What Visual Foundation Model Should We Use in Robotics?

Ryan Chen [1] Bradly C. Stadie [1]

Abstract

Training machine learning agents to solve vision-based robotics tasks is both time and memory intensive. To alleviate this burden, many recent works have chosen not to learn their visual features end-to-end. Rather, they are often borrowed from Visual Foundation Models, large machine learning visual systems designed to transfer to a variety of tasks. Recently, several large scale vision models have been proposed for robotics. However, at present there does not seem to be a clear consensus on what visual foundation model is best, and on what really matters in selecting and deploying a vision foundation model on a robotic system. In this paper, we evaluate several vision foundation models for multi-goal robotics tasks. To our surprise, we find that segmentation models such as Segment Anything (SAM) consistently provide the best performance as a vision backbone for robotics tasks. This holds across a variety of tasks, and is true for both behavioral cloning and offline reinforcement learning. We investigate why this might be the case, and show that the overall fidelity and cohomological structure of representations learned by SAM fundamentally differ from other common vision foundation models, which may lead to improved performance in the robotics domain.

1. Introduction

Vision-based robotic systems often learn a visuomotor policy that is trained end-to-end in order to solve an environment. That is, a policy is jointly trained to simultaneously perceive the environment, and act optimally. These visuomotor policies are often difficult to train, requiring large numbers of samples and iterations to bootstrap both the vision system and optimal control. As large foundational models pre-trained on internet scale data have became more ubiquitous (Bommasani et al., 2021), there has been much community interest in using them to bypass the initialization problem in robotics.

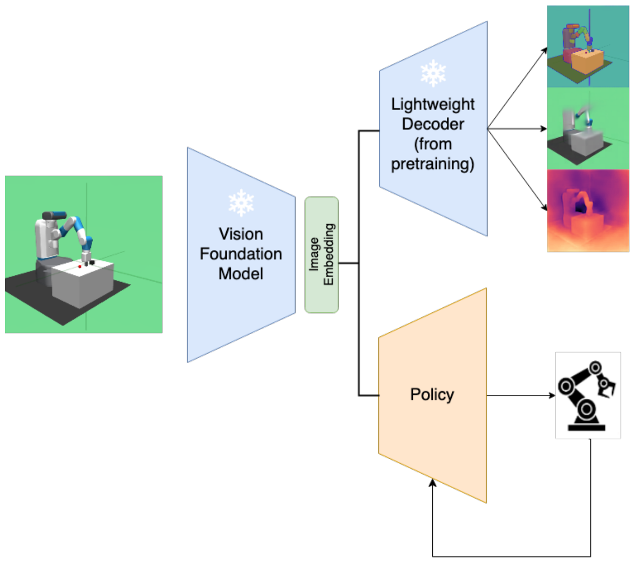

Figure 1. The image rendering of an environment is passed through the pre-trained vision encoder to extract its embeddings. The embeddings serve as a representation of the state and are used to learn a policy to solve the environment. Above shows pre-training tasks such as segmentation, masked auto-encoding, and depth estimation.

Yet, in spite of their clear promise, it is not inherently obvious which vision foundation model has the best inductive bias for solving robotics environments. In this paper, we investigate the use of various foundational vision models (Figure 1) in a variety of simulated robotics domains. Many foundation models are considered, including general models like Segment Anything and MAE (Kirillov et al., 2023; He et al., 2022), language-grounded models like CLIP (Radford et al., 2021), and robotics-specialized models like MVP and R3C (Radosavovic et al., 2022; Nair et al., 2022). To our surprise, we find that across a multitude of settings including imitation learning, offline RL, and goal-conditioned learning, the Segment Anything Model (SAM) consistently provide the best performance on visual robotics tasks.

*Equal contribution [1]Northwestern University, Evanston, IL, USA. Correspondence to: Ryan Chen <ryanchen2025@u.northwestern.edu>.

Submitted, ICML

In addition to providing the best performance, we find that robotics agents trained with SAM features are also more robust. When presented with additional distractor objects, policies learned with SAM suffer less degradation than policies trained with other visual foundation models. Experiments with reconstructing environment images from model embeddings indicate that SAM features maintain high-fidelity of visual objects, whereas other visual foundation models often seem to struggle with tracking objects across various spatial locations.

To better understand why SAM features achieved the best transfer to robotics, we investigated cohomological structures of the various foundational model image embeddings. Such a technique has recently been employed in NLP to better capture the intrinsic dimensionality of writing samples, and we are able to adapt it with some modification to the visual robotics domain. Upon doing so, we see that SAM embeddings achieve a persistent homology intrinsic dimension much lower than that of other visual representations, suggesting they are better able to distinguish detail in the environment.

2. Related Work

2.1. Vision Language Models

Vision language models (VLMs) have been used extensively in robotics. For example, VIMA (Jiang et al., 2023) and RoboCat (Bousmalis et al., 2023) both use a vision transformer to understand the scene and goals while jointly allowing for natural language prompting through an LLM. While these models are useful, the goal of linking the robotic system to an NLP system sits somewhat orthogonal to the aims of this work, which attempts to find the best visual features for solving robotic tasks.

2.2. Visual pre-training for Robotic Control

In robotic control, visual pre-training is concerned with learning visual features that are capable of transferring to a variety of downstream robotics control tasks. Some pre-training objectives accomplish this by using a vision pre-training objective that correlates with affordance maps (Yen-Chen et al., 2020). Other pre-training methods involve building an inductive bias of real world physics into the vision model (Radosavovic et al., 2022; Nair et al., 2022). Still other methods provide an inductive bias towards grasping and pose estimation (Ze et al., 2023). Common among these prior works is the need of a pre-training data sets that are *robotics-adjacent*, relevant to robotics in some way. This is in contrast to more general vision pre-training objectives such as reconstruction, which once trained can provide transfer to a variety of downstream tasks.

