Investigating the role of model-based learning in exploration and transfer

Jacob Walker [*1] **Eszter Vértes** [*1] **Yazhe Li** [*1] **Gabriel Dulac-Arnold** [2] **Ankesh Anand** [1] **Théophane Weber** [†1] **Jessica B. Hamrick** [†1]

Abstract

State of the art reinforcement learning has enabled training agents on tasks of ever increasing complexity. However, the current paradigm tends to favor training agents from scratch on every new task or on collections of tasks with a view towards generalizing to novel task configurations. The former suffers from poor data efficiency while the latter is difficult when test tasks are out-of-distribution. Agents that can effectively transfer their knowledge about the world pose a potential solution to these issues. In this paper, we investigate transfer learning in the context of model-based agents. Specifically, we aim to understand *when* exactly environment models have an advantage and *why*. We find that a model-based approach outperforms controlled model-free baselines for transfer learning. Through ablations, we show that both the policy and dynamics model learnt through exploration matter for successful transfer. We demonstrate our results across three domains which vary in their requirements for transfer: in-distribution procedural (Crafter), in-distribution identical (RoboDesk), and out-of-distribution (Meta-World). Our results show that intrinsic exploration combined with environment models present a viable direction towards agents that are self-supervised and able to generalize to novel reward functions.

1. Introduction

A fundamental component of intelligence is generalization: the ability to transfer knowledge to novel situations and tasks. Although the field of reinforcement learning (RL) focused for many years on single-task settings in which generalization is not required (Mnih et al., 2015; Schulman et al., 2017; Haarnoja et al., 2018), there has been a recent surge of interest in designing agents which can successfully transfer their knowledge after training in both multi-task (Kirk et al., 2021; Hospedales et al., 2021) and unsupervised (Hansen et al., 2019; Campos et al., 2021) settings. At the same time, another thread of research has focused on developing ever-more powerful model-based agents (Schrittwieser et al., 2020b; Hafner et al., 2023) to excel on a wide range of environments but have not focused on the problem of transfer. However, many expect model-based RL to be an essential ingredient in generalization and transfer (Tolman, 1948; Dayan et al., 1995; Ha & Schmidhuber, 2018; Schmidhuber, 1991a; Sutton, 1991). Indeed, previous work has shown that model-based RL is advantageous for generalization under certain assumptions (Sekar et al., 2020; Anand et al., 2021), lending weight to the hypothesis that model-based reasoning is important for transfer.

Although previous work has demonstrated the efficacy of model-based learning for generalization in some cases, it has not shed much insight as to *when* we ought to expect it to help or *why*. In this paper, we address these questions by studying the performance of agents trained with intrinsic exploration and fine-tuned on downstream tasks (see Figure 1 for an overview of our setup). By utilizing unsupervised pre-training via intrinsic exploration, we control the type of knowledge the agent can potentially transfer: it knows about the world it inhabits, but not about possible tasks that are achievable in this world. To identify how much different components such as the model or policy contribute to transfer performance, we fine-tune with either pre-trained or reinitialized weights; were we in a zero-shot generalization setting, any such ablations would lead to catastrophic (and thus uninformative) losses in performance.

To conduct our experiments, we make a variety of implementation choices regarding model-based and model-free learning and intrinsic exploration; however, we emphasize that our contribution is less about these particular choices and more about the insights that our experiments bring. For the model-based agent, we employ a self-supervised variant of MuZero (Schrittwieser et al., 2020b) trained with self-predictive representations (SPR; Schwarzer et al., 2020). This variant of MuZero has previously been shown to perform well at zero-shot generalization Anand et al. (2021),

[*]Equal contribution [1]DeepMind, London, UK [2]Google Research. Correspondence to: Jacob Walker <jcwalker@deepmind.com>, Eszter Vértes <evertes@deepmind.com>, Yazhe Li <yazhe@deepmind.com>.

Proceedings of the 40^{th} International Conference on Machine Learning, Honolulu, Hawaii, USA. PMLR 202, 2023. Copyright 2023 by the author(s).

