# Just Cluster It: An Approach for Exploration in High-Dimensions using Clustering and Pre-Trained Representations

**Stefan Sylvius Wagner** [1] **Stefan Harmeling** [2]

## Abstract

In this paper we adopt a representation-centric perspective on exploration in reinforcement learning, viewing exploration fundamentally as a density estimation problem. We investigate the effectiveness of clustering representations for exploration in 3-D environments, based on the observation that the importance of pixel changes between transitions is less pronounced in 3-D environments compared to 2-D environments, where pixel changes between transitions are typically distinct and significant. We propose a method that performs episodic and global clustering on random representations and on pre-trained DINO representations to count states, i.e, estimate pseudo-counts. Surprisingly, even random features can be clustered effectively to count states in 3-D environments, however when these become visually more complex, pre-trained DINO representations are more effective thanks to the pre-trained inductive biases in the representations. Overall, this presents a pathway for integrating pre-trained biases into exploration. We evaluate our approach on the VizDoom and Habitat environments, demonstrating that our method surpasses other well-known exploration methods in these settings.

## 1. Introduction

Exploration in reinforcement learning can inherently be seen as a density estimation problem. Approximating the density of the state-space throughout learning, allows the agent to determine which parts of the environment it has visited. This paradigm is at the core of exploration via intrinsic motivation where an intrinsic reward signal is created to quantify novelty. A simple way to do this is by counting states (Tang et al., 2016). However, simply counting all states naively is not possible, since counts are very sparse in high-dimensional or stochastic settings and the memory requirements to store every distinct count for a given state are very high. There are several approaches to estimating the state density: One way is to use some density estimation model that given an observation assigns a pseudo-count (Bellemare et al., 2016), where a pseudo-count is an estimated count for a state given a density model. Other methods are based on using the prediction error, where the estimate for the state density is based on whether a model can predict fixed target features provided by a second model (Pathak et al., 2017; Burda et al., 2018; Raileanu & Rocktäschel, 2020). Established methods such as Random Network Distillation (Burda et al., 2018) show that novelty can be determined simply by calculating the prediction error of random features. Different states will provide different feature vectors hence the model learns an indirect density estimate of the state space. Recent methods such as BYOL-Explore (Guo et al., 2022) replace random features with trained features, with great success.

**The need for state aggregation in 3-D vs. 2-D.** Due to its inherent simplicity, counting states for exploration in non-tabular environments has been studied extensively (Bellemare et al., 2016; Tang et al., 2016; Ostrovski et al., 2017). While counting states for exploration works well in tabular settings, it becomes intractable in high dimensional settings if every state is counted explicitly. Bellemare et al. (2016) were among the first to propose the estimation of pseudo-counts, where the counts are modeled by some density model of the state-space. Essentially, this transforms the problem of exploration into a density estimation problem of the state-space. A central property for a density model in exploration is that the model should be able to distinguish different states, such that salient transitions can be identified in the intrinsic reward. In the case of counting states, the states should be aggregated in a sensible way such that the counts are not too granular, but also not too coarse (Ostrovski et al., 2017; Ecoffet et al., 2021). We identify a particular phenomenon regarding the importance of pixel changes

[1]Department of Computer Science, Heinrich Heine University Düsseldorf, Germany [2]Department of Computer Science, Technical University Dortmund, Germany. Correspondence to: Stefan Wagner <stefan.wagner@hhu.de>.

*Proceedings of the 41st International Conference on Machine Learning*, Vienna, Austria. PMLR 235, 2024. 

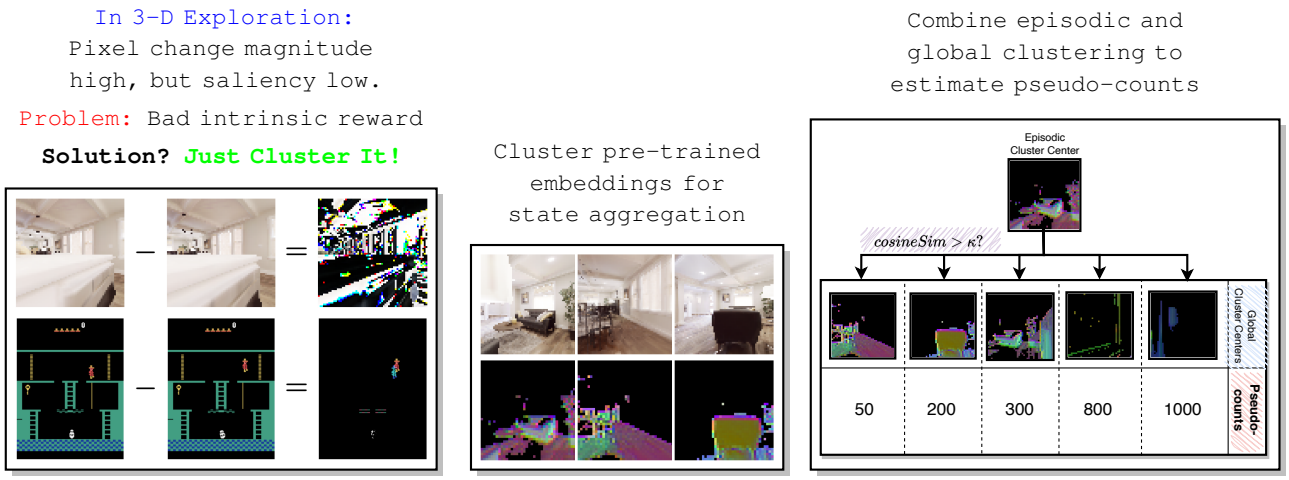



*Figure 1.* **"Just Cluster It" in a nutshell: (a)** Density estimation for exploration in 3-D environments is challenging, because the magnitude of pixel change is large, but the saliency of a single transition is low. This is in contrast to 2-D environments where every transition is distinct and therefore salient. Therefore, we propose clustering representations on an episodic and global level to estimate accurate pseudo-counts that reflect the state-space distribution. **(b)** We show pre-trained representations from a DINO model. The embeddings from the DINO model are able to extract relevant features from the observations, which is useful for clustering. The bottom images are thresholded embeddings. **(c)** To estimate pseudo-counts, we store episodic cluster centers in a global cluster table over time by matching episodic cluster centers with previously added cluster centers in the cluster table. A new cluster center is added to the table, if the cosine similarity to existing cluster centers is below a threshold $\kappa$.

in 3-D environments that requires states to be aggregated in order to provide a salient intrinsic reward. We provide a straightforward example in Figure 1(a), to demonstrate that state aggregation is necessary in 3-D environments compared to 2-D enviroments and thus must be accounted for by a density model in 3-D environments. We characterize this via transition saliency of pixels, i.e., the significance of pixel changes between two observations. For instance, in 2-D environments pixel change between transitions is usually small, but the significance of the pixel change is large. As a consequence, almost every observation contains new information regarding the current state of an agent. One example is Montezumas Revenge where individual groups of pixels change substantially when an action is performed, since the playable character figure changes its pixels when e.g., moving from left to right or when interacting with an object. In 3-D environments by contrast, although the magnitude of pixel change is large, these pixel changes are usually not as signficant, since after a single transition the relationship between the pixels in the observations remains largely the same. Rather, a more effective reference point for estimating the state density should be centered around salient features in the representations, such that observations with similar features are considered similar. In conclusion, in 2-D enviroments the transition saliency of pixels is high, whereas in 3-D environments the transition saliency of pixels is low. Thus, the underyling density model must account for this lack of saliency in 3-D environments to yield a meaningful intrinsic reward. Similar thoughts have also been presented by Ecoffet et al. (2021); Ostrovski et al. (2017). We argue that tackling this transition saliency problem may bridge the gap from count-based methods to prediction-error-based methods which excel at providing a salient intrinsic reward signal when observations are discrete and pixel changes are meaningful. However, they perform worse when the observations are high-dimensional and individual transitions are not as meaningful. Overall, if an appropriate density model can be found to estimate pseudo-counts in high dimensional observations, we can exploit the favorable theoretical guarantees of count-based methods (Bellemare et al., 2016) and avoid catastrophic forgetting in neural networks. With this in mind, we deduce that a clustering-based approach, which is able to find the right level of state aggregation by finding shared features in the observations, may be much more effective in high-dimensional environments, provided the underlying representations are expressive enough. Ideally, the level of state aggregation should be determined from the underlying features contained in the observations.

**Exploration through Episodic and Global Clustering of High-Dimensional Representations.** To this end, we propose a method for exploration that performs episodic and global clustering of (pre-trained) representations to estimate pseudo-counts. More precisely, in each episode we fit a Gaussian mixture model on random features or on features from a pre-trained DINO (Oquab et al., 2023; Caron

et al., 2021) model to exploit the invariances contained in the representations. The Gaussian mixture model determines cluster means, i.e., centers for the observations in the current episode, which can then be compared with cluster centers previously stored in memory (see Figure 1(c)). Particularly when utilizing pre-trained representations, as demonstrated in Figure 1(b), we see that these features align well with objects in the scene. We consider these features as various modes in the observational distribution, which makes them well suited for a Gaussian mixture model. The global clustering is then done in the cluster table, where a cosine similarity threshold $\kappa$ is used to determine whether a new episodic cluster center should be added to memory or not. This is akin to the stick breaking process in Dirichlet processes and thus allows us to control the granularity of the inter-episodic clustering. We then calculate the intrinsic reward for each observation in the episode as the inverse square root of the pseudo-counts (Strehl & Littman, 2008). We conduct experiments on both the VizDoom (Kempka et al., 2016) and Habitat (Savva et al., 2019; Szot et al., 2021; Puig et al., 2023) environments, demonstrating that even features chosen at random outperform other exploration methods when dealing with 3-D environments.