2.3. Inverse Reinforcement Learning from Pixels

Inverse reinforcement learning involves estimating a reward function when given a state. This task is especially difficult when the state is image-based, because of the inherent high dimensional nature of images. InfoGAIL addresses this by using a convolution networks pre-trained on ImageNet to extract image features for the policy. More recently, methods such as IQ-learn have shown how we can use Q-value estiamtion in this context (Garg et al., 2021). More specifically, IQ-learn assumes the Q value implicitly defines a reward function that can train optimal policies. In our work, we demonstrate the usefulness of various vision features for inverse reinforcement learning.

2.4. Transfer Learning

Transfer learning is the broad subset of learning deep representations to apply to downstream tasks. The most common approach is full fine-tuning of a pre-trained model. However, as models grow in size to the order of billions of parameters, especially in the NLP and vision domain (Brown et al., 2020; Touvron et al., 2023; Rombach et al., 2021), it is becoming less popular to full fine-tune. Instead, other methods that are more parameter efficient, leveraging intrinsic dimension (Aghajanyan et al., 2021), such as LoRA and QLoRA (Hu et al., 2022; Dettmers et al., 2023) are the trend of the day. This process can further be improved with a variety of tricks, e.g. embedding space noise (Jain et al., 2024). In this paper, we are specifically looking to avoid fine-tuning and its associated costs. Instead, we examine the current visual foundation models and evaluate which models already provide off-the-shelf transfer to robotics tasks, without any additional fine-tuning of the vision pipeline's features.

3. Vision Encoders

Many foundation vision models rely on a ViT backbone (Dosovitskiy et al., 2021). The ViT uses a transformer architecture, adapted from the NLP setting. First, images are sliced into small patches. Each patch is then projected into an embedding layer and attached with a positional embedding. Next, the embeddings are cast into the key-value-query space with learnable projections as prescribed by the multihead self-attention mechanism (Vaswani et al., 2017). Each transformer block consists of a multihead self-attention mechanism, a fully connected network, and then a layernorm (Ba et al., 2016) with a residual connection (He et al., 2016). After passing through several transformer blocks, the model outputs can be taken and used as features for downstream tasks.

To learn robust visual features, these architectures are usually trained on large-scale data sets with a general pre-training objective, e.g. image encoding-decoding or depth estiamtion. A difference in choice of pretraining objective can lead to different inductive biases, which is then carried into the learned image representations. In this work, we consider several ViT pretraining tasks that are specialized to robotics, aiming to endow the learned representations with knowledge useful for understanding the physical world.

CLIP aims to learn features that align linguistic and visual representations, i.e. an image and the written caption of this image should have similar CLIP embeddings. It was trained on a curated internet-scale data set of text-annotated images (Radford et al., 2021). CLIP's training objective is to match pairs of text and images using an loss function that maximizes cosine similarity between paired image and text embeddings, while minimizing the similarity of embeddings between non-paired texts and images (Mikolov et al., 2013). This type of paired training objective was explored previously by Sohn; Oord et al.. According to Radford et al., the highest performing CLIP model is based on the ViT architecture.

DINO is a class of models that are trained by self-distillation, removing the need to produce a labelled data set for training (Caron et al., 2021; Oquab et al., 2023). DINO's training loop consists of a student ViT g_θ and a teacher ViT f_ψ. An input image $\mathbf{x}$ is augmented in two different ways, with augmentations denoted by $s(\mathbf{x})$ and $t(\mathbf{x})$. These augmentations are passed through the student and teacher networks respectively to produce logits on K classes. These K classes are not labelled - the student g_θ is simply trained match the teacher output f_ψ by updating θ through gradient descent and the teacher's ψ is updated by an exponential moving average of θ. This pre-training objective has emerging capabilities such as the ability to detect object boundaries and producing rich enough features for k-NN classifiers for ImageNet (Caron et al., 2021).

MAE uses a masked autoencoding training process to produce representations. In particular, a ViT is trained to fully reconstruct images that have random patches masked out (He et al., 2022). This process proceeds with an encoder-decoder approach, first encoding the patch-masked image into a latent representation before attempting to reconstruct the original unpatched image from this representation. Trained MAE embeddings have been shown to perform well on a variety of downstream tasks, including ImageNet-1K, segmentation, and classification (He et al., 2022). MAE is a common backbone for other encoders, including SAM and MVP.

MVP Masked visual pre-training is a ViT that is trained the same way as the MAE, but on first person video frames of people performing common every day tasks (Radosavovic et al., 2022).

R3M (Nair et al., 2022) is a ResNet convolution backbone that was pretrained on similar data sets as MVP. However, its pre-training uses both a similar language alignment and time-contrastive objective. While MVP is trained to reconstruct static images of hand object interactions, R3M also captures information from the time domain.

SAM (Segment Anything) is another model that is built on top of the pretrained MAE, which was architecturally adapted to process large images (Kirillov et al., 2023). In addition to the MAE, a shallow convolutional neck, a prompt encoder and a lightweight decoder are added to the architecture to produce segmentation masks. The supervised training procedure evolved from using human mask annotators, which eventually enriched the model to be able to annotate its own segmentation masks in a fully automatic fashion. A total of 1.1 billion masks for 11 million images were used to train SAM, of which 99.1% of masks were automatically generated. SAM is able to segment images, detect edges, and solve object proposal in a zero-shot fashion.