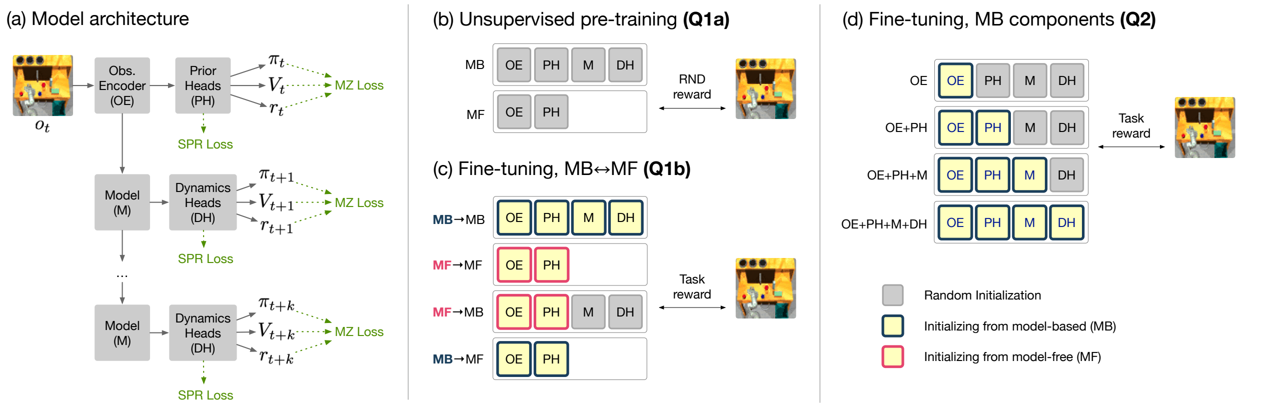


Figure 1. Model-based unsupervised pre-training and fine-tuning. (a) A model-based agent is comprised of an observation encoder (OE), prior heads (PH), model (M), and dynamics heads (DH). These components are trained via a MuZero-style loss (Schrittwieser et al., 2020b) and self-supervision (Schwarzer et al., 2020). (b) During pre-training, the agent interacts with an environment attempting to optimize an intrinsic exploration objective. (c-d) Transfer is evaluated by fine-tuning an agent on a task-based reward in the same environment while carrying over various components from the pre-trained agents. Colored blocks indicate components that are transferred while gray blocks indicate those that are re-initialized. Red indicates transfer from a pre-trained model-free agent, while blue indicates transfer from a pre-trained model-based agent. MB denotes model-based, and MF represents model-free.

thus making it a sensible candidate to test transfer, too. We contrast its performance with a model-free Q-learning agent based on the same architecture. During pre-training, both model-free and model-based agents are trained with random network distillation (RND; Burda et al., 2018b), a straightforward and robust approach to exploration. We evaluate transfer across a number of task suites with different characteristics including procedurally generated environments (Crafter; Hafner, 2021), settings where the environment in the pre-training is the same (RoboDesk; Kannan et al., 2021), and settings where the fine-tuned environment is partially out-of-distribution (Meta-World; Yu et al., 2020).

Overall, we find that model-based exploration combined with model-based fine-tuning results in better transfer performance than model-free baselines. More precisely, we show that: (1) Model-based methods perform better exploration than their model-free counterparts in reward-free environments. (2) Knowledge is transferred most effectively when performing model-based (as opposed to model-free) pre-training and fine-tuning. (3) System dynamics present in the world model seem to improve transfer performance. (4) The model-based advantage is stronger when the dynamics model is trained on the same environment.

2. Background

2.1. Transfer learning in RL

Learning complex tasks from scratch can be prohibitively expensive, if not impossible, and many approaches have been attempted to alleviate this difficulty. One approach is to pre-train an agent on high quality data obtained from humans (Vinyals et al., 2019; Baker et al., 2022); however, this can be costly if data is not readily available. Another approach is to pre-train an agent on a family of (potentially easier) tasks which are related to the task of interest, and then transfer to the target task either directly, within a few attempts, or through fine-tuning (Zhu et al., 2020; Kirk et al., 2021). Generally, the knowledge being transferred can take different forms, such as data (or demonstrations), skills and behaviors, value functions, and dynamics (or world models). Most recently, a new area of focus has been unsupervised reinforcement learning (Watters et al., 2019; Laskin et al., 2021; Campos et al., 2021) in which an agent is pre-trained using without task rewards using intrinsic exploration and then fine-tuned on a task of interest. Our work falls into this category and investigates the differences between model-free and model-based unsupervised RL.