**Contributions**: **(i)** We introduce an exploration method that determines novelty by estimating pseudo-counts of clustered pre-trained representations. We propose a method that involves both episodic and global clustering. The episodic clustering aggregates similar representations by fitting a Gaussian mixture model to pre-trained DINO representations or to random features for every episode. The global clustering is done over time by matching and storing episodic cluster centers in a global cluster table that aggregates pseudo-counts. Empirically, we demonstrate the effectiveness of this procedure in the chosen 3-D environments.

**(ii)** We show that both random and pre-trained DINO features can be clustered effectively to aggregate states in 3-D environments and thus estimate pseudo-counts effectively. While random features work well on a simpler 3-D environment (VizDoom environment), with real-world observations (Habitat environment), the priors contained in the pre-trained DINO representations improve performance substantially.

**(iii)** By ablating two components of our method - the cosine similarity threshold $\kappa$ and the episodic clusters, we show the effectiveness of clustering observations in high-dimensional 3-D environments.

## 2. Background

**Count based methods and estimating pseudo-counts.** Pseudo-counts as defined by Bellemare et al. (2016) stand at the center of generalizing counts to high dimensional state spaces. They define state counts relative to density model that approximates the underlying state-space distribution. In principle, this definition of pseudo-counts is agnostic to the given state representation (Ostrovski et al., 2017). There are various exploration methods that rely on state counting, either to detect repeating observations or in order to normalize the intrinsic reward (Raileanu & Rocktäschel, 2020; Zhang et al., 2021; Wagner et al., 2023; Parisi et al., 2021; Tang et al., 2016). These methods ultimately propose pseudo-counts when extending to high-dimensional state-spaces. Bellemare et al. (2016) also demonstrate that estimating pseudo-counts using a density model provides a measure of either information gain, i.e., prediction gain. This shows that pseudo-counts are related to intrinsic reward methods that utilize prediction error as an intrinsic reward signal. When calculating an intrinsic reward bonus based on pseudo-counts, a common choice to calculate the intrinsic bonus is the inverse square root of the pseudo-counts as proposed by Strehl & Littman (2008).

**Novelty via prediction error.** Most recent exploration methods fall under the category of exploration via prediction error. The idea is that the prediction error of a model with regard to some target features is a measure of novelty. Various versions of these methods exist, like Random Network Distillation (Burda et al., 2018) (RND), in which the target features come from a neural network that is both fixed and initialized randomly. Pathak et al. (2017) learns inverse dynamics with a separate model that learns to predict the correct actions between state transitions which makes the learned representations invariant to uncontrollable parts of the enviroment. Raileanu & Rocktäschel (2020) predict the target features of the next state given the current state instead of predicting the target features of the same state. Recent methods such as BYOL-Explore (Guo et al., 2022) learn representations online with BYOL (Grill et al., 2020) leading to better target features. Overall, these models estimate the density of the state space through the target features they approximate. Thus, the target features influence the state-space density that is learned. An advantage of prediction-error-methods is that since they are parametrized by a neural network they can generate an intrinsic reward signal ad-hoc from raw observations.

**Density estimation via clustering representations.** A basic method for estimating the density of the state-space involves the use of clustering. Ma et al. (2019) cluster the observations directly by performing k-means on the observations. Then they calculate the pseudo-counts on an episodic level and do not store the counts throughout the entirety of learning. "Never Give Up" (NGU, Puigdomènech Badia et al., 2020) also make use of episodic clustering to estimate episodic novelty. For each episode, NGU determines the

sum of squared errors between the k-nearest neighbors of the current state and the current state itself. To capture long-term dependencies a global intrinsic reward is calculated via random network distillation. E3B (Henaff et al., 2022) performs episodic clustering by fitting a covariance matrix at each step and computing eigenvalues of the updated covariance matrix. The scale of the eigenvalues then determines the novelty. Apart from episodic clustering methods there are also methods that perform global clustering. Go-Explore (Ecoffet et al., 2021) maps observations to cells, which can also be seen as a form of clustering. However, instead of performing explicit clustering, a heuristic is used to find suitable values for downscaling parameters such as width, height and depth. The idea is that similar observations are mapped to the same cell. RECODE (Saade et al., 2023) attempts to estimate pseudo-counts by approximating clusters via Kernel Density Estimation (KDE). The current embedding is taken as central point and a density is fitted to find the nearest approximated cluster means stored in a table. If a new embedding is significantly different from its neighbors in the table it replaces an old embedding. RECODE is closely related to our method, with its Dirichlet process inspired approach. However, RECODE is also somewhat susceptible to the problem of transition saliency in 3-D environments, since it updates the cluster means at every step with only a single embedding as reference point. We also simplify the stick breaking process for the cluster table, as we only have a single threshold $\kappa$ that determines whether a new cluster center should be added to the cluster table or not.

## 3. Just Cluster It: Exploration via Episodic and Global Clustering

In this section, we describe our exploration method "Just Cluster It!", which constists of a two-stage clustering: first, we cluster after each episode, which we call *episodic clustering*, and second, we collect the newly found clusters in a global table of cluster centers, which we call *global clustering*. The cluster centers of the global clustering maintain a list of pseudo-counts to approximate the state space distribution. Therefore, it is important that the episodic cluster centers are representative of the observational distribution in the current episode, in order to aggregate the pseudo-counts correctly. Over time, the cluster table will contain pseudo-counts for the whole state-space distribution. Note that we call the counts *pseudo*, since they are defined relative to a mixture model.

For each episode, we employ a two-step process: first, we perform episodic clustering by fitting a Gaussian mixture model on the batch of representations of the current episode. This yields several episodic cluster centers. The second step updates the global cluster table that contains cluster centers from previous episodes and their corresponding counts. For each episode, counts from episodic cluster centers are then either aggregated to existing cluster centers, or a new cluster center is added depending on a cosine similarity threshold. This process of adding cluster centers continuously is similar to the stick breaking procedure in Dirichlet processes. However, instead of sampling weights from a beta distribution to determine the new cluster size, we simply add a new cluster center if the maximum cosine similarity is below a threshold $0 \leq \kappa \leq 1$. In this sense, $\kappa$ controls the granularity of the clusters in the cluster table and is related to the concentration parameter for the Beta distribution in Dirichlet processes. The closer $\kappa$ is to one, the more cluster centers are added to the cluster table as shown in Figure 4(a). The key insight resulting from our method is that (pre-trained) representations in high dimensions are expressive enough to be clustered effectively on an episodic level, while also contributing to the approximation of the global state space distribution.

**Notation and Preliminaries.** We define an MDP as a tuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma)$ with interactions at discrete time steps $t = 0, 1, 2, 3, \ldots$. The agent receives a state from the environment as a random variable $S_t \in \mathcal{S}$, where $S_t = s$ is some representation of the state from the set of states $\mathcal{S}$ at timestep $t$. The agent then selects an action $A_t \in \mathcal{A}$ where $A_t = a$ is some action in the possible set of actions $\mathcal{A}$ for the agent at timestep $t$. This action is selected according to a policy $\pi(a|s)$ or $\pi(s)$ if the policy is deterministic. One time step later after taking the action, the agent receives a numerical reward which is a random variable $R_{t+1} \in \mathbb{R}$, where $R_{t+1} = r$ is some numerical reward at timestep $t + 1$. Finally, the agent is in a new state $S_{t+1} = s'$. As usual $\gamma$ is the discount factor. Furthermore, we define a POMDP $(\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{P}, \mathcal{R}, \mathcal{Q}, \gamma)$ for egocentric 3-D environments, where the true state $s$ is not known. Thus, the agent sees the state $s \in \mathcal{S}$ through an observation $o \in \mathcal{O}$, where an observation function $\mathcal{Q} : \mathcal{S} \to \mathcal{O}$ maps the true state to the agent's observation. The components of our method are as follows: we define a policy network $\pi_\theta : \mathcal{O} \to \mathcal{A}$ with network parameters $\theta$, which transforms an observation $o$ into an action $a$.

We utilize a pre-trained model $f_\nu : \mathcal{O} \to \mathcal{E}$ with network parameters $\nu$ to project observations into a latent space $\mathcal{E}$, which will be learned by self-supervised learning (e.g. DINO, Caron et al., 2021). A clustering algorithm $g : \mathcal{E} \to \{1, \ldots M\}$ is also defined, which assigns a cluster label $m$ to an embedding $e$.

Our method has two hyperparameters: the cosine similarity threshold $\kappa$ and the number of clusters $M$ for the episodic Gaussian mixture model. This is less than other exploration methods based on clustering. In the following we describe

our algorithm. We provide the Python implementation [1] for the following steps in Algorithm 1.

