Contrastive Abstraction for Reinforcement Learning

Vihang Patil[*]   **Markus Hofmarcher**   **Elisabeth Rumetshofer**   **Sepp Hochreiter**

LIT AI Lab, Ellis Unit Linz and Institute for Machine Learning
Johannes Kepler University Linz
`{patil, hofmarch, rumetshofer, hochreit}@ml.jku.at`

Abstract

Learning agents with reinforcement learning is difficult when dealing with long trajectories that involve a large number of states. To address these learning problems effectively, the number of states can be reduced by abstract representations that cluster states. In principle, deep reinforcement learning can find abstract states, but end-to-end learning is unstable. We propose *contrastive abstraction learning* to find abstract states, where we assume that successive states in a trajectory belong to the same abstract state. Such abstract states may be basic locations, achieved subgoals, inventory, or health conditions. *Contrastive abstraction learning* first constructs clusters of state representations by contrastive learning and then applies modern Hopfield networks to determine the abstract states. The first phase of *contrastive abstraction learning* is self-supervised learning, where contrastive learning forces states with sequential proximity to have similar representations. The second phase uses modern Hopfield networks to map similar state representations to the same fixed point, i.e. to an abstract state. The level of abstraction can be adjusted by determining the number of fixed points of the modern Hopfield network. Furthermore, *contrastive abstraction learning* does not require rewards and facilitates efficient reinforcement learning for a wide range of downstream tasks. Our experiments demonstrate the effectiveness of *contrastive abstraction learning* for reinforcement learning.

1 Introduction

Key in reinforcement learning (RL) is to learn proper representation of the environment. If the state space is small and trajectories are short, RL can efficiently construct plans and solve tasks. In particular, for Markov decision processes (MDPs) [72] with few states and short trajectories, an agent can readily learn world models, value functions, and policies. Therefore, the main goal in representation learning for RL is to find clusters of similar states, that form abstract states. If we know how to transit from one abstract state to another one, then state abstraction transforms a complex problem into a simpler problem, e.g. an MDP with many states into an MDP with few states. Deep reinforcement learning gave us the hope that it automatically identifies abstract states, however, end-to-end learning of representations is not stable [54, 50]. Therefore, recent work revisited auxiliary losses and data augmentation to obtain good representations [25]. Also, learning proper state abstractions is computationally expensive, since a proper clustering of states must be identified in large set of possible clusterings. However, if state abstractions are learned reward free they can be used for many RL tasks, thus amortizing the cost of finding these state clusterings.

A promising direction to learn proper RL representations without using rewards is self-supervised learning, where a learning system is trained to capture the mutual dependencies between its inputs [55]. The data stream and not the model designer should determine good representations, which will
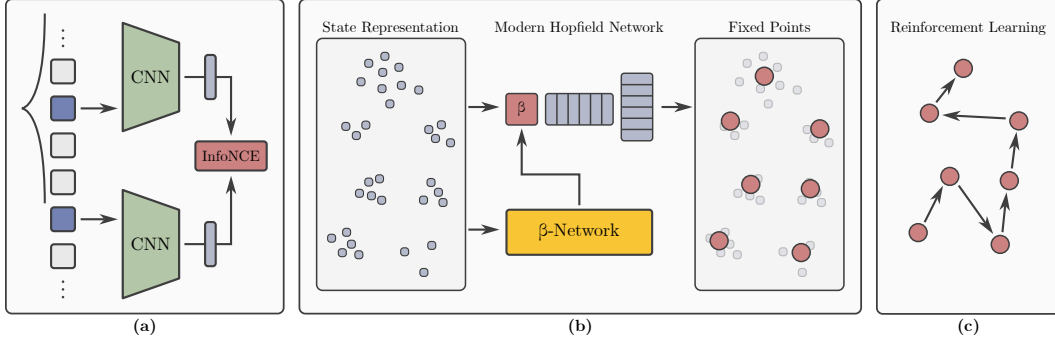
---

[*]Corresponding Author: patil@ml.jku.at

Figure 1: *Contrastive abstraction learning.* **(a)** The first phase applies self-supervised learning via contrastive learning to represent states with sequential proximity in a similar way. Using the InfoNCE objective, two sequentially close states are forced to have a similar representation and non-close states dis-similar representation. **(b)** In the second phase, a modern Hopfield network maps similar representations to the same fixed point which constitutes an abstract state. **(c)** In the last phase, downstream tasks are solved in the reduced state space.

emerge under appropriate training schemes [95]. We leverage such reward free learning to obtain abstract states to make learning efficient in downstream tasks.

Intuition for our approach. Typically, an agent remains in the same abstract state after a state transition, while leaving the abstract state is a rare event. Such abstract states may be basic locations, achieved subgoals, current demands, inventory, or health conditions. If the agent is in the office and wants to buy from the supermarket, then abstract states of the location might be office building, street, and supermarket. The first subgoal would be to leave the office building to the street, the second subgoal to walk on the street to the supermarket, and the final subgoal to buy from the supermarket. These abstract states may be identified by both sequential proximity of original states in the trajectories and similarities of state representations. Therefore, original states should be clustered to create abstract states. In the next step, abstract actions have to be constructed that allow to transit from one abstract state to another. These abstract actions can be subagents, like a subagent for leaving the building to the street. Given abstract states, subagents can be readily trained via reinforcement learning. A start-goal-subagent is placed in a state that belongs to the start abstract state and receives a reward if it reaches a state that belongs to the goal abstract state.

We propose to learn abstractions, i.e. proper state representations, through self-supervised learning. States with sequential proximity in a trajectory are assumed to belong to the same abstract state. Therefore, in the first phase we use contrastive learning to map states with sequential proximity to a similar representation. In the second phase, similar representations are mapped by a modern Hopfield network (MHN) [78] to the same fixed point. Such a fixed point constitutes an abstract state. MHNs allow abstraction at different levels since a temperature parameter determines the number of fixed points. Therefore, the same MHN with the same stored representations enables finer or coarser abstractions. Further, we learn the temperature parameter of the MHN. This is useful as some tasks need to distinguish between states while in some cases we can unify them. Our main contributions are:

- We propose a novel method called "*contrastive abstraction learning*" to learn abstractions in state space without rewards using contrastive learning in the first phase and MHNs in the second phase. Contrastive learning captures the structure of the environment by constructing state clusters, while MHNs supply the abstract states given state clusters.

- *Contrastive abstraction learning* controls the level of abstraction by a temperature parameter of the MHN. The control function adjusts the temperature based on the current state and is trained by contrastive learning.

- We study the resulting abstractions in different environments of varying complexity and evaluate the performance of our *contrastive abstraction learning* in downstream tasks.

2

2 Method

We propose the novel *contrastive abstraction learning* to find abstract states for RL. After learning, original states are mapped to the same abstract state if they are in sequential proximity in the training trajectories. For example, in a grid world, all states within a room would be mapped to the same abstract state. Therefore the number of states is reduced to the number of rooms. Next, we have to construct a small number of sub-policies to move from one room to another room. Importantly, finding such an abstract state does not depend on a goal or reward and, therefore, can be used for various downstream RL tasks. In the grid world example, representing the environment by the rooms helps to solve any task where the agent has to visit specific rooms.

Figure 1 shows an overview of our *contrastive abstraction learning*, which consists of three phases. In the first phase, we sample two states from a sequence of states based on their sequential proximity and use contrastive learning to learn a representation that maps sequentially proximal states to a similar representation. The training trajectories might be expert examples, recorded examples, or examples sampled from a random policy. In the second phase, MHNs reduce the number of states to a small number of abstract states by mapping states to fixed points. In order to control the level of abstraction, the temperature parameters of MHNs can be learned to be controlled. In the final phase, a policy on this small set of abstract states is learned very efficiently since there are only few abstract states. To learn the policy, we have to learn sub-policies that move the agent from one abstract state to another. These sub-policies serve as actions for an MDP built with the abstract states.

2.1 Contrastive Learning of Sequential Proximal States

With the advent of large corpora of unlabeled data in vision and language, contrastive learning methods have become highly successful to learn expressive representations that can be adapted to various downstream tasks [43, 17, 73, 30]. Similarly, contrastive learning can be applied to data of trajectories to construct rich state representations.We utilize contrastive learning to map states with sequential proximity to a similar representation. Given an MDP, the resulting representation encodes the transition structure of the MDP. For example, if an agent transits from state s_1 to s_2 after selecting an action a, then the representation of s_1 should be closer to s_2 than other states s_t.

We define our problem setting as a finite MDP without reward to be a 3-tuple of $(\mathcal{S}, \mathcal{A}, p)$ of finite set $\mathcal{S}$ with states s, $\mathcal{A}$ with actions a, transitions dynamics $p(s_{t+1} = s' \mid s_t = s, a_t = a)$. Here, t is the position in a state-action trajectory $(s_1, a_1, \ldots, s_T, a_T)$ of length T. The agent selects actions $a \sim \pi(s_t = s)$ based on the policy π, which depends on the current state s. For contrastive learning, pairs of states (s_t, s_{t+k}) from the same trajectory with small absolute value of k are selected as positive pairs with high probability, while other pairs are negative pairs, which also include states from different trajectories. We sample a set of positive pairs for each mini-batch of trajectories in the following way. First, we uniformly sample a time index t_i from a trajectory $\tau_{0:T}$. Then, we sample the time index t_j from the same trajectory e.g. by centering a Gauss or Laplace distribution on t_i. The Laplace distribution can be approximated by using a discount factor and normalizing the discounts, where states farther away from i are more discounted and, therefore, have lower probability. Hence, the samples of a positive pair are temporally related. Negative pairs are samples from different trajectories or those with a large sequential distance. With N as the number of samples in the mini-batch, our contrastive objective [93] is

$$\mathrm{L_{InfoNCE}} = -\frac{1}{N} \sum_{i=1}^{N} \ln \frac{\exp(\tau^{-1} s_{t_i}^T s_{t_j})}{\sum_{k=1}^{N} \exp(\tau^{-1} s_{t_i}^T s_{t_k})} \ . \tag{1}$$

2.2 Sampling a Positive Pair

The sampling strategy to construct a positive pair for the contrastive objective has a profound impact on the learned representation. Given a trajectory, the timestep t_i of the first state of a positive pair is sampled uniformly. Then, we center a distribution at the timestep t_i and then sample the timestep t_j of the second state to complete the positive pair. We explore the following four distributions located at t_i for sampling t_j: (I) uniform, (II) Gaussian, (III) Laplace, and (IV) exponential. With sampling from a uniform distribution, the second timestep t_j is sampled uniformly from the same trajectory, thus not using sequential proximity. For sampling from a Gaussian distribution, we center a Gaussian with standard deviation σ at t_i and select t_j that is closest to a sample that can be from both the future

and the past. Analog for sampling from a Laplace distribution with scale parameter b. Also analog for sampling from an exponential distribution inverse scale γ. Since only positive values are sampled, only future states are sampled. For additional details on sampling strategies see Appendix Sec. (B).