A number of previous works have explored directions related to model-based transfer learning, though most do not fit under the umbrella of unsupervised RL. Some works have investigated training models on source tasks and transferring them to target tasks with model predictive control, but do not consider policy fine-tuning (e.g. Dasari et al., 2019; Bucher et al., 2021; Lutter et al., 2021; Byravan et al., 2021). Others do consider policy fine-tuning, but as with most other approaches, utilize a distribution of source tasks rather than unsupervised exploration Byravan et al. (e.g. 2020). Another approach is to use a pre-trained model as a way to speed up policy transfer by learning a policy from interaction with the model (e.g Nagabandi et al., 2018; Sekar et al., 2020). Of note is work by Sekar et al. (2020), which also incorporates an unsupervised exploration phase for model learning. However, their approach to transfer relies on being able to relabel previously collected experience with the target reward function, whereas we make no such assumption.

Our framework consists of an unsupervised pre-training phase in a no reward environment followed by fine-tuning with task-specific reward functions. This is in contrast to other frameworks such as dataset sharing and policy reuse. Dataset sharing (Lambert et al., 2022) assumes access to the downstream task reward functions which is used for relabelling the offline dataset. Our framework does not need such an assumption which is non-trivial in most applications. In addition, storing the whole exploration experience is more expensive than storing the network weights. Policy reuse (Zhang et al.) assumes that source policies are readily available and a higher level policy is learned to perform on the target domain. In this case, source policies are obtained with extrinsic rather than intrinsic rewards. Unlike fine-tuning, the source policies are kept unchanged in the adaptation. If policy reuse is combined with unsupervised skill discovery, it would make the policy reuse framework similar to our setup.

2.2. Intrinsic exploration

The goal of intrinsic exploration is to learn a policy which explores the environment in a general manner without the need for an explicit task-based reward function. This permits the agent to learn from environments where reward labels are unavailable or sparse. Count-based exploration methods (Bellemare et al., 2016; Ostrovski et al., 2017; Tang et al., 2017) keep track of how often each state has been visited, and reward the agent for visiting novel or infrequently visited states. Another approach based on state frequency is to directly maximize the entropy of the state distribution (Hazan et al., 2019; Yarats et al., 2021). However, most real-world environments are not tabular and it is therefore not straightforward how to keep track of visitation counts. As an alternative, curiosity-driven exploration computes a measure of surprise (such as prediction error or uncertainty) and rewards the agent for visiting surprising states (Schmidhuber, 1991b; Oudeyer et al., 2007; Pathak et al., 2017; Burda et al., 2018b). Ensemble-based methods are a subclass of curiosity-based approaches and estimate uncertainty in a principled way by leveraging an ensemble of predictions (Osband et al., 2016; Lowrey et al., 2018; Pathak et al., 2019; Sekar et al., 2020); however, they are often computationally expensive and challenging to implement. Finally, instead of seeking surprising states (as in curiosity-based approaches), empowerment focuses on learning skills to control the environment and which support exploring the state space more efficiently (Klyubin et al., 2005; Gregor et al., 2016; Eysenbach et al., 2018). In this paper, we leverage Random Network Distillation (Burda et al., 2018b), which is a curiosity-based approach in which prediction error is computed using a fixed randomly initialized neural network as the target. RND is straightforward to implement and has been shown to work robustly across a variety of domains (Burda et al., 2018b; Laskin et al., 2021).

3. Methods

To investigate the efficacy of model-based transfer, we begin by considering a model-based architecture (Figure 1a), described in more detail in Section 3.1. During unsupervised **pre-training** (Figure 1b), we train agents based on this architecture to explore using an intrinsic motivation signal derived from curiosity, as described in Section 3.2. Then, during **fine-tuning** (Figure 1c-d), we transfer various components of these agents and train them on downstream tasks using only the task reward and no intrinsic reward. Using this framework, we consider the following questions:

Q1: *Is there an advantage to an agent being model-based during unsupervised exploration and/or fine-tuning?* We investigate this question by first pre-training both model-free and model-based agents to maximize an intrinsic reward in an unsupervised exploration phase. We then fine-tune these agents using either model-based (MB) or model-free (MF) learning, resulting in four variations: MB$\rightarrow$MB, MB$\rightarrow$MF, MF$\rightarrow$MB, and MF$\rightarrow$MF. We also consider a further baseline, 'Scratch', in which the model-based agent is trained on the test task with randomly initialize weights (i.e. without any pre-training). Additional details are provided in Appendix A. By comparing these different agents, we can determine whether model-based pre-training leads to improved transfer performance.