1. **Building episodic clusters.** We define an episode as a sequence of transitions $\{(o^{(t)}, a^{(t)}, r^{(t)}, o'^{(t)})\}_{t=1}^{T}$, where $T$ is the length of the episode and $o'^{(t)}$ is the observation following $o^{(t)}$. Moreover, we define the set of observations as $B = \{o^{(t)}\}_{t=1}^{T}$ and let $\tilde{B} = \{e^{(t)}\}_{t=1}^{T} = f_\nu(B)$ be the embeddings after passing them through the pre-trained model $f_\nu$. We then fit a Gaussian mixture model (GMM) on the embeddings $\tilde{B}$ to obtain $M$ episodic cluster centers

$$\{\mu_{\text{ep}}^{(m)}\}_{m=1}^{M}. \tag{1}$$

The GMM induces the mapping $g : \mathcal{E} \to \{1, \dots M\}$ which assigns each embedding $e^{(t)}$ to one of the clusters, which defines a sequence of cluster labels $m_*(t) = g(e^{(t)})$.

2. **Updating the global cluster table.** The following steps 2.1 to 2.4 are repeated for every episodic cluster $m = 1, \dots, M$. This builds the global cluster table incrementally.

   2.1 **Assigning an episodic cluster center to global cluster centers.** Before the first episode, the global cluster table is empty. In step 2.3 new pairs will be added. Assume for now that we have already $K$ pairs $(\mu_{\text{global}}^{(k)}, \lambda_{\text{global}}^{(k)})$ of global cluster centers and global cluster pseudo counts

$$\{(\mu_{\text{global}}^{(k)}, \lambda_{\text{global}}^{(k)})\}_{k=1}^{K}. \tag{2}$$

   Next, we calculate for fixed $m$ the pairwise cosine similarities between $\mu_{\text{ep}}^{(m)}$ and $\mu_{\text{global}}^{(k)}$, which are collected in a $K$-dimensional vector D. The maximum of $D$ gives us the maximal cosine similarity $d_*(m)$ and a corresponding global cluster label $k_*(m) \in \{1, \dots, K\}$ for each episodic cluster center.

   2.2 **Calculating the intrinsic rewards.** To calculate the intrinsic rewards along the episode, we assign the pseudo-counts for the embeddings $e^{(t)}$ that are assigned to cluster $m$,

$$\rho^{(t)} = \lambda_{\text{global}}^{(k_*(m))} + \sum_{t'=1}^{t} [m_*(t') = m] \tag{3}$$

   such that the pseudo-counts increase for each time an embedding belonging to the cluster $m$ is seen in the episode.

   For time steps $t$ where $\rho^{(t)}$ has just been assigned, we calculate the intrinsic rewards as the inverse square root of the pseudo-counts (Strehl & Littman, 2008),

$$r_{\text{int}}^{(t)} = \frac{1}{\sqrt{\rho^{(t)}}}. \tag{4}$$

   2.3 **Growing the global cluster table.** If the maximal cosine similarity $d_*(m)$ is smaller than the hyperparameter $\kappa$ (a fixed value between 0 and 1), we add $\mu_{\text{ep}}^{(m)}$ to the list of global cluster centers and $K$ is increased by one. The corresponding global pseudo count is initialized to zero, but will get updated in the next step.

   2.4 **Updating the global pseudo counts.** The global pseudo count of the global cluster center that corresponds to $m$, i.e., with index $k_*(m)$, is simply updated using the episodic cluster counts for the assigned episodic cluster center $m$:

$$\lambda_{\text{global}}^{(k_*(m))} := \lambda_{\text{global}}^{(k_*(m))} + \lambda_{\text{ep}}^{(m)}. \tag{5}$$

3. **Combining intrinsic and extrinsic rewards.** Finally, the intrinsic and extrinsic rewards are simply added, and we update the policy network $\pi_\theta$ with the transitions

$$\{(o^{(t)}, a^{(t)}, r_{\text{int}}^{(t)} + r^{(t)}, o'^{(t)})\}_{t=1}^{T} \tag{6}$$

   via Proximal Policy Optimization (PPO).

[1] We release our code on Github: https://github.com/stefanwm13/just-cluster-it-public

## 4. Experiments

Given our proposed ideas and method we aim to answer the following questions: (i) How well does clustering features perform in high-dimensional 3-D environments? (ii) Can our method leverage pre-trained representations to improve exploration? (iii) What effect do episodic clustering and global clustering have on the performance and robustness of our method? We present the environments, experiment setup and baselines, followed by results of our experiments and corresponding ablations.

**Environments.** We test our method on observationally complex environments like VizDoom (Kempka et al., 2016) and Habitat-Replica (Savva et al., 2019; Szot et al., 2021; Puig et al., 2023). In VizDoom every room has a different texture, where some moving textures are also present. In Habitat, the observations are real world scenes of apartments, hotels and office rooms which are very complex. The chosen environments are centered around navigation exploration tasks. This setup enables us to separate the issue of density estimation from the challenge of understanding complex environment dynamics. In general, all methods

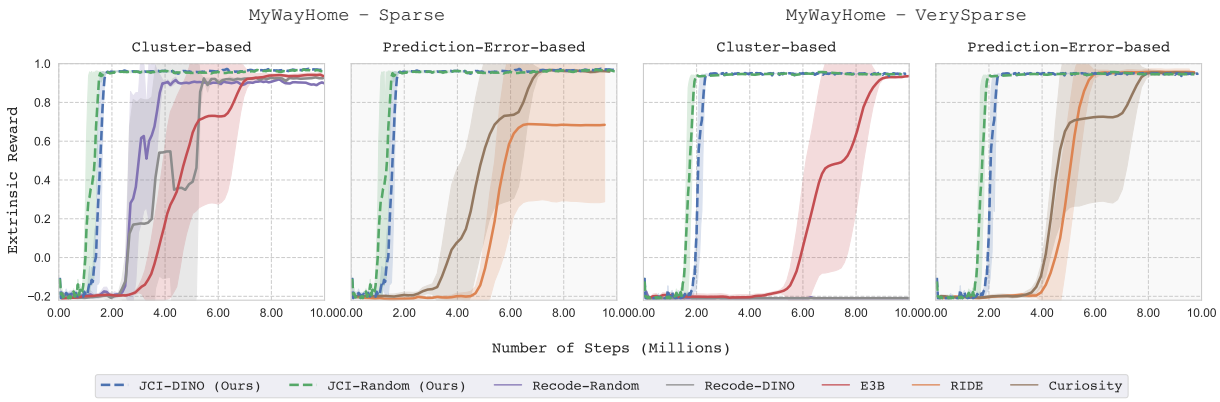



*Figure 2.* **Clustering random features is effective for VizDoom:** Our method with clustering outperforms other traditional exploration methods. Interestingly, random features perform slightly better than the DINO features for VizDoom. While the complexity of the observations is not trivial, in this setting the priors in the pre-trained representations seem to be not more informative than random features. This also shows that in 3-D high-dimensional environments random features are salient enough when clustered to estimate pseudo-counts.

we evaluate including ours focus solely on the problem of density estimation, and we thus share the same assumptions. More details regarding the environments and agent setup can be found in the Appendix D.

**Experimental setup and baselines.** We compare our method to both clustering and prediction-error-based exploration approaches. Thus, we split the experiments into two categories for both environments. Both Figures 2 and 3 show comparison to only (i) clustering-based methods on the left side and to (ii) prediction-error-based methods on the right side. For (i) we compare against RECODE (Saade et al., 2023) and E3B (Henaff et al., 2022). For (ii) we compare against Random Network Distillation (Burda et al., 2018), Curiosity-driven exploration (Pathak et al., 2017) and RIDE (Raileanu & Rocktäschel, 2020). To see how other methods leverage pre-trained representations, we also compare against RECODE with both random features and DINO features. For all configurations we perform 3 runs with different seeds. We use PPO (Schulman et al., 2017) to train the policy network for both our method and our implementation of RECODE. E3B (Henaff et al., 2023) also uses DD-PPO for the Habitat environment. For all other methods the learning method for the policy network is IMPALA (Espeholt et al., 2018). Specific implementation details and hyperparameters can be found in Appendix E.

**Evaluation metrics.** For VizDoom we use the extrinsic reward as a metric for performance. The extrinsic reward is defined from 0 to 1. For Habitat, we explore only using the intrinsic reward, we thus use the number of visited states as a metric for performance as in (Parisi et al., 2021). We show the exact formula in Appendix D.2.

### 4.1. VizDoom Results—The Suprising Effectiveness of Random Features

We plot results for "MyWayHome" in Figure 2, where we compare our method to both cluster-based (left) and prediction-error-based methods (right) for both the "Sparse" and "VerySparse" settings. We see that clustering random features with our method (dashed, green curve) is superior to all other methods converging at around 1.5M and 1.8M steps respectively. Interestingly, random features perform slightly better in VizDoom than the DINO features (dashed, blue curve) which converges slightly later at around 1.7M and 1.9M steps respectively. In this case, the priors in the DINO features do not seem to pose an advantage over random priors for the observations in the VizDoom environment. However, this also shows that random features in 3-D environments can be salient enough to be clustered effectively. When comparing our method to RECODE with DINO features (grey curve), we see that our method is better able to leverage the pre-trained features. For RECODE, DINO features converge at around 5M steps compared to the random features (purple curve) which converge at 3.8M steps, while for our method random features and DINO features converge at almost the same time. For "VerySparse" we could not get RECODE to converge within the specified number of steps. One explanation for this could be that in the "VerySparse" setting, the agent never reaches the goal by chance as opposed to in the "Sparse" setting, such that the agent never finds a sensible policy. The RECODE agent does however manage to reach the goal on a few occasions. Furthermore, comparing our method to the prediction-error-based methods such as RIDE and Curiosity, we see that these methods perform worse needing at least 5M steps to converge. Overall, both our clustering method and RECODE, outperform the prediction-error-based meth-