2.3 Abstractions via Modern Hopfield Networks

After learning state representations based on sequential proximity, we reduce their number by mapping them to abstract states. With a small state space RL can efficiently solve an MDP for a given reward function. MHNs [78] map states to fixed points that correspond to abstract states. Hopfield networks, introduced in the 1980s, are binary associative memories that played a significant role in popularizing artificial neural networks [3, 46, 47]. These networks were designed to store and retrieve samples [16, 71, 7, 33, 1, 48, 12, 52]. In contrast to traditional binary memory networks, we employ continuous associative memory networks with exceptionally high storage capacity. These MHNs, utilized in deep learning architectures, have an energy function with continuous states and possess the ability to retrieve samples with just a single update [78]. The update rule of MHNs is guaranteed to converge to a fixed point. These fixed points correspond to attractors in the network's energy landscape, and the network dynamics converge towards these attractors when starting at an input pattern. MHNs have already been demonstrated to be successful in diverse fields [96, 84, 97, 67, 30]. In summary, MHNs store and retrieve patterns similar to the query (the input).

We leverage the mechanism of MHNs to map every input pattern to a fixed point. The MHNs have N state representations $\boldsymbol{u}_i$ as stored patterns $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N)$. The update rule of MHNs with inverse temperature β is

$$\boldsymbol{\xi}^{t+1} = \boldsymbol{U} \operatorname{softmax}(\beta \boldsymbol{U}^T \boldsymbol{\xi}^t) , \tag{2}$$

which is applied until convergence. The query $\boldsymbol{x}$ of the MHN is a state representation and serves as initial state $\boldsymbol{\xi}^0 = \boldsymbol{x}$. To determine all fixed points of the MHN, we query with the ith stored representations $\boldsymbol{\xi}^0 = \boldsymbol{u}_i$. Therefore, we obtain N fixed points of the MHNs. After removing duplicates, we get all M different fixed points $(\boldsymbol{y}_1, \ldots, \boldsymbol{y}_M)$ of the MHN that stores $\boldsymbol{U}$.

By adjusting the inverse temperature β, we can control the number of fixed points $\boldsymbol{y}$, and thus the level of abstraction. For $\beta = 0$ there is only one fixed point $\boldsymbol{y}$, which is the average of all stored patterns. Conversely, for large β the stored pattern is retrieved that is closest to the query. For intermediate β values, we obtain meta-stable states, that means several stored patterns converge to a fixed point they share. The larger β the smaller the meta-stable states, i.e. less stored patterns are mapped to the same fixed point. Thus, small values for β result in a small number of fixed points while high values of β in a larger number of fixed points. Therefore, adjusting β to control the number of fixed points, and thus the granularity of our abstract representation, is crucial.

2.4 Controlling the Level of Abstraction

For a given state s, we first obtain its state representation $\boldsymbol{x}$, which serves as input to the MHN. Then we obtain the fixed point $\boldsymbol{y} = \boldsymbol{y}(\boldsymbol{x})$, which is the abstract state. Controlling the level of abstraction is essential to generate representations that are useful for a large number of tasks. MHNs are well suited to achieve this control via the inverse temperature β.

Training a β-network. To determine a suitable β for a given state representation $\boldsymbol{x}$ we train a separate β-network (see overview Figure (1)). This β-network is trained using a contrastive setup. One branch of the contrastive model applies random dropout to a given state representation $\boldsymbol{x}$ to obtain a masked representation $\boldsymbol{x}'$. By masking features in the input, the network is forced to adjust the level of abstraction in such a way, that even masked representations are assigned to the correct fixed point. A second branch consists of the β-network followed by a MHN. The original $\boldsymbol{x}$ is input to the β-network, a small FNN, which regresses a β value. This β value is then used in the MHN to map the state representation $\boldsymbol{x}$ to a fixed point $\boldsymbol{y}$. The output of the two branches, the masked representation $\boldsymbol{x}'$ and the fixed point $\boldsymbol{y}$, form a positive pair. Other masked representations from the mini-batch serve as negatives. The contrastive model is trained with the InfoNCE objective. See Appendix Sec. (C).
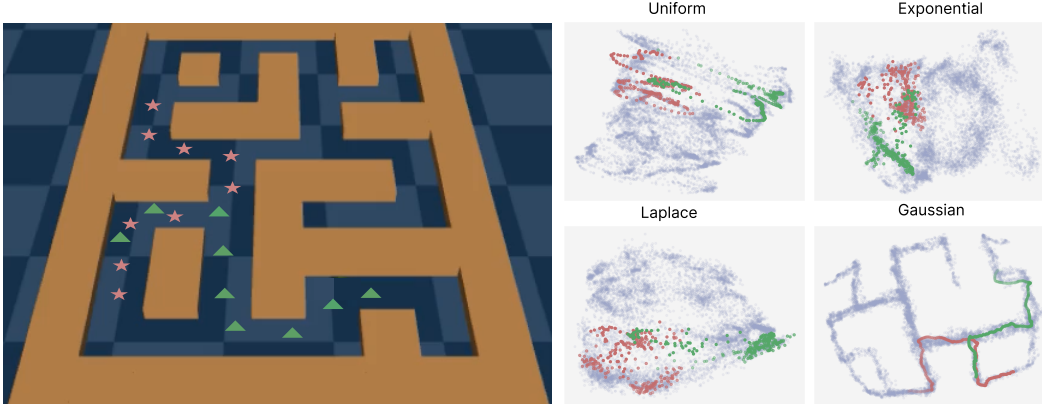
4

Figure 2: Visualization of learned representation of the states for different sampling techniques for contrastive learning. **Left:** Two trajectories (⋆ and ▲) in the Maze2D environment. **Right:** Different sampling techniques of positive pairs for contrastive learning. Red points are states in the dataset, while blue and green are the trajectories from left. Sampling from the Laplace distribution leads to representation well suited for abstraction, while Gaussian does not yield clear clusters.

2.5 Using Abstraction for Downstream Tasks

Abstraction of states can be a powerful tool for improving the efficiency and effectiveness of planning and RL tasks. Abstract states can be used in several ways for downstream applications. The following three approaches demonstrate ways in which abstraction can be applied to downstream tasks. Each offers unique advantages in terms of reduced state space and efficient learning. The specific approach can be chosen depending on the characteristics of the task and the available resources.

Learning a policy from abstract states (Abstract Policy). By reducing the state space to a set of abstract states, we also reduce the complexity of the problem. Instead of operating in the original state space, the policy can now focus on making decisions based on abstract states. We refer this approach as *Abstract Policy*. For this approach, we assume that the optimal policy selects the same action for all original states mapping to one abstract state. While this is a strong assumption we show experimentally, that this approach is viable.

Learning sub-policies as meta-actions (Meta Policy). Sub-policies allow the agent to transition from one abstract state to another, thereby acting as meta-actions. Such sub-policies can be learned from existing trajectories or newly generated trajectories using imitation learning or other RL algorithms. For example, the agent can learn to imitate trajectories between abstract states via Behavior Cloning (BC). The sub-policy for the transition from abstract state y_1 to y_2 starts in a state mapped to y_1 and receives a reward if it arrives at a state which maps to y_2. Subsequently, a meta-policy is trained to select the appropriate sub-policy based on the current abstract state. The advantage of this approach is that it focuses on learning and selecting from the small abstract state space, leading to faster and more targeted learning. We refer to this approach as *Meta Policy*.

Goal-conditioned planning from a graph over abstract states (Planning). In a third approach, a graph is constructed over the abstract states, representing their relationships and connectivity. This graph can then be utilized for planning and decision-making. Given a goal state, the agent can plan a sequence of abstract states that lead to the desired outcome using graph search algorithms. Additionally, learned sub-policies can be employed to execute actions within the environment, facilitating the agent's progress towards the goal state. This approach combines the benefits of abstraction, graph-based planning, and learned sub-policies to enable effective goal-directed behavior.

3 Experiments

In order to show the effectiveness of *contrastive abstraction learning* we perform a series of experiments. First, we verify that abstract representations encode the structure of the environment. Using
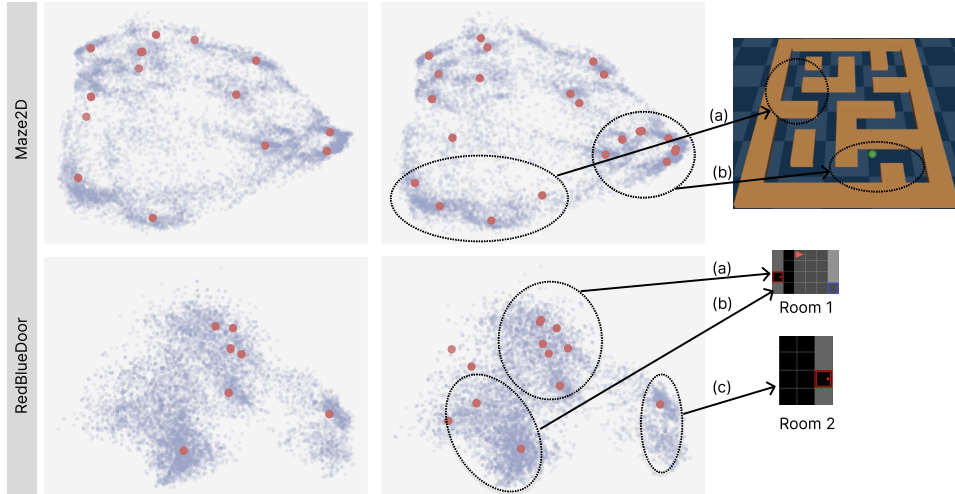
Figure 3: Learned representations (blue) and fixed points (red) of a MHN for Maze2D and Red-BlueDoor. The parameter β of the MHN increases from left to right, thus increasing the number of fixed points. The learned representation forms clusters of states where corridors (top) or rooms (bottom) are close. These clusters are mapped to abstract states via an MHN. In the rooms example (bottom row), (a) corresponds to states where the agent is in Room 1 and the red door is closed, (b) corresponds to the red door being open, (c) corresponds to agent being in Room 2.

trajectories collected from a diverse set of environments we visualize the abstract representations via fixed points. Next, we investigate the connection between β and the level of abstraction. Finally, we show experiments for each of the three approaches of using *contrastive abstraction learning* for downstream tasks detailed in the previous section.

3.1 Environments

We perform all experiments on a set of environments with increasing complexity selected or designed to showcase the effectiveness of a learned abstract representation. Furthermore, for learning a contrastive representation as well as for learning the abstract representation we require a dataset of trajectories. Therefore, we select a set of trajectories for each environment from existing datasets if available or by sampling from a random policy in case no dataset is available. Alternatively, a dataset can easily be acquired with an iterative approach by interacting with the environment but we found that the representation learned from randomly sampled trajectories is often sufficient.

(I) *CifarEnv* is an environment with images and classes from the CIFAR-100 dataset. It is used to verify if the method obtains abstract representations. We select 10 classes with defined transition structures for various goals and tasks. The agent remains in the same state for a specified number of timesteps (e.g., 8), receiving images from the corresponding class. After the timesteps, the agent selects an action to transition to the next state. We sample 100,000 timesteps from this environment to create a dataset for contrastive abstraction learning. **(II)** *Maze2D-large* is an environment from Mujoco Suite [91]. A robot must navigate to a given goal state from a random starting position. We use trajectories from the D4RL dataset [29] for our experiments. **(III)** *RedBlueDoor* is part of the Minigrid library [19] and is a partially observable environment. We generate a dataset by storing all interactions during training of a policy with Proximal Policy Optimization (PPO) [81]. We use the environment and the dataset of human demonstrations released for the MineRL competition [36].