4. Offline Imitation Learning

In the standard Markov Decision Process (MDP) setting, we have states $\mathcal{S}$, actions $\mathcal{A}$, and a state-action conditional transition probability distribution $\mathbf{T}(s_{t+1}|s_t, a_t)$. Let π be a policy, which defines a distribution over possible actions given a state $\pi : s \to \mathcal{D}(\mathcal{A})$. We can define the occupancy measure $\rho^\pi(s, a)$ as a discounted probability distribution over the joint state-action space $(\mathcal{S} \times \mathcal{A})$ that is induced by a policy π. Formally, the occupancy measure is:

$$\begin{aligned}\rho^\pi(s,a) &= \mathbf{E}\left(\sum_{t=0}^{\infty}\gamma^t \mathbf{P}^\pi(s_t, a_t)\right)\\ &= \mathbf{E}\left(\pi(a|s)\sum_{t=0}^{\infty}\gamma^t \mathbf{P}^\pi(s_t)\right)\\ &= \pi(a|s)\rho^\pi(s)\end{aligned}$$

where the expectation is taken over $(a, s) \sim \pi, \mathbf{T}$, and $\mathbf{P}^\pi(s_t, a_t)$ is the visitation probability distribution at time t by policy π.

In imitation learning, the goal is to train a policy π_θ capable of mimicking an expert π_E. This is typically done fitting the student policy π_θ to the data the expert has visited. Assume we are given access to a buffer of expert data $D_E = \{s_0, a_0, s_1, a_1, \ldots, s_N, a_N\} \sim \pi_E$. Then the goal is to use D_E to train a policy π_θ such that $D_{KL}(\rho^{\pi_E}(s)||\rho^{\pi_\theta}(s))$ is minimized.

4.1. Behavioral Cloning

Recovers expert behavior by directly learning a state-conditioned action density model $\pi_\theta(a|s)$. We seek to minimize $\mathbb{E}_{s,a\sim D_E} D_{KL}(\pi_\theta(a|s)||\pi_E(a|s))$. In practice, π_θ is usually trained to minimize the L2 norm between its own actions and the expert's actions. Given $(s_t, a_t) \sim D_E$, we optimize to find

$$\arg\min_\theta \|\pi_\theta(a|s_t) - a_t\|_2^2$$

An alternative formulation learns a mapping $\pi_\theta : s \to \mu, \log\sigma$, from which we get a by sampling $a \sim \mathcal{N}(\mu, \exp(\log(\sigma))$. Under this formulation, we fit the maximum likelihood loss to find μ, σ that maximize the probability $P(a|s;\mu,\sigma)$. The argument is similar for discrete action spaces, replacing the distribution with categorical and the loss instead with standard cross entropy.

Behavior cloning makes assumptions that each draw of $(\mathcal{S}, \mathcal{A})$ is independent of other draws, where each draw represents a single time step. This simplicity makes behavior cloning a convenient choice for imitation learning, however due to the independence assumption, it ignores any notion of transitional dynamics. Furthermore, as it only estimates the occupancy measure using single time steps, it suffers from compounding errors (Ross et al., 2011).

4.2. Inverse Offline Reinforcement Learning

Offline reinforcement learning offers another approach towards learning from a static offline data set. Instead of optimizing $\|\pi_\theta(a|s) - a\|_2^2$ using samples from $(s,a) \sim \mathcal{D}_E$, offline RL treats D_E as a replay buffer and uses it to estimate a Q-function. Naively, the agent tries to maximize the sum of discounted rewards $\sum_i \gamma^i r_i$ by replaying trajectories from $\mathcal{D}_E$.

There is a flaw in the naive approach to offline RL. Since the agent cannot encounter any new experiences that are not in the data set, the estimates for $Q(s,a)$ are typically not robust to (s,a) outside the training distribution. For novel states encountered at test time, the learned $Q(s,a)$ is usually not accurate and of dubious utility. One way address this is with implicit Q-learning (Kostrikov et al., 2022). Implicit Q-learning assumes that for the expert policy π_E, we wish to learn some imitation policy π such that:

$$Q(s,a) = r(s,a) + \mathbf{E}_{a\sim\pi}(Q(s,a))$$
$$\text{s.t. } D_{KL}(\pi\|\pi_E) \le \epsilon$$

Note that $\mathbf{E}_{a\sim\pi}(Q(s,a))$ is precisely the value function $V^\pi(s)$. Thus, minimizing $(V^\pi(s) - Q(s,a))^2$ with respect to the parameters of $V^\pi(s)$ will give us the expectation $\mathbf{E}(Q(s,a))$.

In an expert replay, a state-action pair (s_E, a_E) can serve as an approximation to nearby state-action pairs that do not exist in the expert replay. One plausible method of accomplishing this is to view the value function $V(s_E)$ at a given expert replay state s_E as a random variable with some distribution, with randomness induced from all possible actions that can be taken at state s. To estimate the value $V(s)$ of a given state, recent works propose to estimate an upper quantile of the distributional V by using expectile loss (Koenker & Hallock, 2001; Dabney et al., 2018), an asymmetric version of MSE loss with a slower growing positive tail. Kostrikov et al. show that this is the same as a Bellman backup with an implied argmax policy over the actions in the support of a given state s, that is:

$$\pi = \pi_{\text{implied}}(\alpha|s) = \mathbf{1}(\alpha = \arg\max_{a\in\text{Supp}(s)} Q(s,a))$$
$$\text{Supp}(s) := \{a : \pi_E(a|s) > 0\}$$

The Bellman backup in implicit Q-learning requires knowing the rewards of the expert trajectories to update the Q critic. However, expert trajectories of the form $(\mathcal{S}, \mathcal{A})$ do not contain a reward function. While it is possible to use adversarial methods to perform inverse reinforcement learning (Ho & Ermon, 2016; Li et al., 2017), (Garg et al., 2021) proposes a Q-critic loss where the reward function is implicitly estimated and used for the Q-update, called inverse soft-Q learning. Substituting the critic loss in implicit Q-learning with the critic loss from inverse soft-Q learning, we arrive at an inverse implicit Q-policy.

5. Experimental Setup

5.1. Data Collection

We collect expert data from four environments: FetchPush-v2, FetchPickAndPlace-v2, AdroitHandHammer-v1, AdroitHandDoor-v1. Hindsight Experience Replay (Andrychowicz et al., 2017) experts were trained for Push-v2 and PickAndPlace-v2, and the expert trajectories were collected. The Adroit expert trajectories were collected from the Minari repository (Fu et al., 2020) and experts were cloned to provide a goal representation from any state when doing goal-conditioning.