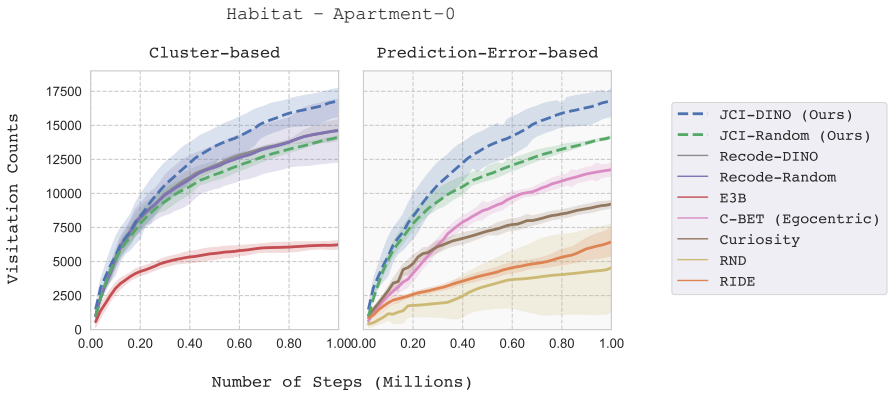



*Figure 3.* **Pre-trained DINO features excel with more complex observations:** When training on the Habitat environment we see that clustering with DINO embeddings is more effective than clustering with random features. Furthermore, only our method is able to leverage the pre-trained DINO features effectively. We argue that the priors present in the DINO embeddings help in agreggating the representations when building the episodic clusters, which ultimately determine the pseudo-counts.

ods. While E3B is a clustering method as well, it only performs episodic clustering as explained in Section 2. Usually, episodic clustering alone performs worse in singleton environments, such as VizDoom as explained in Henaff et al. (2023).

### 4.2. Habitat Results—Pre-trained features help in complex observations

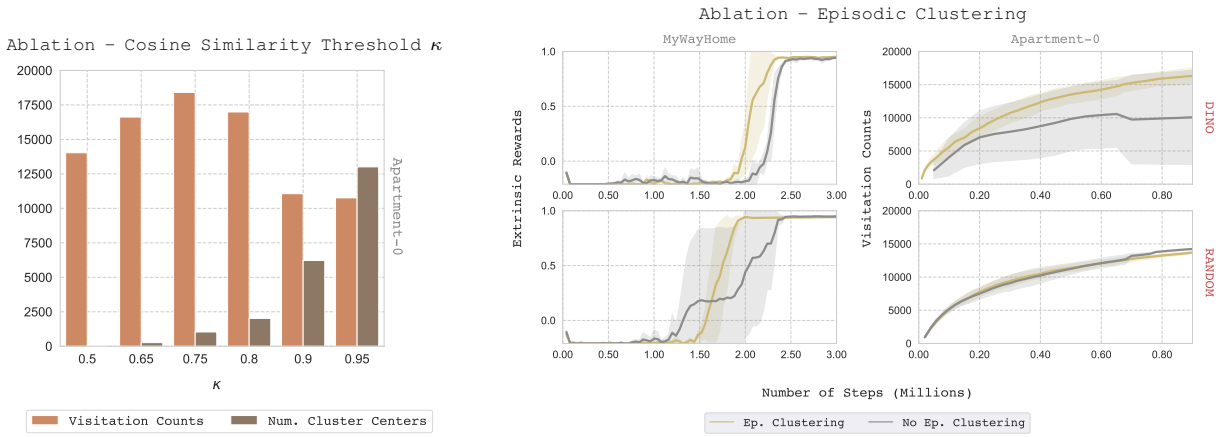
To test our method on more complex observations we use the Habitat environment and adopt the experimental setting from Parisi et al. (2021). That is, we train the agent on the Habitat environment with only intrinsic rewards and plot the visitation counts (see Figure 3). We can see the benefit of using pre-trained DINO features in the visually more complex Habitat environment. When clustering pre-trained DINO features with our method (dashed, blue curve), we achieve 17K visitation counts on average, more than any other method. RECODE by comparison achieves only slightly under 15K visitation counts. Interestingly, our method with random features (dashed, green curve) performs slightly worse than RECODE at 14K visitation counts. We show in Figure 4(b) that for Habitat, episodic clustering of random features is not better than episodic clustering of DINO features, hence the reduced performance compared to DINO features. However, we see again that our method is able to better leverage the pre-trained features compared to RECODE. Again, E3B is not suited for singleton environments barely achieving 7K visitation counts. Looking at the prediction-error-based methods, although Random Network Distillation uses random features as well, it is vastly inferior to the clustering-based methods converging at around 5K visitation counts. This validates our hypothesis about *transition saliency* presented in Section 1: In 3-D environments, the clustering-based methods are able to exploit the structure of the representations more effectively. We will expand on this in the following section.

### 4.3. The Effect of Episodic and Global Clustering

Next, we analyse the effect of the episodic clustering and the global clustering. To ablate the episodic clustering, we perform runs for both "MyWayHome-VerySparse" and "Apartment-0" with and without episodic clustering. To ablate the influence of the global clustering, we vary the cosine similarity threshold $\kappa$ and plot the number of cluster centers and visitation counts for "Apartment-0".

**Increasing cosine similarity threshold $\kappa$ makes pseudo-counts more granular, degrading performance.** We plot the visitation counts and the number of clusters in the cluster table for different values of $\kappa$ in Figure 4(a). We see that increasing the cosine similarity threshold $\kappa$ increases the number of global cluster centers in the cluster table. If the number of clusters in the cluster table is too high, performance decreases as shown by lower visitation counts for higher values of $\kappa$. We argue that the pseudo-counts become too sparse, and thus the intrinsic reward fails to provide a good approximation of the state-space distribution. This aligns with the *transition saliency* problem in Section 1. In this case we revert to the same setting of prediction-error-based methods where the intrinsic reward becomes too granular. Note that the performance at around 10K visitation counts is still better than random network distillation (Burda et al., 2018) in Figure 3, which is likely related to the fact that clustering-based methods are not affected by catastrophic forgetting.

Interestingly, for lower values of $\kappa$ performance does not degrade in the same way. We observe that the smaller the value of $\kappa$ is the less cluster centers get added to the cluster




(a) (b)

*Figure 4.* **Ablation studies: We show the effectiveness of clustering, especially episodic clustering: (a)** Increasing the cosine similarity threshold $\kappa$ increases the number of global clusters in the cluster table. If the number of clusters in the cluster table are too granular, performance decreases as shown by lower visitation counts for higher values of $\kappa$. **(b)** Episodic clustering helps whenever there is structure in the representations. Especially for DINO in Habitat performance drops significantly, if episodic clustering is removed. This can also be seen in VizDoom (left panels) where episodic clustering helps both DINO and random features. For Habitat (right panels) with random features, episodic clustering has no large effect since the random features are not expressive enough for the complex observations.

table. Hence the intrinsic reward is defined over fewer cluster centers. We see in Figure 4(a) that up to $\kappa = 0.5$ performance remains comparable.

**Episodic clustering is effective with good representations.** We plot visitation counts and extrinsic reward with and without episodic clustering in Figure 4(b). We see that episodic clustering helps whenever the underlying representations are structured. This can be seen for Habitat (right panels) with DINO features where performance drops dramatically without episodic clustering (upper right). Even for VizDoom (left panels), episodic clustering helps both with DINO and random features, which shows that there is hidden structure to exploit in the comparatively simpler observations. For Habitat with random features, episodic clustering has no big effect since the random features are probably not expressive enough for the complex observations. We see this in Figure 9 (in the Appendix) where the random features are not clustered effectively compared to the DINO features. Overall, during our tests we found that 10% of the total number of steps in an episode is a good number of clusters for the Gaussian mixture model.

### 4.4. Clustering is Robust to Environmental Noise

To the test the effect of random noise on exploration like in the noisy-TV problem (Pathak et al., 2017), we adopt the experimental setup of Saade et al. (2023), where a frame of random noise is concatenated to the current observation. Note that for each step the different frame with random noise is sampled. We show an example of the observation with the concatenated noise in Figure 6.

**Random noise does not affect exploration for clustering based methods.** We rerun the Habitat Apartment-v0 with noisy observations for both our method and RECODE in Figure 7. We show that our method is unaffected by random noise in the observations. Interestingly, performance slightly increases with noisy observations for both our method and RECODE. When looking at the number of cluster entries, we observe that when using observations with random noise the amount of clusters in the cluster table is lower than without the random noise. We therefore suspect that the random noise has a regularising effect on the clustering.

We also show the saved cluster centers in the cluster table with noisy observations in Figure 5. The quality of the clustering seems unaffected since similar observations are still aggregated correctly despite the random noise.