3.2 Contrastive and Abstract Representation

In order to show that *contrastive abstraction learning* results in a representation that encodes the structure of the environment we present qualitative analysis for the various environments. We visualize the learned representation after contrastive learning and the abstract representation obtained

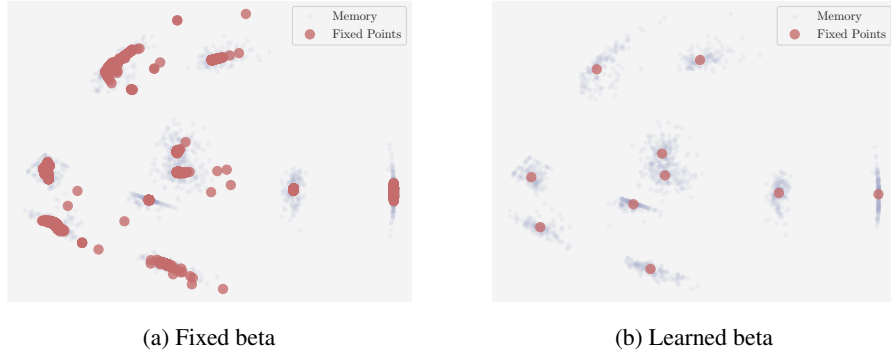(a) Fixed beta   (b) Learned beta

Figure 4: Visualization of all states (blue) in the memory, with the resulting fixed points (red) in *CifarEnv*. We compare the number of fixed points when the temperature parameter is fixed (a) and when the temperature parameter is learned (b). The learned temperature parameter reduces the state space to a number of fixed points equal to the number of Cifar classes in the environment.

via fixed points from MHNs. We use PCA to downproject both the representation after contrastive pre-training as well as the abstract representation.

Sampling strategies for contrastive pre-training. We visualize the learned representation after contrastive learning of sequentially proximal states for different sampling strategies (2.2) in Figure 2. We visualize all samples after contrastive pre-training and highlight two trajectories. While on first look the representation obtained when sampling positive pairs using the Gaussian strategy seems good, for subsequent abstraction this strategy is not well suited. The representation almost perfectly mirrors the original state space, therefore similar states are not closer to each other than without contrastive pre-training. As expected, the Uniform strategy results in unclear clusters and thus a sub-par representation. From the remaining two we prefer sampling with the Laplace strategy as clusters of states emerge and states where both trajectories overlap are closer in the representation.

Controlling the level of abstraction through β. Next, we visualize abstract states obtained via fixed points of the MHN. First, we show the effect of β on the number of abstract states. In Figure 3 we show three different settings of beta for three environments, namely *Maze2D-large* and *RedBlueDoor*. From left to right we increase β and therefore the number of abstract states that emerge. This shows that finding a suitable value for β is crucial for solving downstream tasks.

Learning β. Next, we show on *CifarEnv* that learning β works well and results in a set of abstract states that is both small and covers all modes of the state space. Figure 4 (a) shows a manually selected value for β that at first glance results in a good abstract representation. However, the set of abstract states is still much larger than necessary. In Figure 4 (b) the abstract states, represented by fixed points of the MHN, are much sparser while still covering the state space. Manually tuning β for each environment would be time-consuming, therefore learning an good value for β is crucial.

3.3 Abstract Policy

The first approach for using the abstract representation for downstream tasks we investigate is training an *Abstract Policy*. We assume that the optimal policy selects the same action for every state belonging to an abstract state. As this assumption holds in the *CifarEnv* we train a policy using PPO. At every timestep t the original state s_t is first mapped to it's corresponding abstract state $\boldsymbol{y}$. This abstract state is then used by the policy network. The original action space is not changed. Figure 5 shows that *Abstract Policy* can solve all tasks in this environment, only outperformed by the next *Meta Policy*. For comparison we include two additional baselines. *Contrastive Representation* is trained with PPO using the representation after contrastive pre-training, thus without abstract states. *Original State* is a PPO baseline on the original state space without constrastive abstraction learning.
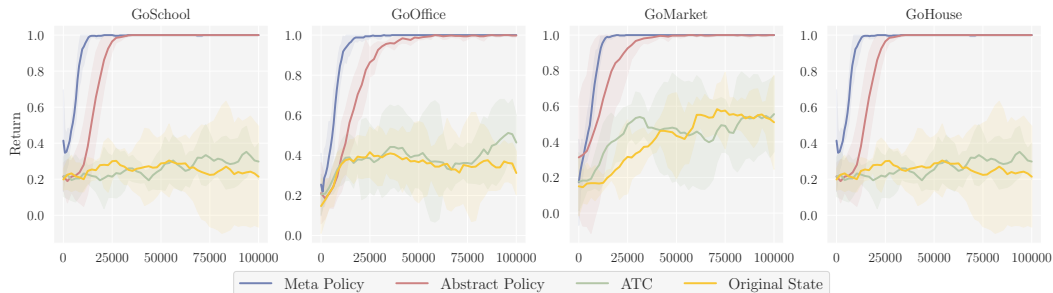
7

Figure 5: We show training of policies from abstract states to solve four tasks in *CifarEnv*. *Abstract Policy* uses abstract states but the original action space. *Meta Policy* selects actions from an abstract action space and executes sub-policies. *Contrastive Representation* uses state representations from contrastive pre-training. For *Original State* both action and state space are not changed. Both methods using abstract states are able to solve all tasks.

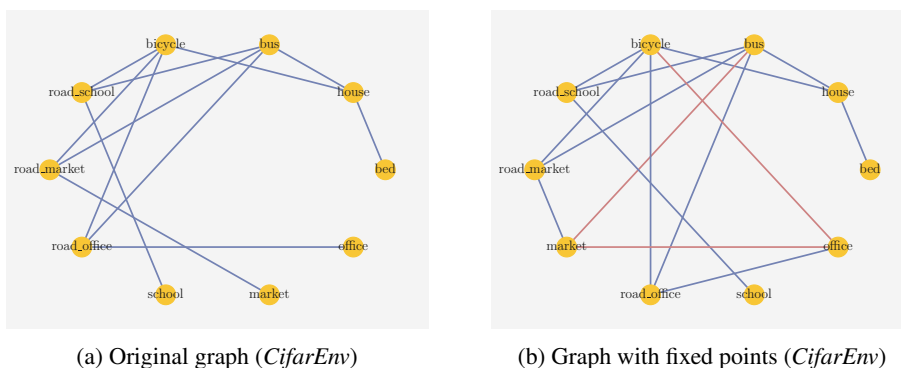(a) Original graph (*CifarEnv*)　　　　(b) Graph with fixed points (*CifarEnv*)

Figure 6: We compare the actual underlying transition graph (a) between states and the graph we construct using fixed points (b). Such a graph can be used for high-level planning in the environment.

3.4　Meta Policy

Next, we train the *Meta Policy*. In contrast to the *Abstract Policy*, the original action space is replaced by an abstract action space where each action represents a transition from one abstract state to another. These abstract actions are learned, either by interacting with the environment or, as in our case, via BC. Thus, each abstract action is a sub-policy that imitates behavior from the dataset. When an abstract action is selected, the corresponding sub-policy is executed in the environment. The *Meta Policy* is again trained with PPO. In Figure 5 we show results using this approach in the *CifarEnv*. *Meta Policy* outperforms all other approaches and baselines.

3.5　Planning

The third approach for obtaining a policy from abstract states is *Planning*. Here, we extract a graph from the abstract states describing an environment and then use it for planning. The graph is constructed by calculating the cosine similarity of all abstract states with each other. Then, a threshold is applied to remove connections between dissimilar abstract states. In Figure 6 we compare the true graph and the graph constructed from the abstract states. While there is a some additional edges, the constructed graph is sufficient to solve almost all tasks. In this experiment, we randomly sample goal states and plan a path towards it using the graph. We find the shortest path from the current abstract state to the abstract state of the goal on the graph. Similar to *Meta Policy*, we use sub-policies to transition from one abstract state to another. These are again learned with BC from the dataset.

In addition to *CifarEnv* we evaluate the robustness of constructing a valid graph from abstract states. For each environment in Table 1 we sample 1000 start and goal states. Using a graph, we find a plan to transition from the start to goal state and execute this plan in the environment. We then record if the goal state is reached within a certain number of timesteps. For more details see Appendix Sec. (D).

Table 1: Planning over graphs. For each environment 1000 abstract start and goal states are sampled. A graph is constructed and used for planning a policy to transition from start to goal state. We show the rate of success of reaching the goal state when the plan is executed in the environment.

Environment	CifarEnv	Maze2D-Large	Maze2D-Medium
Success (%)	86	88	92

4 Related Work

Recently, contrastive learning was very successful at constructing powerful representations with clusters. In the embedding space, contrastive learning brings representations of paired inputs closer to one another while moving unpaired inputs farther away from each other [17, 73, 80]. Following this success, Contrastive Learning has been used in RL to learn representations [87, 25]. CURL [87] learns representations on images by pairing a state with its augmented version. CoBerl [8] masks elements of trajectories in the input and then tries to predict them, analogously to BERT in the language domain. [25] pairs state-actions with future state-actions in a trajectory. Contrastive learning has been used as unsupervised pretraining for RL. APS [40] and ATS [41] learn representations by maximizing entropy during an unsupervised phase and fine-tuning for a subsequent task. ATC [89] pairs state with a state that is k steps ahead in the trajectory. SPR [82, 83], KSL [62] and [4] predict the representation of k future steps and bring it closer to the actual representation. Proto-RL [100] tries similarly to use ideas from Swav [14] to first explore the environment in pre-training to learn representation from diverse samples and then fine-tunes on downstream task.

Abstraction in RL is a well-studied area [10, 34, 6, 79, 90, 57, 21, 5, 94, 53, 70]. Bisimulation [34, 27, 15, 101] tries to simplify MDPs by identifying states which lead to similar behaviors (bisimilar states). Methods based on bisimulation learn abstractions that are dependent on a particular reward function. *Contrastive abstraction learning* enables the identification of sub-goals and sub-tasks, thus making it relevant to hierarchical reinforcement learning (HRL) approaches such as the option framework [90], the MAXQ framework [24], and the recursive composition of option models [86]. Several approaches have been proposed to tackle the problem of learning good options. One approach is selecting frequently observed solution states as targets [88]. Another strategy involves employing gradient-based techniques to enhance the termination function for options [20, 61, 56]. [56] employed policy gradient optimization to learn a unified policy comprising intra-option policies, option termination conditions, and an option selection policy.

Parametrized options can be learned by treating the termination functions as hidden variables and applying expectation maximization [21]. Intrinsic rewards have also been utilized to learn policies within options, while extrinsic rewards are used to learn the policy over options [53]. [5] proposed a method that jointly learns options and their associated policies using the policy gradient theorem. Additionally, a manager module operating on a slow time scale has been introduced to learn sub-goals, which are subsequently achieved by a worker module operating on a fast time scale [94].

5 Conclusion

We propose *contrastive abstraction learning*, a novel approach for state abstraction in RL. Abstract states drastically reduce the number of states for planning in RL. We use contrastive learning to learn a representation where states in sequential proximity are mapped to similar representations. Then we use MHNs to determine abstract states via fixed points. Furthermore, we can adjust the level of abstraction by controlling the number of fixed points. We show how to learn the level of abstraction via contrastive learning. We propose three approaches for using such an abstract representation in RL for efficiently learning policies. Our experiments show, that our approach is able to find a small set of abstract states in several diverse environments and that learning policies based on this representation can be efficient.