In AdroitHandHammer, the environment randomly initializes the position for the hammer and board on the table. The agent must locate the hammer, pick up the hammer, and hit the nail. In AdroitHandDoor, the door position are randomized and the agent must locate and pull the door handle. Likewise in the Fetch environments, the block and the desired destination is randomized. These configurations make the environment suitable for a multi-goal setting. In the training process, we require the policy to learn the kinematics and objects in the environment solely from the visual features. Examples of the environments are shown in Figure 2.

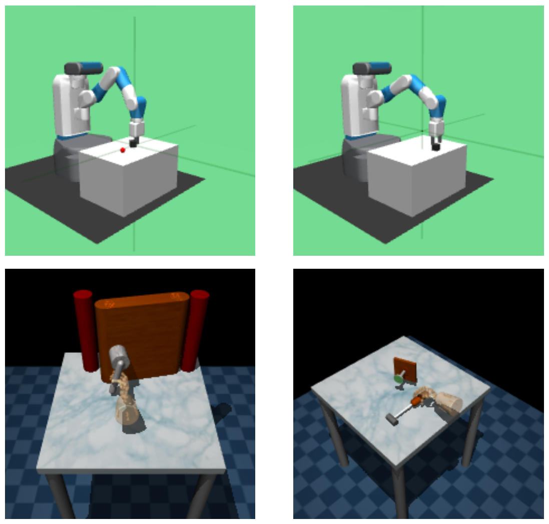

Figure 2. Examples of the state space represented as images for each environment are shown above, from top left to bottom right, Push, Place, Door, Hammer.

5.2. Behavior Cloning

We evaluated the various vision encoders with both behavior cloning and goal-conditioned behavior cloning. We use the same set of environments for both goal-conditioned and non-goal conditioned behavioral cloning, with the only environmental difference being the presence of the goal.

For behavior cloning, we take the images of the expert trajectories and pass them through the vision backbones to form the image representations. We do not include any positional information for the Push, Hammer, and Door environments. For PickAndPlace, we only pass the gripper position and velocity information to the network, as those are directly controlled in the action space. In the goal-conditioned setting, we augment the observations with the goal image that is provided by an expert.

The state and goal embeddings are each passed through a feed forward network with hidden layer sizes [256, 128, 64]. The positional information, if used, is projected to 128-dimensions and concatenated with the output of the image feed forward networks. This concatenated vector is then passed through a network with hidden layer sizes [256, 128, 64] before predicting an action in the same dimension of the action space. The SELU activation is used in between each layer. In the non-goal-conditioned experiments, we do not pass the goal image embeddings, and the positional information is projected to 64-dimensional space.

5.3. Implicit Q-Learning

We also use implicit Q-learning (Kostrikov et al., 2022) to evaluate the various encoders from the offline reinforcement learning framework. As in behavior cloning, we only provide the image embeddings, and do not provide any positional information for Push, Hammer, and Door. PickAndPlace takes only positional information on gripper position and velocity.

In implicit Q-learning, a value estimator network with hidden layer sizes [256, 256] takes an image embedding and estimates a value. When using goal conditioning, the current image and goal image embeddings are concatenated and fed into the network. A twin critic network, each with hidden layer sizes [256, 256] takes in the state information and the value estimate, which is used to estimate Q values. In standard implicit Q-learning, the Bellman backup for Q values relies on knowing the reward which is not in the expert replay. Thus, we modify the critic loss to the soft inverse Q-learning objective from (Garg et al., 2021), which implicitly learns a reward given the state images.

The policy we use is a stochastic Gaussian policy parameterized by a network with hidden layer sizes [256, 256, 256], which takes the image embeddings as inputs and outputs an action sampled from a Gaussian with learned parameters. During inference time, the mean of the Gaussian is output as the action. All networks used ReLU activations. We modified the implementation of implicit Q-learning from (Tarasov et al., 2022) to interface with our replay buffer and critic loss.

6. Experiments

We perform experiments across various visual backbones and environments, using both inverse IQL and behavior cloning. The exact architectures and training parameters used are listed in Appendix A.1 and A.2.

In Table 1 and Table 2, we compare the performance of SAM, DINO-v2, CLIP, MAE, MVP, and R3M vision backbones. We observe from these results that policies using the SAM backbone perform best in 14/16 environment setups. For behavioral cloning, SAM policies performs better than all methods across environments except for Hammer, wherein R3M performs the best. However, in the offline reinforcement learning framework, the SAM based policies perform better than all other policies by a considerable margin. In Table 2, we perform the same experiments without goal conditioning. The results are very similar to that of the goal conditioned regime, with SAM based encoders uniformly performing the best in the offline reinforcement learning setting.

Across the board, behavior cloning and goal conditioned behavior cloning do not perform as well as their offline reinforcement learning counterparts, which agrees with current literature (Kumar et al., 2022), where it has been suggested that offline reinforcement learning's performance should