## 5. Conclusion

The starting point of our work is the insight that the magnitude of pixel-changes can be large in 3-D environments, but usually lacks in saliency. This is problematic in exploration with sparse rewards, since the agent is not able to distinguish between important and unimportant changes in the observations. To mitigate this problem, we proposed a method for state aggregation based on episodic and global clustering of pre-trained representations. We have shown that pre-trained

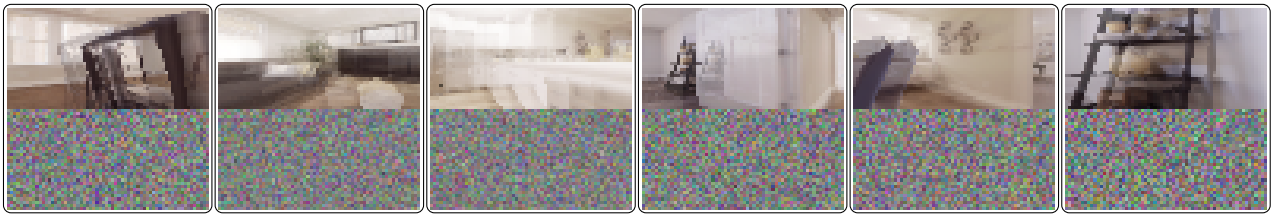


*Figure 5.* **Visualization of DINO clusters for observations with random noise:** We plot different samples from the cluster table for the noisy observations in Habitat. Each image represents the mean image of an episodic cluster that was aggregated to the cluster table. The clustering happens on the 384-dimensional embedding. Even with the concatenated random noise the embeddings are still clustered sensibly.


*Figure 6.* **We concatenate random noise to the observations:** To test the robustness of our method to observations with random noise, i.e, the noisy TV problem, we perform the same experiment as Saade et al. (2023) and concatenate a frame with random noise that changes after every step to the observation.

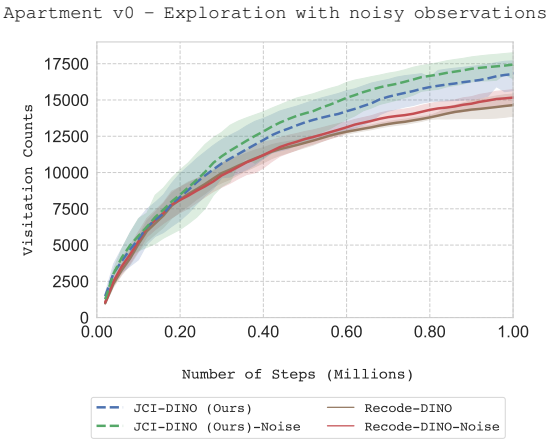



*Figure 7.* **Our method is robust to random noise in the observations:** We see that our method is robust to random noise in the observations. As a sanity check we also evaluate RECODE on noisy observations and confirm the results from Saade et al. (2023). Suprisingly, for both methods the performance increases when adding random noise. We suspect the random noise has a regularising effect on the clustering.

representations and also random features can be leveraged effectively in environments with complex observations. We ablated different components of our method: we show that the cosine similarity threshold $\kappa$ is vital to control the granularity of the clusters in the cluster table. We notice that performance degenerates if the number of clusters in the cluster table is too high, since the pseudo-counts become too sparse and the intrinsic reward too noisy. Furthermore, we found out that episodic clustering leverages the priors in the representations in order to produce better pseudo-count estimates, thus playing a crucial role for complex observations since the counts for the corresponding representations are aggregated more effectively.

**Limitations and future work.** So far, we tested our method on singleton MDP's. We believe our method can also be extended to contextual MDPs, possibly by counting states separately on an episodic level. Overall, our method sheds light onto the problem of transition saliency in 3-D environments which is ubiquitous in exploration with sparse-rewards. In this sense, an interesting direction for future work could be investigating whether mitigating the transition saliency problem can enhance the effectiveness of prediction-error-based methods in 3-D enviroments. Hypothetically, clustering representations and using them as fixed targets for prediction-error-based methods should result in a more salient intrinsic reward. Future research should continue investigating which kinds of representations contain the best priors for exploration with sparse rewards. Potentially, multi-modal generative models could be used to learn representations that not only capture visual invariances, but also environmental dynamics.

## Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

## References


Bellemare, M. G., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying Count-Based Exploration and Intrinsic Motivation. *arXiv e-prints*, art. arXiv:1606.01868, June 2016. doi: 10.48550/arXiv.1606.01868.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by Random Network Distillation. *arXiv e-prints*, art. arXiv:1810.12894, October 2018. doi: 10.48550/arXiv.1810.12894.

Caron, M., Touvron, H., Misra, I., Jégou, H., Mairal, J., Bojanowski, P., and Joulin, A. Emerging Properties in Self-Supervised Vision Transformers. *arXiv e-prints*, art. arXiv:2104.14294, April 2021. doi: 10.48550/arXiv.2104.14294.

Ecoffet, A., Huizinga, J., Lehman, J., Stanley, K. O., and Clune, J. First return, then explore. *nat*, 590(7847):580–586, February 2021. doi: 10.1038/s41586-020-03157-9.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., Dunning, I., Legg, S., and Kavukcuoglu, K. IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures. *arXiv e-prints*, art. arXiv:1802.01561, February 2018. doi: 10.48550/arXiv.1802.01561.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P. H., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z. D., Gheshlaghi Azar, M., Piot, B., Kavukcuoglu, K., Munos, R., and Valko, M. Bootstrap your own latent: A new approach to self-supervised Learning. *arXiv e-prints*, art. arXiv:2006.07733, June 2020. doi: 10.48550/arXiv.2006.07733.

Guo, Z. D., Thakoor, S., Pîslar, M., Avila Pires, B., Altché, F., Tallec, C., Saade, A., Calandriello, D., Grill, J.-B., Tang, Y., Valko, M., Munos, R., Gheshlaghi Azar, M., and Piot, B. BYOL-Explore: Exploration by Bootstrapped Prediction. *arXiv e-prints*, art. arXiv:2206.08332, June 2022. doi: 10.48550/arXiv.2206.08332.

Henaff, M., Raileanu, R., Jiang, M., and Rocktäschel, T. Exploration via Elliptical Episodic Bonuses. *arXiv e-prints*, art. arXiv:2210.05805, October 2022. doi: 10.48550/arXiv.2210.05805.

Henaff, M., Jiang, M., and Raileanu, R. A Study of Global and Episodic Bonuses for Exploration in Contextual MDPs. *arXiv e-prints*, art. arXiv:2306.03236, June 2023. doi: 10.48550/arXiv.2306.03236.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pp. 341–348, Santorini, Greece, Sep 2016. IEEE. doi: 10.1109/CIG.2016.7860433. The Best Paper Award.

Ma, X., Zhao, S.-Y., and Li, W.-J. Clustered Reinforcement Learning. *arXiv e-prints*, art. arXiv:1906.02457, June 2019. doi: 10.48550/arXiv.1906.02457.

Oquab, M., Darcet, T., Moutakanni, T., Vo, H., Szafraniec, M., Khalidov, V., Fernandez, P., Haziza, D., Massa, F., El-Nouby, A., Assran, M., Ballas, N., Galuba, W., Howes, R., Huang, P.-Y., Li, S.-W., Misra, I., Rabbat, M., Sharma, V., Synnaeve, G., Xu, H., Jegou, H., Mairal, J., Labatut, P., Joulin, A., and Bojanowski, P. DINOv2: Learning Robust Visual Features without Supervision. *arXiv e-prints*, art. arXiv:2304.07193, April 2023. doi: 10.48550/arXiv.2304.07193.

Ostrovski, G., Bellemare, M. G., van den Oord, A., and Munos, R. Count-Based Exploration with Neural Density Models. *arXiv e-prints*, art. arXiv:1703.01310, March 2017. doi: 10.48550/arXiv.1703.01310.

Parisi, S., Dean, V., Pathak, D., and Gupta, A. Interesting object, curious agent: Learning task-agnostic exploration. In Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=knKJgksd7kA.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven Exploration by Self-supervised Prediction. *arXiv e-prints*, art. arXiv:1705.05363, May 2017. doi: 10.48550/arXiv.1705.05363.

Puig, X., Undersander, E., Szot, A., Cote, M. D., Partsey, R., Yang, J., Desai, R., Clegg, A. W., Hlavac, M., Min, T., Gervet, T., Vondrus, V., Berges, V.-P., Turner, J., Maksymets, O., Kira, Z., Kalakrishnan, M., Malik, J., Chaplot, D. S., Jain, U., Batra, D., Rai, A., and Mottaghi, R. Habitat 3.0: A co-habitat for humans, avatars and robots, 2023.

Puigdomènech Badia, A., Sprechmann, P., Vitvitskyi, A., Guo, D., Piot, B., Kapturowski, S., Tieleman, O., Arjovsky, M., Pritzel, A., Bolt, A., and Blundell, C. Never Give Up: Learning Directed Exploration Strategies. *arXiv e-prints*, art. arXiv:2002.06038, February 2020. doi: 10.48550/arXiv.2002.06038.

Raileanu, R. and Rocktäschel, T. RIDE: Rewarding Impact-Driven Exploration for Procedurally-Generated Environments. *arXiv e-prints*, art. arXiv:2002.12292, February 2020. doi: 10.48550/arXiv.2002.12292.

Saade, A., Kapturowski, S., Calandriello, D., Blundell, C., Sprechmann, P., Sarra, L., Groth, O., Valko, M., and Piot, B. Unlocking the Power of Representations in Long-term Novelty-based Exploration. *arXiv e-prints*, art. arXiv:2305.01521, May 2023. doi: 10.48550/arXiv.2305.01521.

Savva, M., Kadian, A., Maksymets, O., Zhao, Y., Wijmans, E., Jain, B., Straub, J., Liu, J., Koltun, V., Malik, J., Parikh, D., and Batra, D. Habitat: A Platform for Embodied AI Research. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2019.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal Policy Optimization Algorithms. *arXiv e-prints*, art. arXiv:1707.06347, July 2017. doi: 10.48550/arXiv.1707.06347.