Acknowledgments and Disclosure of Funding

References

[1] L. F. Abbott and Y. Arian. Storage capacity of generalized networks. *Physical Review A*, 36: 5091–5094, 1987. doi: 10.1103/PhysRevA.36.5091.

[2] S. Agarwal, G. Krueger, J. Clark, A. Radford, J. W. Kim, and M. Brundage. Evaluating CLIP: Towards characterization of broader capabilities and downstream implications. *ArXiv*, 2108.02818, 2021.

[3] S.-I. Amari. Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on Computers*, C-21(11):1197–1206, 1972. doi: 10.1109/T-C. 1972.223477.

[4] A. Anand, J. Walker, Y. Li, E. Vértes, J. Schrittwieser, S. Ozair, T. Weber, and J. Hamrick. Procedural generalization by planning with self-supervised world models. *arXiv*, 2111.01587, 2021. URL https://arxiv.org/abs/2111.01587.

[5] P. L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, page 1726–1734. AAAI Press, 2017. doi: 10.5555/3298483.3298491.

[6] R. Balaraman. An algebraic approach to abstraction in reinforcement learning. 2003.

[7] P. Baldi and S. S. Venkatesh. Number of stable points for spin-glasses and neural networks of higher orders. *Physical Review Letters*, 58:913–916, 1987. doi: 10.1103/PhysRevLett.58.913.

[8] A. Banino, A. Badia, J. Walker, T. Scholtes, J. Mitrovic, and C. Blundell. Coberl: Contrastive bert for reinforcement learning. *arXiv*, 2107.05431, 2022.

[9] D. Bau, A. Andonian, A. Cui, Y Park, A. Jahanian, A. Oliva, and A. Torralba. Paint by word. *ArXiv*, 2103.10951, 2021.

[10] D.P. Bertsekas and D.A. Castanon. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*, 34(6):589–598, 1989. doi: 10.1109/ 9.24227.

[11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *ArXiv*, 2016.

[12] B. Caputo and H. Niemann. Storage capacity of kernel associative memories. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, page 51–56, Berlin, Heidelberg, 2002. Springer-Verlag.

[13] N. Carlini and A. Terzis. Poisoning and backdooring contrastive learning. *ArXiv*, 2106.09667, 2021.

[14] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924. Curran Associates, Inc., 2020.

[15] P. S. Castro. Scalable methods for computing state similarity in deterministic Markov decision processes. *ArXiv*, 1911.09291, 2019.

[16] H. H. Chen, Y. C. Lee, G. Z. Sun, H. Y. Lee, T. Maxwell, and C. Lee Giles. High order correlation model for associative memory. *AIP Conference Proceedings*, 151(1):86–99, 1986. doi: 10.1063/1.36224.

[17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In H. Daumé and A. Singh, editors, *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1597–1607, 2020.

[18] X. Chen and K. He. Exploring simple siamese representation learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15750–15758, 2021.

[19] M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic gridworld environment for gymnasium, 2018. URL https://github.com/Farama-Foundation/Minigrid.

[20] G. Comanici and D. Precup. Optimal policy switching algorithms for reinforcement learning. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 2, pages 709–714, 2010. doi: 10.1145/1838206.1838300.

[21] C. Daniel, H. vanHoof, J. Peters, and G. Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104, 2016. doi: 10.1007/s10994-016-5580-x.

[22] F. Deng, I.Jang, and S. Ahn. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. *arXiv*, 2021.

[23] B. Devillers, R. Bielawski, B. Choski, and R. VanRullen. Does language help generalization in vision models? *ArXiv*, 2104.08313, 2021.

[24] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):1, 2000.

[25] B. Eysenbach, T. Zhang, R. Salakhutdinov, and S. Levine. Contrastive learning as goal-conditioned reinforcement learning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=vGQiU5sqUe3.

[26] H. Fang, P. Xiong, L. Xu, and Y. Chen. CLIP2Video: Mastering video-text retrieval via image CLIP. *ArXiv*, 2106.11097, 2021.

[27] N. Ferns, P. S. Castro, D. Precup, and P. Panangaden. Methods for computing state similarity in Markov decision processes. *ArXiv*, 1206.6836, 2012.

[28] K. Frans, L. B. Soros, and O. Witkowski. CLIPDraw: Exploring text-to-drawing synthesis through language-image encoders. *ArXiv*, 2106.14843, 2021.

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

[30] A. Fürst, E. Rumetshofer, J. Lehner, V. Tran, F. Tang, H. Ramsauer, D. Kreil, M. Kopp, G. Klambauer, Q. Bitto-Nemling, and S. Hochreiter. CLOOB: modern Hopfield networks with InfoLOOB outperform CLIP. *ArXiv*, 2110.11316, 2021.

[31] F. A. Galatolo, M. G. C. A. Cimino, and G. Vaglini. Generating images from caption and vice versa via CLIP-guided generative latent space search. *ArXiv*, 2102.01645, 2021.

[32] T. Gao, X. Yao, and D. Chen. SimCSE: Simple contrastive learning of sentence embeddings. *ArXiv*, 2104.08821, 2021.

[33] E. Gardner. Multiconnected neural network models. *Journal of Physics A*, 20(11):3453–3464, 1987. doi: 10.1088/0305-4470/20/11/046.

[34] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1):163–223, 2003. doi: 10.1016/S0004-3702(02)00376-4.

[35] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Ávila Pires, Z. D. Guo, M. Gheshlaghi Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 21271–21284, 2020.

[36] W. H. Guss, C. Codel, K. Hofmann, B. Houghton, N. Kuno, S. Milani, S. P. Mohanty, D. P. Liebana, R. Salakhutdinov, N. Topin, M. Veloso, and P. Wang. The MineRL competition on sample efficient reinforcement learning using human priors. *arXiv*, 2019.

[37] W. H. Guss, B. Houghton, N. Topin, P. Wang, C. Codel, M. Veloso, and R. Salakhutdinov. MineRL: A large-scale dataset of Minecraft demonstrations. In *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence (IJCAI'19)*, 2019. URL http://minerl.io.

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

[39] T. Han, W. Xie, and A. Zisserman. Self-supervised co-training for video representation learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5679–5690, 2020.

[40] L. Hao and P. Abbeel. Aps: Active pretraining with successor features. In *In Advances in Neural Information Processing Systems*, page 34:18459–73, 2021.

[41] L. Hao and P. Abbeel. Behavior from the void: Unsupervised active pre-training. In *In Proceedings of the 38th International Conference on Machine Learning, PMLR*, page 6736–47, 2021.

[42] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2.

[43] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[44] O. J. Hénaff, A. Srinivas, J. DeFauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. vanDenOord. Data-efficient image recognition with contrastive predictive coding. *ArXiv*, 1905.09272, 2019.

[45] M. L. Henderson, R. Al-Rfou, B. Strope, Y.-H. Sung, L. Lukács, R. Guo, S. Kumar, B. Miklos, and R. Kurzweil. Efficient natural language response suggestion for smart reply. *ArXiv*, 1705.00652, 2017.

[46] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[47] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81(10):3088–3092, 1984. doi: 10.1073/pnas.81.10.3088.

[48] D. Horn and M. Usher. Capacities of multiconnected memory models. *Journal of Physics France*, 49(3):389–395, 1988. doi: 10.1051/jphys:01988004903038900.

[49] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

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

[51] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Deptartment of Computer Science, University of Toronto, 2009.

[52] D. Krotov and J. J. Hopfield. Dense associative memory for pattern recognition. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1172–1180, 2016.

[53] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and Josh J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 3675–3683. Curran Associates, Inc., 2016.

[54] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *ArXiv*, 2004.14990, 2020.

[55] Y. LeCun. A path towards autonomous machine intelligence. Openreview, 2022. URL https://openreview.net/forum?id=BZ5a1r-kVsf.

[56] K. Y. Levy and N. Shimkin. Unified inter and intra options learning using policy gradient methods. In S. Sanner and M. Hutter, editors, *Recent Advances in Reinforcement Learning*, pages 153–164. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-29946-9_17.

[57] L. Li, T. J. Walsh, and M. L. Littman. Towards a unified theory of state abstraction for MDPs. In *Ninth International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2006.

[58] H. Liu, T. Zahavy, V. Mnih, and S. Singh. Palm up: Playing in the latent manifold for unsupervised pretraining. *arXiv*, abs/2210.10913, 2022.

[59] L. Logeswaran and H. Lee. An efficient framework for learning sentence representations. In *International Conference on Learning Representations (ICLR)*. OpenReview, 2018.

[60] H. Luo, L. Ji, M. Zhong, Y. Chen, W. Lei, N. Duan, and T. Li. CLIP4Clip: An empirical study of CLIP for end to end video clip retrieval. *ArXiv*, 2104.08860, 2021.

[61] D. J. Mankowitz, T. A. Mann, and S. Mannor. Adaptive skills adaptive partitions (ASAP). In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 1588–1596. Curran Associates, Inc., 2016.

[62] T. McInroe, L. Schäfer, and S. Albrecht. Learning temporally-consistent representations for data-efficient reinforcement learning. *arxiv*, abs/2110.04935, 2021.

[63] T. Milbich, K. Roth, S. Sinha, L. Schmidt, M. Ghassemi, and B. Ommer. Characterizing generalization under out-of-distribution shifts in deep metric learning. *ArXiv*, 2107.09562, 2021.

[64] J. Miller, R. Taori, A. Raghunathan, S. Sagawa, P. W. Koh, V. Shankar, P. Liang, Y. Carmon, and L. Schmidt. Accuracy on the line: On the strong correlation between out-of-distribution and in-distribution generalization. *ArXiv*, 2107.04649, 2021.

[65] I. Misra and L. vanDerMaaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[66] M. Narasimhan, A. Rohrbach, and T. Darrell. CLIP-It! Language-guided video summarization. *ArXiv*, 2107.00650, 2021.

[67] F. Paischer, T. Adler, V. Patil, A. Bitto-Nemling, M. Holzleitner, S. Lehner, H. Eghbal-Zadeh, and S. Hochreiter. History compression via language models in reinforcement learning. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, pages 17156–17185, 2022.

[68] D. Pakhomov, S. Hira, N. Wagle, K. E. Green, and N. Navab. Segmentation in style: Unsupervised semantic image segmentation with stylegan and CLIP. *ArXiv*, 2107.12518, 2021.

[69] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[70] Vihang Patil, Markus Hofmarcher, Marius-Constantin Dinu, Matthias Dorfer, Patrick M Blies, Johannes Brandstetter, José Arjona-Medina, and Sepp Hochreiter. Align-RUDDER: Learning from few demonstrations by reward redistribution. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17531–17572. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/patil22a.html.

[71] D. Psaltis and H. P. Cheol. Nonlinear discriminant functions and associative memories. *AIP Conference Proceedings*, 151(1):370–375, 1986. doi: 10.1063/1.36241.

[72] M. L. Puterman. *Markov Decision Processes*. John Wiley & Sons, Inc., 2005. ISBN 978-0-471-72782-8.

[73] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[74] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. *ArXiv*, 2103.00020, 2021.