Q2: *What are the contributions of each component of a model-based agent for downstream task learning?* We investigate this by looking at various ablations of the full model-based agent (MB$\rightarrow$MB), where only certain components are transferred whereas others are reset. In particular, we investigate the effect of the dynamics heads (DH), the dynamics model (M), and the prior heads (PH). We also specifically ablate the prior policy head (PP) apart from the prior reward and value heads (PRV). By performing these ablations over the various combinations of agent components, we can understand the contributions of each model to downstream task performance.

Q3: *How well does the model-based agent deal with environmental shift between the unsupervised and downstream phases?* Finally, by investigating transfer performance in different classes of environments we can consider the effects of environment mismatch on downstream performance. We propose three environments to look at: a procedurally generated environment with the same distribution in both the unsupervised and the downstream phase (Crafter; Hafner, 2021), one with the same MDP (reward function put aside) during both the unsupervised phase and the downstream phase (RoboDesk; Kannan et al., 2021), and one with a stronger shift where system dynamics are maintained but the objects the agent interacts in downstream tasks have various degrees of novelty relative to ones seen during unsupervised exploration (Meta-World; Yu et al., 2020).

3.1. Agents

Model-Based We use MuZero (Schrittwieser et al., 2020b) as the "backbone" of our agent. As MuZero has shown superior generalization capabilities when combined with representation learning techniques, we use a variant of MuZero with self-predictive (SPR) loss as the auxiliary loss (Anand et al., 2021).

MuZero learns a partial world model that predicts rewards, actions, and values. Similar to AlphaZero (Silver et al., 2018), it then incorporates this model in Monte-Carlo Tree Search (Kocsis & Szepesvári, 2006; Coulom, 2006) to plan and choose optimal actions. The reward, value, and policy are learned as the agent collects experience from the world and plans. The loss at time-step t is the following:

$$\begin{aligned} l_t(\theta) &= \sum_{k=0}^{K} l_\pi^k + l_v^k + l_r^k + l_{\text{SPR}}^k \qquad (1)\\ &= \sum_{k=0}^{K} \text{CE}(\hat{\pi}^k, \pi^k) + \text{CE}(\hat{v}^k, v^k) \\ &\qquad + \text{CE}(\hat{r}^k, r^k) + \text{CS}(\hat{y}^k, y^k), \end{aligned}$$

where K represents the number of steps MuZero plans in the future. $\hat{\pi}^k, \hat{v}^k$ and $\hat{r}^k$ are predictions for policy π^k from the search tree, v^k is an n-step return bootstrapped by a target network, and r^k is the true reward from the environment. As in the original MuZero (Schrittwieser et al., 2020a), we define the loss for training the value and reward functions via a cross-entropy (CE) distributional RL loss. The SPR loss is computed as a cosine similarity (CS) between the projections predicted by the dynamics model $\hat{y}_k$ and projections computed using observations at timestep $t+k$, y_k. We also use replay via Reanalyse (Schrittwieser et al., 2021) for better data efficiency. For environments with a continuous action space, we use sampled MuZero (Hubert et al., 2021) which plans over sampled actions. More details of the MuZero agent are provided in Section A.2.

Model-Free As a model-free baseline for pre-training, we use the Q-Learning agent as described in Anand et al. (2021). The Q-Learning baseline is based on the same architecture as MuZero agent, and in particular, utilizes the same architecture for the observation encoder and prior heads. This allows us to transfer the component weights from the model-free baseline to the model-based agent, and vice versa (see Figure 1). To transfer MuZero weights to a Q-Learning agent (MB$\rightarrow$MF), we transfer only the observation encoder and prior heads. To transfer Q-Learning weights to a MuZero agent (MF$\rightarrow$MB), we transfer over the observation encoder and prior heads while initializing the dynamics model and dynamics heads from scratch. More details of the Q-Learning agent are provided in Section A.3.

3.2. Exploration

To drive exploration during the unsupervised pre-training phase, we use intrinsic rewards computed from Random Network Distillation (RND; Burda et al., 2018b). RND can be seen as a method for approximately quantifying uncertainty, and which has proven empirically to be a robust approach to intrinsic exploration (Burda et al., 2018a; Laskin et al., 2021). We found that RND worked well with relatively less tuning with our framework versus approaches such as BYOL-explore (Guo et al., 2022).