Straub, J., Whelan, T., Ma, L., Chen, Y., Wijmans, E., Green, S., Engel, J. J., Mur-Artal, R., Ren, C., Verma, S., Clarkson, A., Yan, M., Budge, B., Yan, Y., Pan, X., Yon, J., Zou, Y., Leon, K., Carter, N., Briales, J., Gillingham, T., Mueggler, E., Pesqueira, L., Savva, M., Batra, D., Strasdat, H. M., Nardi, R. D., Goesele, M., Lovegrove, S., and Newcombe, R. The Replica dataset: A digital replica of indoor spaces. *arXiv preprint arXiv:1906.05797*, 2019.

Strehl, A. L. and Littman, M. L. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 74(8):1309–1331, 2008. ISSN 0022-0000. doi: https://doi.org/10.1016/j.jcss.2007.08.009. URL https://www.sciencedirect.com/science/article/pii/S0022000008000767. Learning Theory 2005.

Szot, A., Clegg, A., Undersander, E., Wijmans, E., Zhao, Y., Turner, J., Maestre, N., Mukadam, M., Chaplot, D., Maksymets, O., Gokaslan, A., Vondrus, V., Dharur, S., Meier, F., Galuba, W., Chang, A., Kira, Z., Koltun, V., Malik, J., Savva, M., and Batra, D. Habitat 2.0: Training home assistants to rearrange their habitat. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Chen, X., Duan, Y., Schulman, J., De Turck, F., and Abbeel, P. #Exploration: A Study of Count-Based Exploration for Deep Reinforcement Learning. *arXiv e-prints*, art. arXiv:1611.04717, November 2016. doi: 10.48550/arXiv.1611.04717.

Wagner, S. S., Arndt, P., Robine, J., and Harmeling, S. Cyclophobic Reinforcement Learning. *arXiv e-prints*, art. arXiv:2308.15911, August 2023. doi: 10.48550/arXiv.2308.15911.

Zhang, T., Xu, H., Wang, X., Wu, Y., Keutzer, K., Gonzalez, J. E., and Tian, Y. Noveld: A simple yet effective exploration criterion. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 25217–25230. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/d428d070622e0f4363fceae11f4a3576-Paper.pdf.

## A. Algorithm

```
# Hyperparameters
kappa = 0.8 # Cosine similarity threshold
M = 250 # Number of episodic clusters

# Global variables
self.gmm = GaussianMixture(n_components=M, reg_covar=1e-6, covariance_type="full",
  random_state=1, warm_start=False)
self.cluster_table = [np.zeros(embedding_size), 0] # (Equation 2)

def update_cluster_table(self, embeddings, rewards):
  # Step 1 (Build episodic clusters)
  self.gmm.fit(embeddings) # Fit GMM (Equation 1)
  m_prime = self.gmm.predict(embeddings) # Get cluster labels

  # Assign embeddings to episodic clusters
  cluster_dict = {}
  for index, embedding in enumerate(embeddings):
    if m_prime[index] not in cluster_dict:
      cluster_dict[m_prime[index]] = []
    cluster_dict[m_prime[index]].append([embedding])

  # Step 2 (Updating the global cluster table)
  for m in cluster_dict:
    episodic_cluster = np.stack(cluster_dict[m])
    lambda_ep = episodic_cluster.shape[0]

    # Calculate episodic cluster center
    mu_ep = np.mean(np.stack(episodic_cluster[:, 0]), 0) # (Equation 1)
    mu_globals = np.stack(self.cluster_table[:, 0])

    # Step 2.1 (Assigning episodic cluster center(s) to global cluster centers)
    distances = cosine_similarity(np.expand_dims(mu_ep, 0), mu_globals)
    k_prime = np.argmax(distances)
    d_prime = np.max(distances)

    # Step 2.2 (Calculating intrinsic rewards)
    iverson_bracket_count = 0
    max_lambda_global = self.cluster_table[k_prime, 1]
    lambda_global = 0 if d_prime < kappa else max_lambda_global

    for index, _ in enumerate(embeddings):
      # (Equation 3)
      if m_prime[index] == m:
        iverson_bracket_count += 1
        rho = lambda_global + iverson_bracket_count

        # (Equation 4)
        ir = 1 / np.sqrt(rho)
        rewards[index] += 0.1*ir # Scale intrinsic reward with 0.1

    # Step 2.3 (Growing the cluster table)
    if d_prime < kappa:
      self.cluster_table = np.vstack((self.cluster_table, [mu_ep, lambda_ep]))
    else:
      # Step 2.4 (Updating the global pseudo counts)
      self.cluster_table[k_prime, 1] += lambda_ep  # (Equation 5)

  return rewards # Step 3
```

*Listing 1.* Just Cluster It! Algorithm

# B. Visualizations

## B.1. What do the clusters look like?

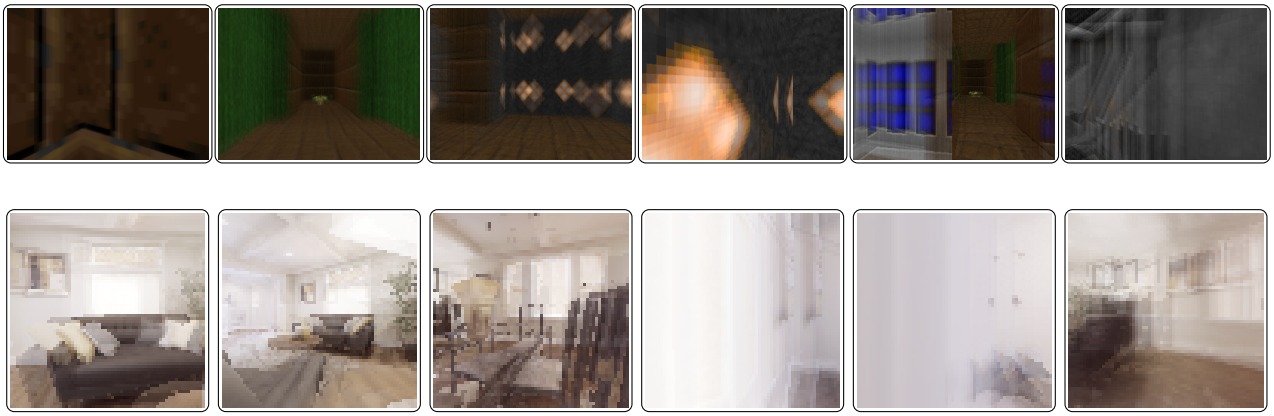


*Figure 8.* **Visualization of DINO clusters for VizDoom and Habitat:** We plot different samples from the cluster table for VizDoom. Each image represents the mean image of an episodic cluster that was aggregated to the cluster table. The clustering happens on the 384-dimensional embedding. We can see that observations are aggregated sensibly and account for similar states, where having a separate pseudo-count would not be beneficial. By looking at the counts and appearances we can also see that the clusters are not too granular, since the counts are not too low and the same cluster center is encountered multiple times across different episodes.

We visualize some of the clusters centers stored in the global cluster table, which are technically episodic clusters centers that were aggregated to the global cluster table. We show the mean image of each cluster center in Figure 8 for VizDoom and for Habitat. Note that the actual clustering was performed with the 384-dimensional DINO embedding, but we show the images for better interpretability. We can see that the clusters are aggregated sensibly and account for similar states, where having a separate pseudo-count would not be beneficial. Especially for Habitat, the complex observations are aggregated to similar states, which is a testament to the effectiveness of the DINO features.

## B.2. Clustering DINO vs Random Features

We compare the clusters from DINO (magenta) and random features (cyan) in Figure 9. We can see that the episodic clusters from DINO and random features are fundamentally different. Overall, the DINO cluster centers are much less granular compared to the random features. This can be seen by the fact that the cluster mean images for the random features are free from distortions. Moreover, some cluster centers for the random features are very similar to each other, which is harder to find for the DINO features. From the counts we can see that random features still are able to aggregate counts, however very similar observations may not be aggregated as well as with the DINO clusters. This can be seen by the fact that some counts are unusually high for the random features indicating that some episodic clusters may be assigned erroneously to the same global cluster center. We were not able to find such high counts for the DINO features. Overall, we see that episodic clustering together with global clustering is effective for both DINO and random features, since episodic cluster centers are matched multiple times throughout learning to the global cluster centers, as can be seen by the number of appearances "Apps" in Figure 9.


*Figure 9.* **Comparison of DINO clusters and Random clusters:** From the observations in the Habitat environment we plot different samples from the cluster table for DINO (magenta) and random features (cyan). The episodic clusters from DINO and random features are fundamentally different. While the DINO clusters aggregate more dissimilar observations that share common features, the random clusters are more granular seen by the fact that the cluster mean images are free from distortions. From the counts we can see that random features still are able to aggregate counts, however very similar observations may not be aggregated as well as with the DINO clusters.

