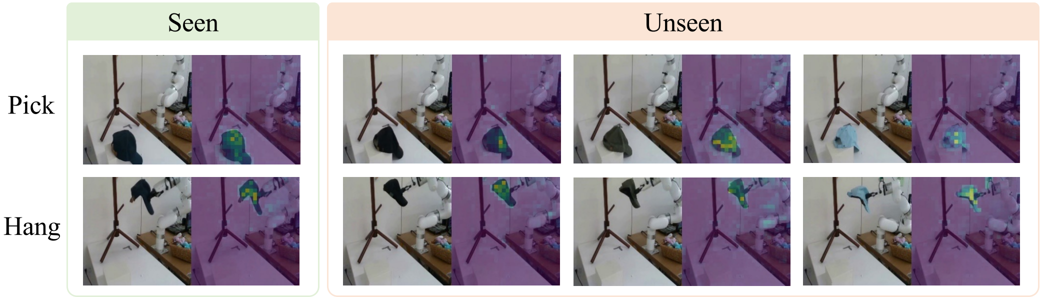


Figure 9: Visualized adapter features for the *Hang Hat* task. When grasping the hat, the adapter highlights relevant parts of the hat, such as the brim and front. When hanging the hat, the adapter highlights relevant parts of the hat, such as the back edge.

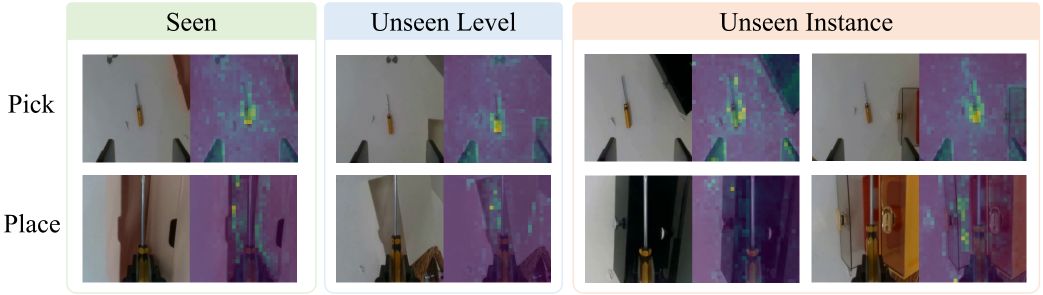


Figure 10: Visualized adapter features for the *Tidy Tools* task. When grasping the tool, the adapter highlights its handle, even with novel drawers in the background. When placing the tool in the drawer, the adapter highlights the drawer's front edge.