Table 1. Performance of encoders with goal conditioning

	FROZEN BACKBONE	INVERSE IQL
DOOR-V1		
SAM	$\mathbf{0.968 \pm 0.02}$	$\mathbf{0.966 \pm 0.02}$
DINO-v2	0.614 ± 0.28	0.911 ± 0.08
CLIP	0.629 ± 0.32	0.922 ± 0.03
MAE	0.563 ± 0.32	0.650 ± 0.17
MVP	0.000 ± 0.00	0.000 ± 0.00
R3M	0.960 ± 0.07	0.872 ± 0.03
HAMMER-V1		
SAM	0.769 ± 0.16	$\mathbf{0.863 \pm 0.09}$
DINO-v2	0.685 ± 0.15	0.779 ± 0.09
CLIP	0.479 ± 0.11	0.690 ± 0.09
MAE	0.371 ± 0.23	0.747 ± 0.05
MVP	0.000 ± 0.00	0.000 ± 0.00
R3M	$\mathbf{0.968 \pm 0.07}$	0.363 ± 0.23
PUSH-V2		
SAM	$\mathbf{0.684 \pm 0.08}$	$\mathbf{0.793 \pm 0.06}$
DINO-v2	0.170 ± 0.02	0.350 ± 0.02
CLIP	0.186 ± 0.02	0.340 ± 0.01
MAE	0.148 ± 0.01	0.256 ± 0.01
MVP	0.140 ± 0.00	0.253 ± 0.02
R3M	0.147 ± 0.01	0.360 ± 0.02
PLACE-V2		
SAM	$\mathbf{0.621 \pm 0.03}$	$\mathbf{0.440 \pm 0.03}$
DINO-v2	0.233 ± 0.03	0.270 ± 0.02
CLIP	0.216 ± 0.03	0.210 ± 0.03
MAE	0.100 ± 0.02	0.247 ± 0.02
MVP	0.096 ± 0.01	0.130 ± 0.02
R3M	0.273 ± 0.06	0.190 ± 0.06

Table 2. Performance of encoders without goals

	FROZEN BACKBONE	INVERSE IQL
DOOR-V1		
SAM	$\mathbf{0.967 \pm 0.02}$	$\mathbf{0.967 \pm 0.02}$
DINO-v2	0.909 ± 0.12	0.961 ± 0.03
CLIP	0.878 ± 0.13	0.944 ± 0.03
MAE	0.574 ± 0.33	0.694 ± 0.14
MVP	0.000 ± 0.00	0.000 ± 0.00
R3M	0.878 ± 0.20	0.872 ± 0.03
HAMMER-V1		
SAM	0.721 ± 0.26	$\mathbf{0.833 \pm 0.02}$
DINO-v2	0.767 ± 0.19	0.686 ± 0.05
CLIP	0.489 ± 0.11	0.617 ± 0.11
MAE	0.387 ± 0.25	0.233 ± 0.15
MVP	0.000 ± 0.00	0.000 ± 0.00
R3M	$\mathbf{0.818 \pm 0.24}$	0.463 ± 0.08
PUSH-V2		
SAM	$\mathbf{0.504 \pm 0.08}$	$\mathbf{0.723 \pm 0.05}$
DINO-v2	0.166 ± 0.02	0.357 ± 0.06
CLIP	0.166 ± 0.03	0.280 ± 0.08
MAE	0.140 ± 0.00	0.213 ± 0.06
MVP	0.140 ± 0.00	0.207 ± 0.07
R3M	0.142 ± 0.01	0.320 ± 0.07
PLACE-V2		
SAM	$\mathbf{0.271 \pm 0.074}$	$\mathbf{0.370 \pm 0.05}$
DINO-v2	0.144 ± 0.02	0.187 ± 0.03
CLIP	0.134 ± 0.04	0.210 ± 0.02
MAE	0.075 ± 0.02	0.223 ± 0.04
MVP	0.055 ± 0.01	0.093 ± 0.06
R3M	0.135 ± 0.05	0.367 ± 0.05

dominate in noisy environments with sparse rewards.

6.1. Distractions in the Environment

The state space, especially when given as an image, may contain extraneous information that is irrelevant to the task. Thus embeddings will also contain extra information, which can make them sensitive to irrelevant change in the environment. To test the sensitivity of various image backbones to irrelevent environmental change, we modify the environments to include distractor object (e.g. another geometric shape of a different color). During training, the standard environment is used. During test time, we inject the distractor into the environment, but ensure it does not obstruct the task. This makes the entire inference time experience out of distribution from the data it was trained on, but hopefully in a way that will allow test time generalization when using strong visual features.

Across the board, SAM based policies perform best as shown in Table 3. Its success rate in Push with distractions only decreased by about four percentage points, which is still within the error bounds of its performance in the Push environment with no distractors. Meanwhile, its average success rate in PickAndPlace still lies well within the 0.440 ± 0.03 range from the PickAndPlace environment with no distractions. In comparison, other vision encoders, such as CLIP and DINO-v2, appear to have a significant drop in performance when the distractor is introduced into the environment, with R3M dropping from 0.36 to 0.19 in the Push environment with distractions.

7. Why Segment Anything Features Work Well

7.1. Fidelity of learned representations

In order for a visual backbone to provide useful information to solve the environment, the image embeddings should be able to provide some level of information about the locations of the objects in the environment. To understand the abilities of various models in this arena, we train a lightweight model to reconstruct the beginning and ending images of a trajectory from their image embeddings. The Fetch environment provides a good benchmark to evaluate the various embeddings because the block, goal, and robotic arm are free to move on the table. This variability in the environ-

Table 3. Performance of encoders in environments with a distraction using goal-conditioned learning

ENCODER	SAM	DINO-V2	CLIP	MAE	MVP	R3M
FETCHPUSH-V2	$\mathbf{0.733 \pm 0.03}$	0.380 ± 0.04	0.303 ± 0.01	0.207 ± 0.07	0.210 ± 0.01	0.230 ± 0.06
FETCHPICKANDPLACE-V2	$\mathbf{0.436 \pm 0.04}$	0.167 ± 0.02	0.123 ± 0.01	0.233 ± 0.01	0.113 ± 0.03	0.160 ± 0.02