## C. Further Ablations

### C.1. Comparing DINO model size

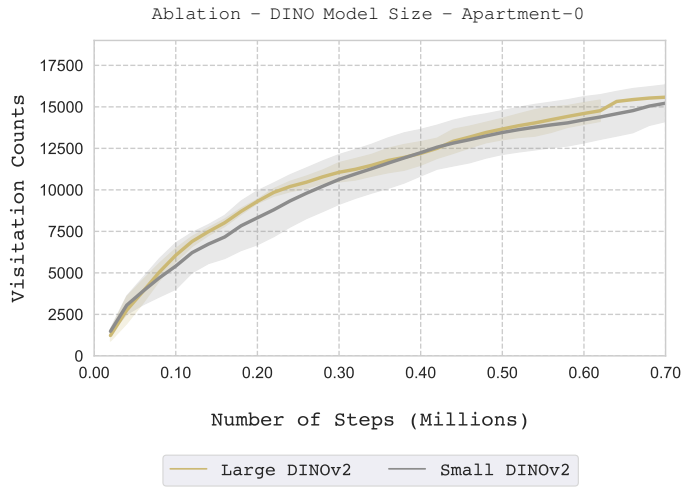



*Figure 10.* **Comparing different model sizes for DINO:** We test whether a larger model size for DINO improves performance for Habitat. Our findings reveal that a larger model size does not improve performance by large margin. We argue that the model size is already large enough to capture the priors in the observations for clustering. This is also indicated by the rather small performance gap between DINOv2 small and DINOv2 large (Caron et al., 2021).

In our experiments we use DINOv2 small (Caron et al., 2021) for the pre-trained representations. We test whether a larger model size for DINO improves performance for Habitat. We plot the visitation counts for the "small" and "large" model sizes in Figure 11. Our findings reveal that a larger model size does not improve performance by large margin. The model

size is likely already large enough to capture the priors in the observations for clustering. This is also indicated by the rather small performance gap between DINOv2 small and DINOv2 large (Caron et al., 2021). However, inference time does increase substantially with larger model sizes, which is why we use DINOv2 small for our experiments.

### C.2. Cluster Counts with and without episodic clustering

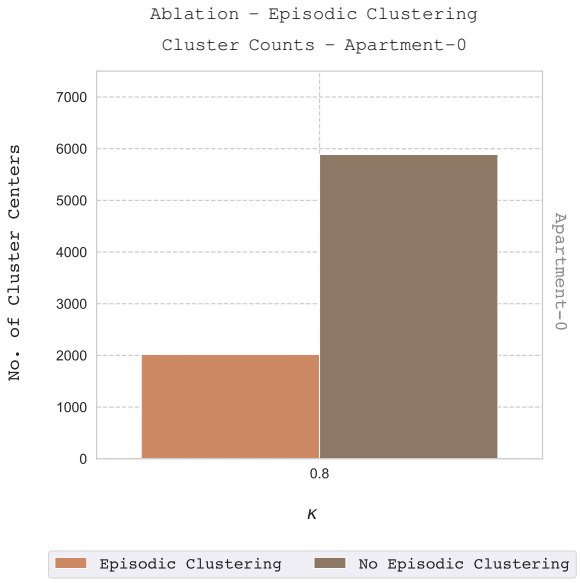



*Figure 11.* **Without episodic clustering the number of cluster centers is higher:** We plot the number of cluster centers in the cluster table with and without episodic clustering. We see that without episodic clustering the number of cluster centers is higher. This is because the episodic clusters are not aggregated as well as with episodic clustering to the cluster table.

We also investigate the effect of episodic clustering on the number of cluster centers in the cluster table. We plot the number of cluster centers in the cluster table with and without episodic clustering in Figure 11. We see that without episodic clustering the number of cluster centers is higher. This is because the episodic clusters are not aggregated as well as with episodic clustering to the cluster table. This means that not performing episodic clustering leads to a more granular cluster table, since the cluster centers in this case only represent a single embedding for every step.

### C.3. Progression of the cluster table throughout learning

To give insight into how exploration progresses with episodic and global clustering we show the progression of the added cluster centers to the cluster table during training in Figure **??**. To locate the cluster centers within the VizDoom environment we additionally track the position of each observation and calculate the mean position of all observations in the cluster as the position for the cluster center. For visualization purposes we show the progression of the cluster table for $\kappa = 0.5$

Already after 50K steps of training the first 4 are represented in the cluster table. At 500K steps almost all rooms are represented in the cluster table. This means that the intrinsic reward will be calculated from these cluster centers as reference points, effectively abstracting the state space. After convergence, the number of cluster centers barely changes since an optimal policy has been found.

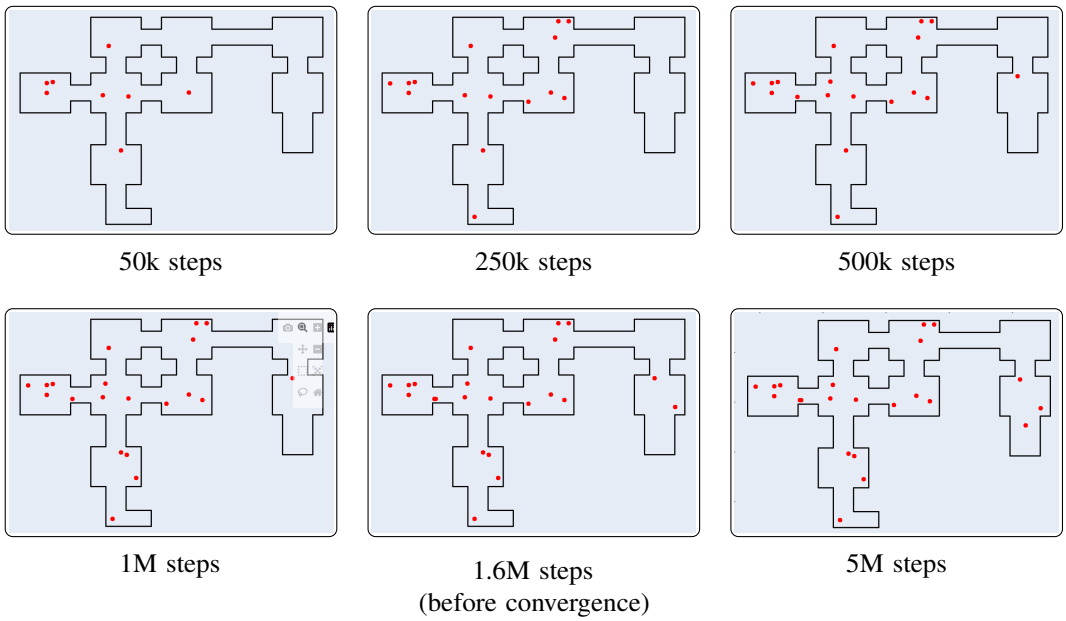



*Figure 12.* **Cluster centers emerge in previously unexplored areas:** We visualize the progression of the cluster table throughout learning for VizDoom. At the beginning of training the cluster table is empty, but as the agent explores the environment, cluster centers emerge in previously unexplored areas. Already at 50k steps the cluster table starts to fill up with cluster centers, mapping out the first 5 rooms of the environment. At convergence for all rooms there is at least a cluster center in the cluster table. After convergence the cluster table is stable since the optimal policy has been found.

## D. Environments

### D.1. VizDoom Environment Details

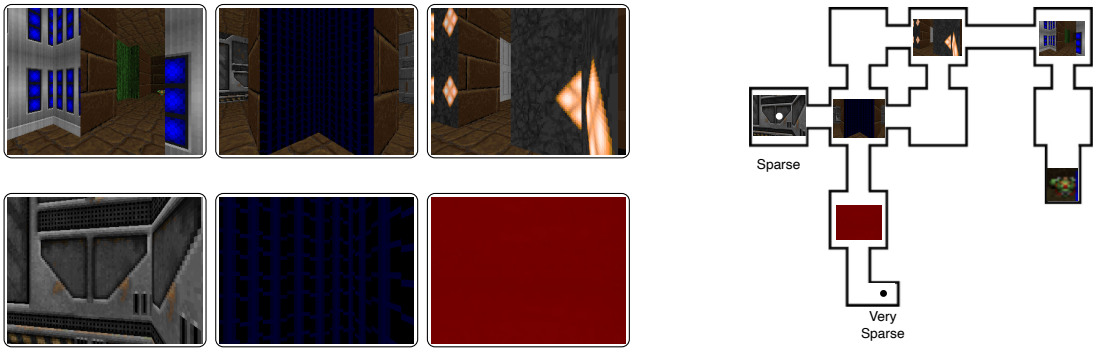



*Figure 13.* **VizDoom:** The VizDoom environment is a 3-D environment based on the 3-D game Doom. The observations are rooms with different textures, where some textures even move. The agent starts in the room on the left and has to reach the armor in the room on the right. There are two configurations for this enviroments: "Sparse" and "VerySparse", which differ in the starting position of the agent.

We use the "MyWayHome" environment from **VizDoom** (Kempka et al., 2016) in both its sparse reward configurations "Sparse" and "VerySparse". In this environment the agent has to reach a piece of armor located in the farthest accesible room of the environment. The configurations differ in the starting position of the agent, one being further away from the goal than the other. In "Sparse", 270 steps away from goal and in "VerySparse", 340 steps away from goal. The agent has an action space of *Wait, Left, Right, Forward* and an episode has a length of 2100 steps. We use a standard frame skip of $4$. The agent receives a reward of $+1$ when it reaches the armor and a $-0.0001$ penalty for every timestep.