[75] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[76] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with CLIP latents. *ArXiv*, 2204.06125, 2022. URL https://openai.com/dall-e-2/.

[77] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, V. Greiff, D. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. Hopfield networks is all you need. *ArXiv*, 2008.02217, 2020.

[78] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, V. Greiff, D. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. Hopfield networks is all you need. In *International Conference on Learning Representations (ICLR)*. OpenReview, 2021.

[79] B. Ravindran and A. G. Barto. SMDP homomorphisms: An algebraic approach to abstraction in semi-Markov decision processes. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, pages 1011–1016, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[80] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[81] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *ArXiv*, 2018.

[82] M. Schwarzer, A. Anand, R. Goel, D. Hjelm, A. Courville, and P. Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv*, 2007.05929, 2020. URL https://arxiv.org/abs/2007.05929.

[83] M. Schwarzer, N. Rajkumar, M. Noukhovitch, A. Anand, L. Charlin, D. Hjelm, P. Bachman, and A. Courville. Pretraining representations for data-efficient reinforcement learning. *arXiv*, 2106.04799, 2021. URL https://arxiv.org/abs/2106.04799.

[84] P. Seidl, P. Renz, N. Dyubankova, P. Neves, J. Verhoeven, J. K. Wegner, M. Segler, S. Hochreiter, and G. Klambauer. Improving few-and zero-shot reaction template prediction using modern Hopfield networks. *Journal of Chemical Information and Modeling*, 2022.

[85] S. Shen, L. H. Li, H. Tan, M. Bansal, A. Rohrbach, K.-W. Chang, Z. Yao, and K. Keutzer. How much can CLIP benefit vision-and-language tasks? *ArXiv*, 2107.06383, 2021.

[86] D. Silver and K. Ciosek. Compositional planning using optimal option models. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, volume 2, 2012. arXiv 1206.6473.

[87] A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv*, 2004.04136, 2020.

[88] M. Stolle and D. Precup. Learning options in reinforcement learning. In *Lecture Notes in Computer Science*, volume 2371, pages 212–223, 2002. doi: 10.1007/3-540-45622-8_16.

[89] A. Stooke, L. Kimin, P. Abbeel, and M. Laskin. Decoupling representation learning from reinforcement learning. *arXiv*, 2009.08319, 2021.

[90] R. S. Sutton, D. Precup, and S. P. Singh. Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[91] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[92] Unknown. Stable diffusion repository on github. Github, 2022. URL https://github.com/CompVis/stable-diffusion.

[93] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, 1807.03748, 2018.

[94] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. FeUdal networks for hierarchical reinforcement learning. *arXiv*, abs/1703.01161, 2017.

[95] H. Wang, E. Miahi, M. White, M. C. Machado, Z. Abbasr, R. Kumaraswamy, V. Liu, and A. White. Investigating the properties of neural network representations in reinforcement learning. *ArXiv*, 2203.15955, 2022.

[96] M. Widrich, B. Schäfl, M. Pavlović, H. Ramsauer, L. Gruber, M. Holzleitner, J. Brandstetter, G. K. Sandve, V. Greiff, S. Hochreiter, and G. Klambauer. Modern Hopfield networks and attention for immune repertoire classification. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 18832–18845, 2020.

[97] M. Widrich, M. Hofmarcher, V. P. Patil, A. Bitto-Nemling, and S. Hochreiter. Modern Hopfield networks for return decomposition for delayed rewards. In *Deep RL Workshop at NeurIPS 2021*, 2021. URL https://openreview.net/forum?id=t0PQSDcqAiy.

[98] M. Wortsman, G. Ilharco, M. Li, J. W. Kim, H. Hajishirzi, A. Farhadi, H. Namkoong, and L. Schmidt. Robust fine-tuning of zero-shot models. *ArXiv*, 2109.01903, 2021.

[99] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3733–3742, Los Alamitos, CA, USA, 2018. doi: 10.1109/CVPR.2018.00393.

[100] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Reinforcement learning with prototypical representations. *arXiv*, 2102.11271, 2021. URL https://arxiv.org/abs/2102.11271.

[101] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine. Learning invariant representations for reinforcement learning without reconstruction. *ArXiv*, 2006.10742, 2020.

[102] K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Learning to prompt for vision-language models. *ArXiv*, 2109.01134, 2021.

Contents of the Appendix

A Extended Related Work

A.1 Contrastive Learning

Non-Contrastive learning objectives for self-supervised learning, such as those of BYOL [35] and SimSiam [18] do not require negative samples. However, most self-supervised learning methods use contrastive learning. The most popular constrative objective is InfoNCE [93]. InfoNCE pairs an anchor sample with a positive sample (positive pair) and then contrasts this pair to pairs of the anchor sample with negative samples (negative pairs). InfoNCE has been utilized in transfer learning [44], aiding natural language response suggestions [45], acquiring sentence representations from unlabelled data [59], and facilitating unsupervised feature learning by maximizing distinctions between instances [99]. Furthermore, InfoNCE has been effectively employed for learning visual representations in Pretext-Invariant Representation Learning (PIRL) [65], Momentum Contrast (MoCo) [43], and SimCLR [17]. Given the success of the InfoNCE [93] with sequential data [45, 59], we use it as an objective for contrastive learning. Specifically, we use contrastive learning using the InfoNCE [93] objective.

Using contrastive learning for constructing powerful representations is well established. Contrastive Predictive Coding (CPC) [93] learns abstract representations in an unsupervised way. The representation build by CPC was general enough that it allowed for transfer learning [44]. Contrastive learning has been applied to natural language tasks to learn sentence representations from unlabelled data [45, 59] and to unsupervised feature learning [99]. Also, SimCSE learns sentence embeddings [32] In vision, Pretext-Invariant Representation Learning (PIRL) contrasts representations of transformed versions of the same image with representations of other images [65]. Momentum Contrast (MoCo) was very successful for unsupervised visual representation learning [43]. SimCLR is one of the best-known contrastive learning methods for visual representations, which was highly effective for transfer learning [17]. Vision applications have been extended to video representation learning [39]. A milestone in contrastive learning was Contrastive Language-Image Pre-training (CLIP), which yielded very impressive results at zero-shot transfer learning [73, 74]. CLIP learns expressive image embeddings directly from raw text, thereby leverages a much richer source of supervision than just labels. The representation learned by CLIP are so powerful that a plethora of follow-up publications used CLIP's representations. The CLIP model is used in Vision-and-Language tasks [85]. The CLIP model guided generative models via an additional training objective [9, 31, 28] and improved clustering of latent representations [68]. It is used in studies of out of distribution performance [23, 63, 64], of fine-tuning robustness [98], of zero-shot prompts [102] and of adversarial attacks to uncurated datasets [13]. It stirred discussions about more holistic evaluation schemes in computer vision [2]. Multiple methods utilize the CLIP model in a straightforward way to perform text-to-video retrieval [26, 60, 66]. Latent diffusion models [80] lead to stable diffusion models version 1 [92] and version 2 [92], which are a recent open-source image generation models comparable to proprietary models such as DALL-E 2 [76]. Stable diffusion model version 1 uses CLIP for textual guiding, while version 2 uses OpenCLIP. The representation learned by CLIP is key to gernerate impressive images by stable diffusion models. Stable diffusion models hat high impact on the scientific community as well as in the public.

A.2 Contrastive Learning in Reinforcement Learning

CURL[87] learns representation on raw pixels by bringing closer the representation of a state and its augmented version. CoBerl[8] uses contrastive learning to learn representations by using the masked prediction objective of Bert in the time domain for reinforcement learning. [25] attempt to move the representation of a state-action closer to another state if it is present in the future and away from another state if it is not present in the future. Some methods use contrastive learning during an unsupervised phase before online learning. APS [40] and ATS [41] learn representations by maximizing entropy during an unsupervised phase and fine-tuning for a subsequent task. ATC [89] brings closer the representation of a state to another state which is k steps in the future. SPR [82, 83], KSL [62] and [4] predict the representation of k future steps and bring it closer to the actual representation. DreamerPro [22] combines the ideas from Dreamer [38] and Swav [14] to do Model-Based Reinforcement learning without reconstruction. Proto-RL [100] tries similarly to use ideas from Swav to first explore the environment in pre-training to lean representation from diverse samples and then fine-tunes on downstream task. Palm Up [58] connect large-scale vision models to reinforcement learning by using them as an environment to learn representations.

A.3 Abstraction in Reinforcement Learning

Abstraction in RL is a well-studied area [10, 34, 6, 79, 90, 57, 21, 5, 94, 53]. Bisimulation [34, 27, 15, 101] tries to simplify MDPs by identifying states which lead to similar behaviors (bisimilar states). Methods based on bisimulation learn abstractions that are dependent on a particular reward function. *Contrastive abstraction learning* enables the identification of sub-goals and sub-tasks, thus making it relevant to hierarchical reinforcement learning (HRL) approaches such as the option framework [90], the MAXQ framework [24], and the recursive composition of option models [86]. Several approaches have been proposed to tackle the problem of learning good options. One approach is selecting frequently observed solution states as targets [88]. Another strategy involves employing gradient-based techniques to enhance the termination function for options [20, 61, 56]. [56] employed policy gradient optimization to learn a unified policy comprising intra-option policies, option termination conditions, and an option selection policy. Parametrized options can be learned by treating the termination functions as hidden variables and applying expectation maximization [21]. Intrinsic rewards have also been utilized to learn policies within options, while extrinsic rewards are used to learn the policy over options [53]. [5] proposed a method that jointly learns options and their associated policies using the policy gradient theorem. Additionally, a manager module operating on a slow time scale has been introduced to learn sub-goals, which are subsequently achieved by a worker module operating on a fast time scale [94].

B Contrastive Learning of Sequential Proximity

B.1 Hyperparameters for Constrastive Learning

In the maze experiments, a four-layer fully connected network was employed, comprising of 256, 128, 64, and 32 neurons in each respective layer. The network utilized ReLU activations. The τ value was set to 30 for 1. A batch size of 4096 was used, and the training process consisted of 1000 epochs. The learning rate was set to 1e-3, and a weight decay of 1e-5 was applied.

In the CIFAR environment experiments, a network architecture consisting of three CNN layers followed by three fully connected layers was utilized. The CNN layers had output channels of 64, 32, and 32, with kernel sizes of 4, 4, and 3 respectively. A stride and padding value of 2 was applied throughout. The fully connected layers had sizes of 256, 64, and 8 neurons respectively, employing ReLU activations. The τ value was set to 30 for 1. A batch size of 4096 was used, and the training process spanned 8000 epochs. The learning rate was annealed using cosine annealing, starting from 1e-3 and gradually decreasing to a minimum value of 1e-6. A weight decay of 1e-5 was applied.

In the MiniGrid experiments, a network architecture comprising of three CNN layers followed by two fully connected layers was employed. The CNN layers had output channels of 64 and 32, with kernel sizes of 4, 4, and 2 respectively. Stride and padding values of 2 were uniformly used. The fully connected layers had sizes of 64 and 32 neurons respectively, employing ReLU activations. Following these layers, an LSTM layer and another fully connected layer were added, both with an output size of 32. The τ value was set to 30 for 1. A batch size of 4096 was used, and the training process spanned 8000 epochs. The learning rate was annealed using cosine annealing, starting from 1e-3 and decreasing gradually to a minimum value of 1e-6. Additionally, a weight decay of 1e-5 was applied.