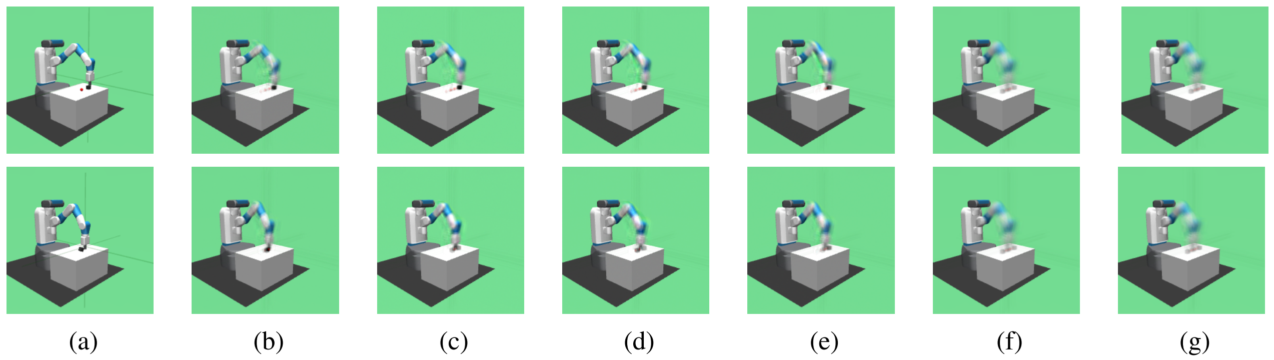

Figure 3. Top Row: Image representation at the beginning of the trajectory. Bottom Row: Image representation at the end of the trajectory a) True image generated by the environment b) SAM reconstruction c) DINO-v2 reconstruction d) CLIP reconstruction e) MAE reconstruction f) MVP reconstruction g) R3M reconstruction.

ment requires more informative representations in order to reconstruct image representations. The decoder training details are shown in Appendix A.3.

Additionally, since the performance difference between SAM based policies and the other encoder policies is the most pronounced in FetchPush, we particularly visualize the reconstructions of the FetchPush trajectories in Figure 3.

SAM, DINO-v2, CLIP embeddings (b), (c), (d) are sufficient to reconstruct the beginning of the environment. However, as the trajectory proceeds, both DINO-v2 and CLIP begin to reconstruct the block on both sides of the gripper, suggesting it is harder to extract exact details of the block location from these embeddings. While the reconstruction from MAE embeddings (e) can locate the block, it does not reconstruct the goal in the beginning, and eventually the MAE embedding produces reconstructions where the block appears to be on the wrong side of the gripper. Both MVP (f) and R3M (g) are unable to encode the arm position to any degree of certainty, and reconstruction from these features are unable to recover the location of either the goal or the block throughout the trajectory.

The information encoded in SAM embeddings about block location, arm position, and goal location are more easily extracted. While SAM cannot initially reconstruct the red goal marker in the correct positions, SAM embeddings do contain rich enough information to reconstruct the goal image representation. Thus goal conditioning still allows an agent to learn to arrive at the goal even if the goal representation at the beginning is not clear.

7.2. Topological Analysis

When additional objects appear in the image representations of the state during runtime, the image embeddings will be fundamentally different from the embeddings in the training data set, leading to an out-of-distribution problem. However, SAM embeddings empirically appear to be more robust to these changes in runtime experience. To investigate why, we draw on the cohomological structure of these high dimensional image representations.

Suppose an image embedding of the state representation from the offline data set is of dimension d, $e_t^s \in \mathbf{R}^d$. Then a trajectory represented by these image embeddings can be seen as a sequence $(e_t^s)_{t\geq 0}$ in $\mathbf{R}^d$ space. Imitation learning learns a function of e_t^s, $\pi(a_t|e_t^s)$, which should generalize over the entire manifold where e_t^s lies. If during runtime, the agent experiences e_t^* embeddings that lie on a manifold fundamentally different in structure than that of the expert data set, then the learned $\pi(a_t|e_t^s)$ is not expected to be a good approximation to an optimal policy to trajectories on the runtime manifold.

We measure the difference in manifold structure by persistent homology intrinsic dimension ($PHID_0$). Using $PHID_0$ to find differences in embedding space has had success in NLP applications (Tulchinskii et al., 2023) and we apply a similar strategy to analyze robustness of policies that are based on image embeddings.

$PHID_0$ is built upon persistence homologies, which studies general manifold topology by sampling points from the manifold. At initialization, each sampled point is treated as its own connected component. From each point, a hypersphere of radius δ is born, and the radius is grown until it intersects with another sphere. At that point, we say two connected components are merged into one, implying that one connected component has died, and its lifespan is recorded. The spheres continue to grow, merging with others until the entire sample has merged into a single connected component. From the merges, we can record the lifespan of each connected component $|I|$, which resembles the building blocks of the Hausdorff dimension. In fact, PHID belongs to the set of box-counting dimensions which are an estimate of fractal dimension (Jaquette & Schweinhart, 2020). We define the $PHID_0$ as the exponentially weighted sum of lifespans:

$$PHID_0 = \sum_{i \in PH_0} |I_i|^\alpha$$

A sequence of image embeddings that represent an environment trajectory can be seen as a sequence of points sampled from a $\mathbf{R}^d$ manifold. We use persistence homology to estimate the intrinsic dimension $PHID_0$ of the image embedding manifold on which the trajectories lie. If z_1 is the $PHID_0$ for a trajectory of embeddings in the FetchPush-v2 environment given by an encoder e, and z_2 is the $PHID_0$ for the same trajectory in the FetchPush-v2 environment with distractions, then the difference in $PHID_0$ is given by $|z_1 - z_2|$. This quantifies how different the manifolds are between the environment without distractions, and the environment with distractions.

If for trajectory i, an encoder e produces embeddings in $\mathbf{R}^d$, then we define $\Delta_i = \frac{|z_1 - z_2|}{d}$ to denote average change in $PHID_0$ attributed to each dimension d, which accounts for the curse of dimensionality and allows for comparison across encoders.