#### D.1.1. HYPERPARAMETERS-VIZDOOM

For VizDoom, we show the hyperparameters for PPO in Table 1 and the hyperparameters for the episodic clustering in Table 2. Additionally, we also show the hyperparameters for RECODE in Table 3. We use the same hyperparameters for both "Sparse" and "VerySparse" configurations. Note that in Table 3 we tried different sizes for the RECODE memory as we found this had the most predictable effect on performance, however for VerySparse even with different memory sizes we were not able to get RECODE to work.

| **Hyperparameter** | **Value** |
|---|---|
| Learning rate | 0.0001 |
| Batch size | 256 |
| Epochs | 4 |
| Gamma | 0.99 |
| Clip epsilon | 0.1 |
| Entropy coefficient | 0.005 |
| Value function coefficient | 0.5 |
| Workers | 32 |
| Recurrence | 64 |
| Intrinsic Reward Scaling | 0.1 |

*Table 1.* PPO Hyperparameters for VizDoom

| **Parameter** | **Value** |
|---|---|
| Cosine Similarity Threshold $\kappa$ (DINO) | 0.8 |
| Cosine Similarity Threshold $\kappa$ (Random) | 0.8 |
| Number of Clusters $M$ | 250 |

*Table 2.* Cosine Similarity and Episodic Clustering Parameters

| **Parameter** | **Value** |
|---|---|
| RECODE memory size | $\{100,$ **700**$, 1 \times 10^4\}$ |
| RECODE discount $\gamma$ | **0.999** |
| RECODE insertion probability $\eta$ | **0.05** |
| RECODE relative tolerance $\kappa$ | **0.2** |
| RECODE reward constant $c$ | **0.01** |
| RECODE decay rate $\tau$ | **0.9999** |
| RECODE neighbors $k$ | **20** |

*Table 3.* RECODE Parameters VizDoom - Bold value is used for experiments

### D.2. Habitat Environment Details

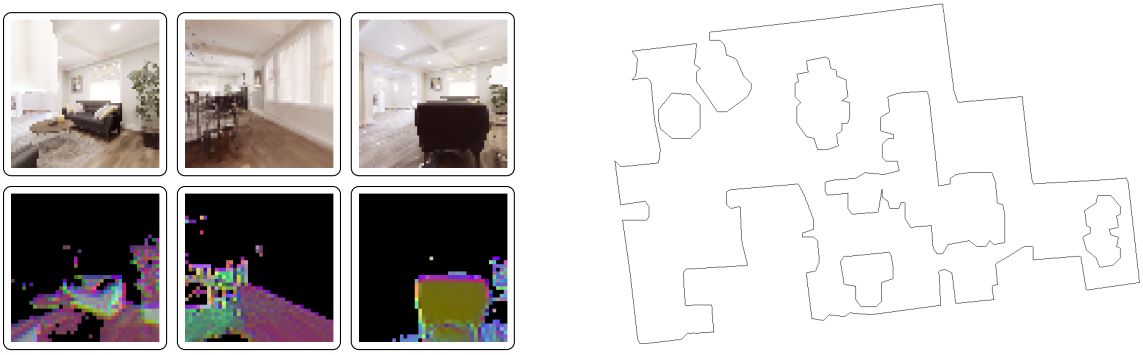


*Figure 14.* **Habitat (Apartment-0):** The Habitat environment is a real world 3-D environment with complex observations. The observations are real world scenes of apartments, hotels and office rooms. The observations are very complex and thus we expect the pre-trained representations to be more effective than random features.

**Habitat-Replica** (Savva et al., 2019; Szot et al., 2021; Puig et al., 2023) is a framework for exploration in real world 3-D scenes. The Habitat-Lab API is made to be used with different datasets. We use the Replica dataset (Straub et al., 2019) as in Parisi et al. (2021), where we train on the "Apartment-0" environment, which is the biggest room in the Replica dataset. The agent has an action space of *Forward* by 0.25 coordinate points, *Left 10°* and *Right 10°* and the episode has a length of 500 steps. The agent is rewarded with intrinsic reward only, where the count of visited locations is the performance metric.

#### D.2.1. Calculating agent position for visitation counts

We calculate the position of the agent for the visitation counts as follows:

$$\text{position} = \text{round}\left(\frac{\text{round}(\text{position}, 2) \times 20}{20}\right).$$

This is directly taken from (Parisi et al., 2021)'s experiments. The position is a tuppe $(x, y, z)$. We round the position to two decimal places and then multiply it by 20. This is done to get a more granular position, since the agent moves by 0.25 coordinate points. Finally we round the position to the nearest integer, which is the position we use for the visitation counts. The visitations counts are then simply counted by aggregating them in a Python dictionary.

#### D.2.2. Hyperparameters

We show the hyperparameters for PPO in Table 4 and the hyperparameters for the episodic clustering in Table 5. Again, we also show the hyperparameters for RECODE in Table 6. For our method with random features we had to increase $\kappa = 0.9$ (compared to $\kappa = 0.8$ in VizDoom) for adequate performance. For the DINO features $\kappa = 0.8$ works well for both enviroments.

| Hyperparameter | Value |
|---|---|
| Learning rate | 0.0001 |
| Batch size | 256 |
| Epochs | 4 |
| Gamma | 0.99 |
| Clip epsilon | 0.1 |
| Entropy coefficient | 0.005 |
| Value function coefficient | 0.5 |
| Workers | 1 |
| Recurrence | 64 |
| Intrinsic Reward Scaling | 0.1 |

*Table 4.* PPO Hyperparameters for Habitat

| Parameter | Value |
|---|---|
| Cosine Similarity Threshold $\kappa$ (DINO) | 0.8 |
| Cosine Similarity Threshold $\kappa$ (Random) | 0.9 |
| Number of Clusters $M$ | 30 |

*Table 5.* Cosine Similarity and Episodic Clustering Parameters

| Parameter | Value |
|---|---|
| RECODE memory size | $\{100,$ **700**, $1 \times 10^4,$ $5 \times 10^4\}$ |
| RECODE discount $\gamma$ | **0.999** |
| RECODE insertion probability $\eta$ | **0.05** |
| RECODE relative tolerance $\kappa$ | **0.2** |
| RECODE reward constant $c$ | **0.01** |
| RECODE decay rate $\tau$ | **0.9999** |
| RECODE neighbors $k$ | **20** |

*Table 6.* RECODE Parameters Habitat - Bold value is used for the experiments.

## E. Implementation Details

### E.1. Observation encoding and network architecture.

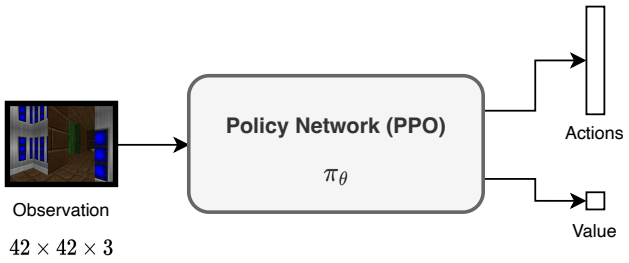



*Figure 15.* **High level overview of policy network:** The policy network receives the observations and outputs the logits for the action probabilities and the value function.

As policy network we take the Proximal Policy Optimization (PPO) (Schulman et al., 2017) implementation from lcswillems and modify it to our needs. As shown in Figure 15, we downscale the observations to $42 \times 42 \times 3$ and feed the downscaled observations to the following network architecture,

```
[...
ACModel(
  (image_conv): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (1): ELU(alpha=1.0)
    (2): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (3): ELU(alpha=1.0)
    (4): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (5): ELU(alpha=1.0)
    (6): Conv2d(32, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1, 1))
    (7): ELU(alpha=1.0)
  )
  (memory_rnn1): LSTMCell(288, 256)
  (memory_rnn2): LSTMCell(288, 256)
  (actor): Sequential(
    (0): Linear(in_features=256, out_features=64, bias=True)
    (1): Tanh()
    (2): Linear(in_features=64, out_features=4, bias=True)
  )
```

```
    (critic): Sequential(
      (0): Linear(in_features=256, out_features=64, bias=True)
      (1): Tanh()
      (2): Linear(in_features=64, out_features=1, bias=True)
    )
    )
```

*Listing 2.* PPO Network Architecture

where the "out_features" for the actor and critic are the number of actions and 1, respectively. Since we have partial observability in our 3-D environments, we additionally use two LSTM cells after the convolutional encoder.

### E.2. DINO Setup

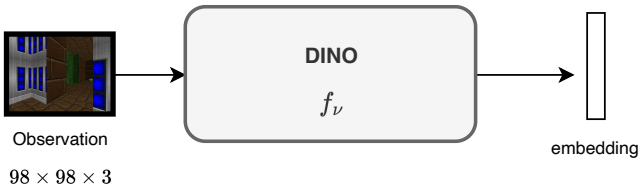



*Figure 16.* **High level overview of how we use DINO:** We use the pre-trained DINO model to extract embeddings from the observations. We then use the cosine similarity to cluster the embeddings.

For the pre-trained DINO representations we use the "small" version of the "dino-v2" (Oquab et al., 2023) model. As shown in Figure 16, the model receives an input of size $98 \times 98 \times 3$ since it splits the observation into $14 \times 14$ patches. After passing the observations through the model, we take the CLS token of the DINO model as feature vector which is $384$-dimensional. For the model size ablation, we use the "large" model instead which has a $1024$-dimensional feature vector with $196 \times 196 \times 3$ sized observations as input.

### E.3. Random Features Setup

For the random features we use the target network in Random Network Distillation which is analogous to Listing 2 without the LSTM cells. After the convolutional encoder we simply output a $384$-dimensional feature vector. As a consquenece, the embeddings, i.e., the cluster means are also $384$-dimensional.

### E.4. Other Libraries

For the Gaussian mixture model we simply use the implementation from scikit-learn.