In the Minecraft experiments, the network architecture involves processing a sequence of the 32 most recent frames as input. The first stage of the network consists of four batch-normalized convolution layers with ReLU activation functions. These layers are structured as follows: Conv-Layer-1 with 16 feature maps, a kernel size of 4, a stride of 2, and zero padding of 1; Conv-Layer-2 with 32 feature maps, a kernel size of 4, a stride of 2, and zero padding of 1; Conv-Layer-3 with 64 feature maps, a kernel size of 3, and a stride of 2; and Conv-Layer-4 with 32 feature maps, a kernel size of 3, and a stride of 2. The resulting flattened latent representation (dimension: $R32\times288$) from the convolution stage is then fed into an LSTM layer with 256 units for further processing. The inverse tau value is set to 30 for the contrastive loss. A batch size of 1024 is used, and the training process involves 300 epochs. The learning rate is annealed using cosine annealing, starting from 1e-3 and gradually decreasing to a minimum value of 1e-6. Additionally, a weight decay of 1e-5 is applied.

B.2 Hyperparameters for Sampling Strategies

For the Gaussian sampling strategy, standard deviation (σ) of 15 is used for all experiments. For exponential distribution, we use a γ of 0.99 for all experiments.

C Abstraction using Modern Hopfield Networks

C.1 Review of Modern Hopfield Networks

We briefly review continuous MHNs that are used for deep learning architectures. MHNs [78] map states to fixed points that correspond to abstract states. Hopfield networks, introduced in the 1980s, are binary associative memories that played a significant role in popularizing artificial neural networks [3, 46, 47]. These networks were designed to store and retrieve samples [16, 71, 7, 33, 1, 48, 12, 52]. Since they exhibit continuity and differentiability, these functions are compatible with gradient descent in deep architectures. As they can be updated in a single step, they can be activated similarly to other layers in deep learning. With their exponential storage capacity, they are capable of handling large-scale problems effectively. In contrast to traditional binary memory networks, we employ continuous associative memory networks with exceptionally high storage capacity. These MHNs, utilized in deep learning architectures, have an energy function with continuous states and possess the ability to retrieve samples with just a single update [78]. The update rule of MHNs is guaranteed to converge to a fixed point. These fixed points correspond to attractors in the network's energy landscape, and the network dynamics converge towards these attractors when starting at an input pattern. MHNs have already been demonstrated to be successful in diverse fields [96, 84, 97, 67, 30].

We assume a set of patterns $\{\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N\} \subset \mathbb{R}^d$ that are stacked as columns to the matrix $\boldsymbol{U} = (\boldsymbol{u}_1, \ldots, \boldsymbol{u}_N)$ and a state pattern (query) $\boldsymbol{\xi} \in \mathbb{R}^d$ that represents the current state. The largest norm of a stored pattern is $M = \max_i \|\boldsymbol{u}_i\|$. Continuous MHNs with state $\boldsymbol{\xi}$ have the energy

$$\mathrm{E} \ = \ - \beta^{-1} \, \log \left(\sum_{i=1}^{N} \exp(\beta \boldsymbol{u}_i^T \boldsymbol{\xi}) \right) \ + \ \beta^{-1} \log N \ + \ \frac{1}{2} \, \boldsymbol{\xi}^T \boldsymbol{\xi} \ + \ \frac{1}{2} \, M^2 \, . \tag{A1}$$

For energy E and state $\boldsymbol{\xi}$, the update rule

$$\boldsymbol{\xi}^{\mathrm{new}} \ = \ f(\boldsymbol{\xi}; \boldsymbol{U}, \beta) \ = \ \boldsymbol{U} \, \boldsymbol{p} \ = \ \boldsymbol{U} \, \mathrm{softmax}(\beta \boldsymbol{U}^T \boldsymbol{\xi}) \tag{A2}$$

has been proven to converge globally to stationary points of the energy E, which are almost always local minima [77, 78]. The update rule Eq. (A2) is also the formula of the well-known transformer attention mechanism [77, 78], therefore Hopfield retrieval and transformer attention coincide.

The *separation* Δ_i of a pattern $\boldsymbol{u}_i$ is defined as its minimal dot product difference to any of the other patterns: $\Delta_i = \min_{j, j \neq i} \left(\boldsymbol{u}_i^T \boldsymbol{u}_i - \boldsymbol{u}_i^T \boldsymbol{u}_j \right)$. A pattern is *well-separated* from the data if $\Delta_i \geq \frac{2}{\beta N} + \frac{1}{\beta} \log \left(2(N-1)N\beta M^2 \right)$.

In the case where the patterns $\boldsymbol{u}_i$ exhibit good separation, the iteration defined by Eq. (A2) converges to a stable point that is in close proximity to one of the stored patterns. However, if certain patterns are similar to each other and lack clear separation, the update rule Eq. (A2) converges to a stable point that is close to the mean of those similar patterns. This particular fixed point represents a metastable state of the energy function, effectively averaging over the similar patterns.

According to the next theorem, when the patterns are well separated, the update rule defined by Eq. (A2) typically achieves convergence within a single update. Additionally, the theorem asserts that the retrieval error is exponentially small with respect to the separation Δ_i between the patterns.

Theorem A1 (Modern Hopfield Networks: Retrieval with One Update [78]). *With query $\boldsymbol{\xi}$, after one update the distance of the new point $f(\boldsymbol{\xi})$ to the fixed point $\boldsymbol{u}_i^*$ is exponentially small in the separation Δ_i. The precise bounds using the Jacobian $\mathrm{J} = \frac{\partial f(\boldsymbol{\xi})}{\partial \boldsymbol{\xi}}$ and its value J^m in the mean value theorem are:*

$$\|f(\boldsymbol{\xi}) \ - \ \boldsymbol{u}_i^*\| \ \leqslant \ \|\mathrm{J}^m\|_2 \, \|\boldsymbol{\xi} \ - \ \boldsymbol{u}_i^*\| \, , \tag{A3}$$

$$\|\mathrm{J}^m\|_2 \ \leqslant \ 2 \, \beta \, N \, M^2 \, (N-1) \exp(- \, \beta \, (\Delta_i \ - \ 2 \, \max\{\|\boldsymbol{\xi} \ - \ \boldsymbol{u}_i\|, \|\boldsymbol{u}_i^* \ - \ \boldsymbol{u}_i\|\} \, M)) \, . \tag{A4}$$

For given ϵ and sufficient large Δ_i, we have $\|f(\boldsymbol{\xi}) \ - \ \boldsymbol{u}_i^*\| < \epsilon$, that is, retrieval with one update. The retrieval error $\|f(\boldsymbol{\xi}) \ - \ \boldsymbol{u}_i\|$ of pattern $\boldsymbol{u}_i$ is bounded by

$$\|f(\boldsymbol{\xi}) \ - \ \boldsymbol{u}_i\| \ \leqslant \ 2 \, (N-1) \exp(- \, \beta \, (\Delta_i \ - \ 2 \, \max\{\|\boldsymbol{\xi} \ - \ \boldsymbol{u}_i\|, \|\boldsymbol{u}_i^* \ - \ \boldsymbol{u}_i\|\} \, M)) \, M \, . \tag{A5}$$

For a proof see [77, 78].

Our objective is to store a potentially extensive collection of embeddings. To accomplish this, we initially establish the concept of storing and retrieving patterns within a contemporary Hopfield network.

Definition A1 (Pattern Stored and Retrieved [78])**.** *We assume that around every pattern $\boldsymbol{u}_i$ a sphere S_i is given. We say $\boldsymbol{u}_i$ is stored if there is a single fixed point $\boldsymbol{u}_i^* \in \mathrm{S}_i$ to which all points $\boldsymbol{\xi} \in \mathrm{S}_i$ converge, and $\mathrm{S}_i \cap \mathrm{S}_j = \emptyset$ for $i \neq j$. We say $\boldsymbol{u}_i$ is retrieved for a given ϵ if iteration (update rule) Eq. (A2) gives a point $\tilde{\boldsymbol{x}}_i$ that is at least ϵ-close to the single fixed point $\boldsymbol{u}_i^* \in \mathrm{S}_i$. The retrieval error is $\|\tilde{\boldsymbol{x}}_i - \boldsymbol{u}_i\|$.*

Similar to classical Hopfield networks, we focus on patterns that lie on a sphere, meaning patterns with a constant norm. In the case of randomly selected patterns, the number of patterns that can be stored increases exponentially with the dimensionality d of the pattern space ($\boldsymbol{u}_i \in \mathbb{R}^d$).

Theorem A2 (Modern Hopfield Networks: Exponential Storage Capacity [78])**.** *We assume a failure probability $0 < p \leqslant 1$ and randomly chosen patterns on the sphere with radius $M := K\sqrt{d-1}$. We define $a := \frac{2}{d-1}(1 + \ln(2\beta K^2 p(d-1)))$, $b := \frac{2K^2\beta}{5}$, and $c := \frac{b}{W_0(\exp(a+\ln(b)))}$, where W_0 is the upper branch of the Lambert W function 4.13, and ensure $c \geq \left(\frac{2}{\sqrt{p}}\right)^{\frac{4}{d-1}}$. Then with probability $1 - p$, the number of random patterns that can be stored is*

$$N \geq \sqrt{p}\, c^{\frac{d-1}{4}} . \tag{A6}$$

Therefore it is proven for $c \geq 3.1546$ with $\beta = 1$, $K = 3$, $d = 20$ and $p = 0.001$ ($a + \ln(b) > 1.27$) and proven for $c \geq 1.3718$ with $\beta = 1$, $K = 1$, $d = 75$, and $p = 0.001$ ($a + \ln(b) < -0.94$).

For a proof see [77, 78].

The validity of employing continuous modern Hopfield networks in substituting retrieved embeddings instead of the original embeddings for large batch sizes is supported by this theorem. Even with a substantial number of embeddings, reaching into the hundreds of thousands, the continuous modern Hopfield network demonstrates its capability to retrieve the embeddings, provided that the dimension of the embeddings is sufficiently large.

C.2   Hyperparameters for Hopfield Network

For the fixed β visualisation experiments Fig 3 on Maze2D environment, we use β values of 20, 25 and 35.

For the fixed β visualisation experiments, Fig 3 on RedBlueDoor environment, we use β values of 100, 150 and 200 (left to right in Fig 3.

For the fixed β visualisation experiments, Fig 3 on Minecraft environment, we use β values of 30, 35 and 50.

We currently do not have any learnable weights in the hopfield, but a hopfield network can be trained with learnable weights also.

C.3   Learned Beta Training

For all our experiments we use a single fully connected network as the β network. This β network has a sigmoid output neuron, which predicts a value between 0 and 1. Further, we multiply this output with β_{max}, which is the maximum value the β can have. Thus, bounding the β value. For all our experiments we use the β_{max} value of 200.

We use a single fully connected network after hopfield network. In the experiment, the learning rate was set to 1e-3. A weight decay of 1e-5 was applied to control the regularization of the model's weights. The training process spanned a total of 1000 epochs, with each epoch representing a complete iteration through the dataset. Furthermore, a masking ratio of 0.3 was employed, indicating the proportion of the input or data that was randomly masked or hidden during training.