We sample 100 trajectories from the FetchPush and FetchPickAndPlace environments, and compared them with the same trajectories, but played in the FetchPushDistractions and FetchPickAndPlaceDistractions environments. For each trajectory, we calculate the Δ_i of the intrinsic dimension, then take the mean. Due to the variability from sampling, we also standardize with the variance. The final $PHID_0$ score is:

$$PHID^*_{0,\text{encoder}} = \frac{\frac{\bar{\Delta}}{d_{\text{encoder}}}}{\frac{s}{\sqrt{d_{\text{encoder}}}}}$$

The standardized differenced $PHID_0$ scores are in Table 4. SAM shows the lowest difference between the dimensions. When a visiomotor policy using a SAM backbone is presented with a distractor in the environment, the trajectory of SAM embeddings do not fundamentally change in structure, as much as the other embedding trajectories from the other encoders do. Thus the learned SAM-based policy from an undistracted environment has more ability to generalize to an environment with a distractor. That is, SAM provides relatively more robust trajectory embeddings, that are not too fundamentally different from the embeddings generated when a distractor is present.

Table 4. Difference in persistent homology intrinsic dimension

ENCODER	FETCHPUSH	FETCHPICKANDPLACE
SAM	**0.26**	**0.27**
R3M	1.45	0.71
MAE	1.51	1.29
MVP	2.17	0.64
CLIP	0.69	0.46
DINO-v2	1.94	0.70

8. Conclusion and Opportunities

We perform experiments to evaluate whether SAM based visiomotor policies perform better in robotics environments. We evaluated performance in environments using both behavior cloning and offline reinforcement learning algorithms with an expert replay buffer. SAM performs best in most environments with behavior cloning, and performs the best across all environments using offline reinforcement learning. To investigate the success of SAM embeddings in these robotic environments, we observe that SAM can reconstruct image representations with higher fidelity, relative to the embeddings from other encoders. Additionally, we looked at how well the encoders can produce good policies even in out-of-distribution situations such as when there is an additional distraction object in the environment. SAM features turn out to be robust in these out-of-distribution environments, and we examine why this happens using tools from homology theory.

As vision foundation models become popular, and models with better generalizability and stronger visual features are trained, we expect there to be more competitive visual backbones that can provide impressive downstream transfer to robotic manipulation tasks. We looked at the how persistence homologies could explain robustness of features, which can be adapted as a regularization technique for future vision foundation model pre-training objectives. As vision language models for embodied agents become popular, another future direction is to study how well the embodied agent can perform a task using a segmentation visual backbone in conjunction with various language models. Finally, we are interested in using diffusion-based approaches on the embedding space of vision foundation models to solve visiomotor tasks using online reinforcement learning.

References

Aghajanyan, A., Gupta, S., and Zettlemoyer, L. Intrinsic dimensionality explains the effectiveness of language model fine-tuning. In Zong, C., Xia, F., Li, W., and Navigli, R. (eds.), *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pp. 7319–7328. Association for Computational Linguistics, August 2021.

Andrychowicz, M., Wolski, F., Ray, A., Schneider, J., Fong, R., Welinder, P., McGrew, B., Tobin, J., Pieter Abbeel, O., and Zaremba, W. Hindsight experience replay. *Advances in neural information processing systems*, 30, 2017.

Ba, L. J., Kiros, J. R., and Hinton, G. E. Layer normalization. *CoRR*, abs/1607.06450, 2016.

Bommasani, R., Hudson, D. A., Adeli, E., Altman, R., Arora, S., von Arx, S., Bernstein, M. S., Bohg, J., Bosselut, A., Brunskill, E., et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

Bousmalis, K., Vezzani, G., Rao, D., Devin, C., Lee, A. X., Bauza, M., Davchev, T., Zhou, Y., Gupta, A., Raju, A., Laurens, A., Fantacci, C., Dalibard, V., Zambelli, M., Martins, M., Pevceviciute, R., Blokzijl, M., Denil, M., Batchelor, N., Lampe, T., Parisotto, E., Żołna, K., Reed, S., Colmenarejo, S. G., Scholz, J., Abdolmaleki, A., Groth, O., Regli, J.-B., Sushkov, O., Rothörl, T., Chen, J. E., Aytar, Y., Barker, D., Ortiz, J., Riedmiller, M., Springenberg, J. T., Hadsell, R., Nori, F., and Heess, N. RoboCat: A self-improving foundation agent for robotic manipulation. *arXiv preprint arXiv:2306.11706*, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 1877–1901. Curran Associates, Inc., 2020.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 9650–9660, 2021.

Dabney, W., Rowland, M., Bellemare, M., and Munos, R. Distributional reinforcement learning with quantile regression. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Darcet, T., Oquab, M., Mairal, J., and Bojanowski, P. Vision transformers need registers, 2023.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms. *arXiv preprint arXiv:2305.14314*, 2023.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

Fu, J., Kumar, A., Nachum, O., Tucker, G., and Levine, S. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

Garg, D., Chakraborty, S., Cundy, C., Song, J., and Ermon, S. Iq-learn: Inverse soft-q learning for imitation. In *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

He, K., Zhang, X., Ren, S., and Sun, J. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.

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

Ho, J. and Ermon, S. Generative adversarial imitation learning. *Advances in neural information processing systems*, 29, 2016.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*, 2022.

Ilharco, G., Wortsman, M., Wightman, R., Gordon, C., Carlini, N., Taori, R., Dave, A., Shankar, V., Namkoong, H., Miller, J., Hajishirzi, H., Farhadi, A., and Schmidt, L. Openclip, July 2021. URL https://doi.org/10.5281/zenodo.5143773.

Jain, N., yeh Chiang, P., Wen, Y., Kirchenbauer, J., Chu, H.-M., Somepalli, G., Bartoldson, B. R., Kailkhura, B., Schwarzschild, A., Saha, A., Goldblum, M., Geiping, J., and Goldstein, T. NEFTune: Noisy embeddings improve instruction finetuning. In *The Twelfth International Conference on Learning Representations*, 2024.

Jaquette, J. and Schweinhart, B. Fractal dimension estimation with persistent homology: A comparative study. *Communications in Nonlinear Science and Numerical Simulation*, 84:105163, 2020.

Jiang, Y., Gupta, A., Zhang, Z., Wang, G., Dou, Y., Chen, Y., Fei-Fei, L., Anandkumar, A., Zhu, Y., and Fan, L. Vima: General robot manipulation with multimodal prompts. In *Fortieth International Conference on Machine Learning*, 2023.

Kirillov, A., Mintun, E., Ravi, N., Mao, H., Rolland, C., Gustafson, L., Xiao, T., Whitehead, S., Berg, A. C., Lo, W.-Y., Dollar, P., and Girshick, R. Segment anything. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 4015–4026, October 2023.

Koenker, R. and Hallock, K. F. Quantile regression. *Journal of economic perspectives*, 15(4):143–156, 2001.

Kostrikov, I., Nair, A., and Levine, S. Offline reinforcement learning with implicit q-learning. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Kumar, A., Hong, J., Singh, A., and Levine, S. Should i run offline reinforcement learning or behavioral cloning? In *International Conference on Learning Representations*, 2022.

Li, Y., Song, J., and Ermon, S. Infogail: Interpretable imitation learning from visual demonstrations. *Advances in neural information processing systems*, 30, 2017.

Mikolov, T., Chen, K., Corrado, G., and Dean, J. Efficient estimation of word representations in vector space. In Bengio, Y. and LeCun, Y. (eds.), *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, 2013.

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

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

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

Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., Sastry, G., Askell, A., Mishkin, P., Clark, J., et al. Learning transferable visual models from natural language supervision. In *International conference on machine learning*, pp. 8748–8763. PMLR, 2021.

Radosavovic, I., Xiao, T., James, S., Abbeel, P., Malik, J., and Darrell, T. Real-world robot learning with masked visual pre-training. *CoRL*, 2022.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-resolution image synthesis with latent diffusion models, 2021.

Ross, S., Gordon, G., and Bagnell, D. 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*, pp. 627–635. JMLR Workshop and Conference Proceedings, 2011.

Sohn, K. Improved deep metric learning with multi-class n-pair loss objective. *Advances in neural information processing systems*, 29, 2016.

Tarasov, D., Nikulin, A., Akimov, D., Kurenkov, V., and Kolesnikov, S. CORL: Research-oriented deep offline reinforcement learning library. In *3rd Offline RL Workshop: Offline RL as a "Launchpad"*, 2022.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. Llama 2: Open foundation and fine-tuned chat models, 2023.

Tulchinskii, E., Kuznetsov, K., Laida, K., Cherniavskii, D., Nikolenko, S., Burnaev, E., Barannikov, S., and Piontkovskaya, I. Intrinsic dimension estimation for robust detection of AI-generated texts. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

Yen-Chen, L., Zeng, A., Song, S., Isola, P., and Lin, T.-Y. Learning to see before learning to act: Visual pre-training for manipulation. In *IEEE International Conference on Robotics and Automation (ICRA)*, 2020.

Ze, Y., Liu, Y., Shi, R., Qin, J., Yuan, Z., Wang, J., and Xu, H. H-index: Visual reinforcement learning with hand-informed representations for dexterous manipulation. *NeurIPS*, 2023.

A. Model Details

A.1. Vision Encoders

Each vision encoder has a slightly different architecture. We extracted the vision embeddings from the following layers:

CLIP was implemented in (Ilharco et al., 2021). We use the ViT-B-16 encoder and extract the embeddings from the visual encoder after the pooling layer is projected to an embedding of dimension of 512.

MAE was implemented in `https://github.com/facebookresearch/mae`. We use the ViT-B-16 encoder and extracted embeddings from the normalization layer that comes after the attention blocks. This resulted in embeddings of dimension 768.

MVP was implemented by (Radosavovic et al., 2022) and is built on top of the ViT encoders. We also used the ViT-B-16 encoder to extract embeddings of dimension 768.

DINO-v2 was implemented in (Oquab et al., 2023). We used the ViT-B-14 encoder trained with registers due to the significant performance gains documented in (Darcet et al., 2023). The visual backbone produces embeddings that we extracted after the norming layers that come after the attention blocks. This resulting in embeddings of dimension 768.

SAM was implemented in (Kirillov et al., 2023). SAM uses a modified MAE, where the final transformer block output is sent through a 2 layer convolutional neck. We extract image embeddings from the output of the neck, which are the image embeddings as specified in `https://github.com/facebookresearch/segment-anything/`. The image embeddings we use here are of size 9216.

R3M was implemented by (Nair et al., 2022). This is a ResNet based architecture, where the embeddings are the output of the ResNet-50 model. This yielded embeddings of size 2048.

A.2. Behavioral cloning and inverse IQL

The following model parameters were determined by performance of imitation learning on state space information. We chose parameters for behavior cloning where BC on state space achieved > 0.95 success rate. For behavior cloning, a learning rate of 0.008 was chosen for all experiments. We chose parameters for IQL where IQL on state space achieved > 0.95 success for the last 5 evaluation steps. In IQL, expectile loss parameter was chosen to be $\tau = 0.7$, and the learning rates for the actor, critic, and value estimator are 1.5×10^{-4}, 3×10^{-4} and 3×10^{-4} respectively.

Table 5. Model Training Parameters

	Environment	Push-v2	PickAndPlace-v2	Hammer-v1	Door-v1
Behavior Cloning	Trajectories	2900	2900	5	50
	Epochs	1000	1000	100	30
	Batch size	256	256	32	32
IQL	Batch size	1024	1024	256	256
	Steps	6×10^4	10^5	5000	1500

A.3. Reconstruction

For a given set of embeddings from each of the visual encoders, we train a separate decoder. We pass the embeddings through a simple feed forward network with hidden layer size [1024, 2048, 5096] with ReLU activations. The output was reshaped to size $3 \times 224 \times 224$. Each reconstructor, was trained on 20 epochs using images from 1,000 trajectories, with a batch size of 32. We used AdamW with a learning rate of 0.001.