D Policy and Planning Experiments

D.1 Environment Details

CifarEnv is an environment consisting of images and classes from the CIFAR-100 [51] dataset. This environment is designed to verify the abstract representation as the true number of abstract states and their relation is known. We select 10 classes and define a transition structure for various goals and tasks (see Figure 6(a)). For example, in *GoSchool* the agent starts in state *bed* and receives reward if it reaches the state *school*. Other tasks include, *GoMarket, GoOffice, GoHouse*. The agent stays in the same state for n timesteps (where $n = 8$ for our experiments) in this environment, and is presented with an image sampled from the respective class in each timestep. After n timesteps the agent then selects an action in order to transition to the next state. For example, if the agent is in the state *house* then n different images of this class are sampled from CIFAR-100 sequentially. We randomly sample $100,000$ timesteps from this environment and build a dataset for constrastive abstraction learning.

Maze2D-large is an environment from Mujoco Suite [91]. A robot must navigate to a given goal state from a random starting position. The state is an 4-dimensional vector and the action space is continuous, controlling the agents directional movement. We use trajectories from the D4RL dataset [29] for our experiments.

RedBlueDoor is part of the Minigrid library [19] and is a partially observable environment. The agent has access to a partial view of the environment and can select from 7 discrete actions. We generate a dataset by storing all interactions during training of a policy with Proximal Policy Optimization (PPO) [81].

Minecraft is an environment based on the popular sandbox game Minecraft. The environment was released together with a dataset of demonstrations from human players performing various tasks for the MineRL competition [36]. Several tasks, such as mining a diamond, were defined for the competition. We use the environment and the dataset of human demonstrations released for the MineRL competition [36].

D.2 Policy experiments

We use the same PPO hyperparameters across all the methods. The clip coefficient was set to 0.2, which limits the size of the policy update to prevent drastic changes. The entropy coefficient was set to 0.0001, encouraging exploration by adding an entropy term to the objective function. The learning rate was set to 1e-5, indicating the step size used in the optimization process. There was no annealing applied to the learning rate or clipping parameter throughout the training process. Additionally, to mitigate the impact of large gradients, the gradient was clipped if its norm exceeded 0.5. The experiments were averaged over 10 different seeds.

D.3 Learning Policy between fixed points

We learn policy between fixed points using behavior cloning. We first map each state to the corresponding fixed point. Further, we pair the state with the next fixed point in the sequence, thus obtaining a dataset where each sample has the following information, $(s_t, \boldsymbol{\xi}_t, \boldsymbol{\xi}_{t'}, a_t)$. Further, we condition the policy network with the current state $_t$ and future fixed point $\boldsymbol{\xi}_{t'}$ and predict the action a_t in the dataset.

These policy between fixed points are used for both MetaPolicy and Planning experiments.

D.4 Planning

For the CIFAR experiments, we use a threshold of 0.68 to prune the connections in the graph. For Maze2D experiments, we use a threshold of 0.6. We also remove self-connection by removing most similar connections to the current node using another threshold (0.97, 0.9 respectively). Instead of using a threshold, we can also use the dataset to prune connections. We can look for consecutive fixed points in the entire dataset and remove connections which were not found.

E Compute and Software Libraries

E.1 Compute Details

For all experiments, during development 2 to 3 nodes each with 4 GPUs of an internal GPU cluster were used for roughly six months of GPU compute time (Nvidia A100).

E.2 Software Libraries

We are thankful towards the developers of Stable-Baselines [75], PyTorch [69], OpenAI Gym [11], Numpy [42], Matplotlib [49] and Minecraft [37]. We also used code from [70].

References

[1] L. F. Abbott and Y. Arian. Storage capacity of generalized networks. *Physical Review A*, 36: 5091–5094, 1987. doi: 10.1103/PhysRevA.36.5091.

[2] S. Agarwal, G. Krueger, J. Clark, A. Radford, J. W. Kim, and M. Brundage. Evaluating CLIP: Towards characterization of broader capabilities and downstream implications. *ArXiv*, 2108.02818, 2021.

[3] S.-I. Amari. Learning patterns and pattern sequences by self-organizing nets of threshold elements. *IEEE Transactions on Computers*, C-21(11):1197–1206, 1972. doi: 10.1109/T-C. 1972.223477.

[4] A. Anand, J. Walker, Y. Li, E. Vértes, J. Schrittwieser, S. Ozair, T. Weber, and J. Hamrick. Procedural generalization by planning with self-supervised world models. *arXiv*, 2111.01587, 2021. URL https://arxiv.org/abs/2111.01587.

[5] P. L. Bacon, J. Harb, and D. Precup. The option-critic architecture. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, page 1726–1734. AAAI Press, 2017. doi: 10.5555/3298483.3298491.

[6] R. Balaraman. An algebraic approach to abstraction in reinforcement learning. 2003.

[7] P. Baldi and S. S. Venkatesh. Number of stable points for spin-glasses and neural networks of higher orders. *Physical Review Letters*, 58:913–916, 1987. doi: 10.1103/PhysRevLett.58.913.

[8] A. Banino, A. Badia, J. Walker, T. Scholtes, J. Mitrovic, and C. Blundell. Coberl: Contrastive bert for reinforcement learning. *arXiv*, 2107.05431, 2022.

[9] D. Bau, A. Andonian, A. Cui, Y Park, A. Jahanian, A. Oliva, and A. Torralba. Paint by word. *ArXiv*, 2103.10951, 2021.

[10] D.P. Bertsekas and D.A. Castanon. Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Transactions on Automatic Control*, 34(6):589–598, 1989. doi: 10.1109/ 9.24227.

[11] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba. Openai gym. *ArXiv*, 2016.

[12] B. Caputo and H. Niemann. Storage capacity of kernel associative memories. In *Proceedings of the International Conference on Artificial Neural Networks (ICANN)*, page 51–56, Berlin, Heidelberg, 2002. Springer-Verlag.

[13] N. Carlini and A. Terzis. Poisoning and backdooring contrastive learning. *ArXiv*, 2106.09667, 2021.

[14] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin. Unsupervised learning of visual features by contrasting cluster assignments. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 9912–9924. Curran Associates, Inc., 2020.

[15] P. S. Castro. Scalable methods for computing state similarity in deterministic Markov decision processes. *ArXiv*, 1911.09291, 2019.

[16] H. H. Chen, Y. C. Lee, G. Z. Sun, H. Y. Lee, T. Maxwell, and C. Lee Giles. High order correlation model for associative memory. *AIP Conference Proceedings*, 151(1):86–99, 1986. doi: 10.1063/1.36224.

[17] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton. A simple framework for contrastive learning of visual representations. In H. Daumé and A. Singh, editors, *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1597–1607, 2020.

[18] X. Chen and K. He. Exploring simple siamese representation learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 15750–15758, 2021.

[19] M. Chevalier-Boisvert, L. Willems, and S. Pal. Minimalistic gridworld environment for gymnasium, 2018. URL https://github.com/Farama-Foundation/Minigrid.

[20] G. Comanici and D. Precup. Optimal policy switching algorithms for reinforcement learning. In *Proceedings of the International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS)*, volume 2, pages 709–714, 2010. doi: 10.1145/1838206.1838300.

[21] C. Daniel, H. vanHoof, J. Peters, and G. Neumann. Probabilistic inference for determining options in reinforcement learning. *Machine Learning*, 104, 2016. doi: 10.1007/s10994-016-5580-x.

[22] F. Deng, I.Jang, and S. Ahn. Dreamerpro: Reconstruction-free model-based reinforcement learning with prototypical representations. *arXiv*, 2021.

[23] B. Devillers, R. Bielawski, B. Choski, and R. VanRullen. Does language help generalization in vision models? *ArXiv*, 2104.08313, 2021.

[24] T. G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning*, 40(2):1, 2000.

[25] B. Eysenbach, T. Zhang, R. Salakhutdinov, and S. Levine. Contrastive learning as goal-conditioned reinforcement learning. In A. H. Oh, A. Agarwal, D. Belgrave, and K. Cho, editors, *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=vGQiU5sqUe3.

[26] H. Fang, P. Xiong, L. Xu, and Y. Chen. CLIP2Video: Mastering video-text retrieval via image CLIP. *ArXiv*, 2106.11097, 2021.

[27] N. Ferns, P. S. Castro, D. Precup, and P. Panangaden. Methods for computing state similarity in Markov decision processes. *ArXiv*, 1206.6836, 2012.

[28] K. Frans, L. B. Soros, and O. Witkowski. CLIPDraw: Exploring text-to-drawing synthesis through language-image encoders. *ArXiv*, 2106.14843, 2021.

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

[30] A. Fürst, E. Rumetshofer, J. Lehner, V. Tran, F. Tang, H. Ramsauer, D. Kreil, M. Kopp, G. Klambauer, Q. Bitto-Nemling, and S. Hochreiter. CLOOB: modern Hopfield networks with InfoLOOB outperform CLIP. *ArXiv*, 2110.11316, 2021.

[31] F. A. Galatolo, M. G. C. A. Cimino, and G. Vaglini. Generating images from caption and vice versa via CLIP-guided generative latent space search. *ArXiv*, 2102.01645, 2021.

[32] T. Gao, X. Yao, and D. Chen. SimCSE: Simple contrastive learning of sentence embeddings. *ArXiv*, 2104.08821, 2021.

[33] E. Gardner. Multiconnected neural network models. *Journal of Physics A*, 20(11):3453–3464, 1987. doi: 10.1088/0305-4470/20/11/046.

[34] R. Givan, T. Dean, and M. Greig. Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, 147(1):163–223, 2003. doi: 10.1016/S0004-3702(02)00376-4.

[35] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. H. Richemond, E. Buchatskaya, C. Doersch, B. Ávila Pires, Z. D. Guo, M. Gheshlaghi Azar, B. Piot, K. Kavukcuoglu, R. Munos, and M. Valko. Bootstrap your own latent - a new approach to self-supervised learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 21271–21284, 2020.

[36] W. H. Guss, C. Codel, K. Hofmann, B. Houghton, N. Kuno, S. Milani, S. P. Mohanty, D. P. Liebana, R. Salakhutdinov, N. Topin, M. Veloso, and P. Wang. The MineRL competition on sample efficient reinforcement learning using human priors. *arXiv*, 2019.

[37] W. H. Guss, B. Houghton, N. Topin, P. Wang, C. Codel, M. Veloso, and R. Salakhutdinov. MineRL: A large-scale dataset of Minecraft demonstrations. In *Proc. of the 28th Int. Joint Conf. on Artificial Intelligence (IJCAI'19)*, 2019. URL http://minerl.io.

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

[39] T. Han, W. Xie, and A. Zisserman. Self-supervised co-training for video representation learning. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 5679–5690, 2020.

[40] L. Hao and P. Abbeel. Aps: Active pretraining with successor features. In *In Advances in Neural Information Processing Systems*, page 34:18459–73, 2021.

[41] L. Hao and P. Abbeel. Behavior from the void: Unsupervised active pre-training. In *In Proceedings of the 38th International Conference on Machine Learning, PMLR*, page 6736–47, 2021.

[42] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith, R. Kern, M. Picus, S. Hoyer, M. H. van Kerkwijk, M. Brett, A. Haldane, J. F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T. E. Oliphant. Array programming with NumPy. *Nature*, 585(7825):357–362, September 2020. doi: 10.1038/s41586-020-2649-2.

[43] K. He, H. Fan, Y. Wu, S. Xie, and R. B. Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[44] O. J. Hénaff, A. Srinivas, J. DeFauw, A. Razavi, C. Doersch, S. M. A. Eslami, and A. vanDenOord. Data-efficient image recognition with contrastive predictive coding. *ArXiv*, 1905.09272, 2019.

[45] M. L. Henderson, R. Al-Rfou, B. Strope, Y.-H. Sung, L. Lukács, R. Guo, S. Kumar, B. Miklos, and R. Kurzweil. Efficient natural language response suggestion for smart reply. *ArXiv*, 1705.00652, 2017.

[46] J. J. Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982.

[47] J. J. Hopfield. Neurons with graded response have collective computational properties like those of two-state neurons. *Proceedings of the National Academy of Sciences*, 81(10):3088–3092, 1984. doi: 10.1073/pnas.81.10.3088.

[48] D. Horn and M. Usher. Capacities of multiconnected memory models. *Journal of Physics France*, 49(3):389–395, 1988. doi: 10.1051/jphys:01988004903038900.

[49] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9 (3):90–95, 2007. doi: 10.1109/MCSE.2007.55.

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

[51] A. Krizhevsky. Learning multiple layers of features from tiny images. Master's thesis, Deptartment of Computer Science, University of Toronto, 2009.

[52] D. Krotov and J. J. Hopfield. Dense associative memory for pattern recognition. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 1172–1180, 2016.

[53] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and Josh J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 3675–3683. Curran Associates, Inc., 2016.

[54] M. Laskin, K. Lee, A. Stooke, L. Pinto, P. Abbeel, and A. Srinivas. Reinforcement learning with augmented data. *ArXiv*, 2004.14990, 2020.

[55] Y. LeCun. A path towards autonomous machine intelligence. Openreview, 2022. URL https://openreview.net/forum?id=BZ5a1r-kVsf.

[56] K. Y. Levy and N. Shimkin. Unified inter and intra options learning using policy gradient methods. In S. Sanner and M. Hutter, editors, *Recent Advances in Reinforcement Learning*, pages 153–164. Springer Berlin Heidelberg, 2012. doi: 10.1007/978-3-642-29946-9_17.

[57] L. Li, T. J. Walsh, and M. L. Littman. Towards a unified theory of state abstraction for MDPs. In *Ninth International Symposium on Artificial Intelligence and Mathematics (ISAIM)*, 2006.

[58] H. Liu, T. Zahavy, V. Mnih, and S. Singh. Palm up: Playing in the latent manifold for unsupervised pretraining. *arXiv*, abs/2210.10913, 2022.

[59] L. Logeswaran and H. Lee. An efficient framework for learning sentence representations. In *International Conference on Learning Representations (ICLR)*. OpenReview, 2018.

[60] H. Luo, L. Ji, M. Zhong, Y. Chen, W. Lei, N. Duan, and T. Li. CLIP4Clip: An empirical study of CLIP for end to end video clip retrieval. *ArXiv*, 2104.08860, 2021.

[61] D. J. Mankowitz, T. A. Mann, and S. Mannor. Adaptive skills adaptive partitions (ASAP). In D. Lee, M. Sugiyama, U. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 29, pages 1588–1596. Curran Associates, Inc., 2016.

[62] T. McInroe, L. Schäfer, and S. Albrecht. Learning temporally-consistent representations for data-efficient reinforcement learning. *arxiv*, abs/2110.04935, 2021.

[63] T. Milbich, K. Roth, S. Sinha, L. Schmidt, M. Ghassemi, and B. Ommer. Characterizing generalization under out-of-distribution shifts in deep metric learning. *ArXiv*, 2107.09562, 2021.

[64] J. Miller, R. Taori, A. Raghunathan, S. Sagawa, P. W. Koh, V. Shankar, P. Liang, Y. Carmon, and L. Schmidt. Accuracy on the line: On the strong correlation between out-of-distribution and in-distribution generalization. *ArXiv*, 2107.04649, 2021.

[65] I. Misra and L. vanDerMaaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[66] M. Narasimhan, A. Rohrbach, and T. Darrell. CLIP-It! Language-guided video summarization. *ArXiv*, 2107.00650, 2021.

[67] F. Paischer, T. Adler, V. Patil, A. Bitto-Nemling, M. Holzleitner, S. Lehner, H. Eghbal-Zadeh, and S. Hochreiter. History compression via language models in reinforcement learning. In K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, pages 17156–17185, 2022.

[68] D. Pakhomov, S. Hira, N. Wagle, K. E. Green, and N. Navab. Segmentation in style: Unsupervised semantic image segmentation with stylegan and CLIP. *ArXiv*, 2107.12518, 2021.

[69] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[70] Vihang Patil, Markus Hofmarcher, Marius-Constantin Dinu, Matthias Dorfer, Patrick M Blies, Johannes Brandstetter, José Arjona-Medina, and Sepp Hochreiter. Align-RUDDER: Learning from few demonstrations by reward redistribution. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 17531–17572. PMLR, 17–23 Jul 2022. URL https://proceedings.mlr.press/v162/patil22a.html.

[71] D. Psaltis and H. P. Cheol. Nonlinear discriminant functions and associative memories. *AIP Conference Proceedings*, 151(1):370–375, 1986. doi: 10.1063/1.36241.

[72] M. L. Puterman. *Markov Decision Processes*. John Wiley & Sons, Inc., 2005. ISBN 978-0-471-72782-8.

[73] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. In *Proceedings of the International Conference on Machine Learning (ICML)*, 2021.

[74] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger, and I. Sutskever. Learning transferable visual models from natural language supervision. *ArXiv*, 2103.00020, 2021.

[75] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

[76] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen. Hierarchical text-conditional image generation with CLIP latents. *ArXiv*, 2204.06125, 2022. URL https://openai.com/dall-e-2/.

[77] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, V. Greiff, D. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. Hopfield networks is all you need. *ArXiv*, 2008.02217, 2020.

[78] H. Ramsauer, B. Schäfl, J. Lehner, P. Seidl, M. Widrich, L. Gruber, M. Holzleitner, M. Pavlović, G. K. Sandve, V. Greiff, D. Kreil, M. Kopp, G. Klambauer, J. Brandstetter, and S. Hochreiter. Hopfield networks is all you need. In *International Conference on Learning Representations (ICLR)*. OpenReview, 2021.

[79] B. Ravindran and A. G. Barto. SMDP homomorphisms: An algebraic approach to abstraction in semi-Markov decision processes. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI'03)*, pages 1011–1016, San Francisco, CA, USA, 2003. Morgan Kaufmann Publishers Inc.

[80] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10684–10695, 2022.

[81] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *ArXiv*, 2018.

[82] M. Schwarzer, A. Anand, R. Goel, D. Hjelm, A. Courville, and P. Bachman. Data-efficient reinforcement learning with self-predictive representations. *arXiv*, 2007.05929, 2020. URL https://arxiv.org/abs/2007.05929.

[83] M. Schwarzer, N. Rajkumar, M. Noukhovitch, A. Anand, L. Charlin, D. Hjelm, P. Bachman, and A. Courville. Pretraining representations for data-efficient reinforcement learning. *arXiv*, 2106.04799, 2021. URL https://arxiv.org/abs/2106.04799.

[84] P. Seidl, P. Renz, N. Dyubankova, P. Neves, J. Verhoeven, J. K. Wegner, M. Segler, S. Hochreiter, and G. Klambauer. Improving few-and zero-shot reaction template prediction using modern Hopfield networks. *Journal of Chemical Information and Modeling*, 2022.

[85] S. Shen, L. H. Li, H. Tan, M. Bansal, A. Rohrbach, K.-W. Chang, Z. Yao, and K. Keutzer. How much can CLIP benefit vision-and-language tasks? *ArXiv*, 2107.06383, 2021.

[86] D. Silver and K. Ciosek. Compositional planning using optimal option models. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*, volume 2, 2012. arXiv 1206.6473.

[87] A. Srinivas, M. Laskin, and P. Abbeel. Curl: Contrastive unsupervised representations for reinforcement learning. *arXiv*, 2004.04136, 2020.

[88] M. Stolle and D. Precup. Learning options in reinforcement learning. In *Lecture Notes in Computer Science*, volume 2371, pages 212–223, 2002. doi: 10.1007/3-540-45622-8_16.

[89] A. Stooke, L. Kimin, P. Abbeel, and M. Laskin. Decoupling representation learning from reinforcement learning. *arXiv*, 2009.08319, 2021.

[90] R. S. Sutton, D. Precup, and S. P. Singh. Between MDPs and Semi-MDPs: A framework for temporal abstraction in reinforcement learning. *Artificial Intelligence*, 112(1-2):181–211, 1999.

[91] E. Todorov, T. Erez, and Y. Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 5026–5033. IEEE, 2012.

[92] Unknown. Stable diffusion repository on github. Github, 2022. URL https://github.com/CompVis/stable-diffusion.

[93] A. van den Oord, Y. Li, and O. Vinyals. Representation learning with contrastive predictive coding. *ArXiv*, 1807.03748, 2018.

[94] A. S. Vezhnevets, S. Osindero, T. Schaul, N. Heess, M. Jaderberg, D. Silver, and K. Kavukcuoglu. FeUdal networks for hierarchical reinforcement learning. *arXiv*, abs/1703.01161, 2017.

[95] H. Wang, E. Miahi, M. White, M. C. Machado, Z. Abbasr, R. Kumaraswamy, V. Liu, and A. White. Investigating the properties of neural network representations in reinforcement learning. *ArXiv*, 2203.15955, 2022.

[96] M. Widrich, B. Schäfl, M. Pavlović, H. Ramsauer, L. Gruber, M. Holzleitner, J. Brandstetter, G. K. Sandve, V. Greiff, S. Hochreiter, and G. Klambauer. Modern Hopfield networks and attention for immune repertoire classification. In H. Larochelle, M. Ranzato, R. Hadsell, M. F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems (NeurIPS)*, pages 18832–18845, 2020.

[97] M. Widrich, M. Hofmarcher, V. P. Patil, A. Bitto-Nemling, and S. Hochreiter. Modern Hopfield networks for return decomposition for delayed rewards. In *Deep RL Workshop at NeurIPS 2021*, 2021. URL https://openreview.net/forum?id=t0PQSDcqAiy.

[98] M. Wortsman, G. Ilharco, M. Li, J. W. Kim, H. Hajishirzi, A. Farhadi, H. Namkoong, and L. Schmidt. Robust fine-tuning of zero-shot models. *ArXiv*, 2109.01903, 2021.

[99] Z. Wu, Y. Xiong, S. X. Yu, and D. Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3733–3742, Los Alamitos, CA, USA, 2018. doi: 10.1109/CVPR.2018.00393.

[100] D. Yarats, R. Fergus, A. Lazaric, and L. Pinto. Reinforcement learning with prototypical representations. *arXiv*, 2102.11271, 2021. URL https://arxiv.org/abs/2102.11271.

[101] A. Zhang, R. McAllister, R. Calandra, Y. Gal, and S. Levine. Learning invariant representations for reinforcement learning without reconstruction. *ArXiv*, 2006.10742, 2020.

[102] K. Zhou, J. Yang, C. C. Loy, and Z. Liu. Learning to prompt for vision-language models. *ArXiv*, 2109.01134, 2021.