RND defines an intrinsic reward by randomly projecting an observation to feature space $z = f_{\text{rand}}(o)$. For a given observation o, the agent attempts to predict the random projection via $\hat{z} = f_\theta(o)$, where θ are learnable parameters. The error between the agent's prediction and the random projection, $e = (z - \hat{z})^2$, provides an intrinsic reward signal. Intuitively, since f_{rand} is not known, the value of f_{rand} in a state can only be known (predictable) to the agent if the agent has visited the state before; novel states in the environment so far unvisited by the agent should have large error, and correspondingly high curiosity signal.

To incorporate RND into MuZero, we replace r^k with $e^k = (z^k - \hat{z}^k)^2$ in Equation 1, where z^k is the random projection of observation o^k, and $\hat{z}^k$ is the output of the observation encoder given observation o^k. Through experimentation we found that a few modifications from the standard RND improved the performance of RND specifically for our architectural setup. We made z convolutional features, and our projector for z shares the same architecture as the observational encoder for simplicity.

We also investigated using the output of the dynamics model to construct the prediction $\hat{z}^k$ which led to inferior exploration performance. Note that in contrast to model-free methods, MuZero optimizes the predicted RND signal as opposed to the measured RND signal.

3.3. Environments

We conduct our transfer experiments in three settings. The first, Crafter (Hafner, 2021)), has the fine-tuning environments in-distribution with the pre-training phase, but episodes are procedurally generated (so that test environments are never encountered during training); the second RoboDesk, (Kannan et al., 2021)), is the same environment during pre-training and fine-tuning phase, the only difference being the availability of tasks rewards during the

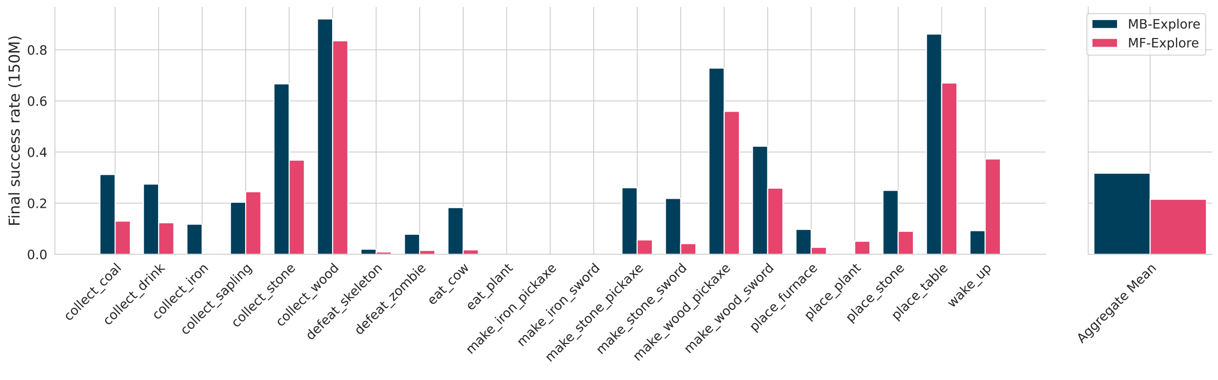
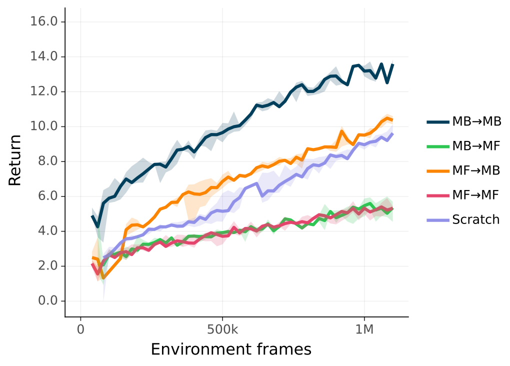


Figure 2. Exploration performance of model-based and model-free agents on Crafter as measured by the per-task and aggregate success rates after pre-training for 150 million environment steps.

(a) Finetuning return curves



(b) Crafter scores and final rewards

Method	Score	Reward
Human Experts (Hafner, 2021)	50.5 ± 6.8	14.3 ± 2.3
MB→MB	$\mathbf{16.4 \pm 1.5}$	$\mathbf{12.7 \pm 0.4}$
MB→MF	8.8 ± 0.4	5.0 ± 0.2
MF→MB	6.2 ± 0.5	9.3 ± 0.3
MF→MF	6.7 ± 0.6	5.0 ± 0.2
DreamerV3 (Hafner et al., 2023)	14.5 ± 1.6	11.7 ± 1.9
LSTM-SPCNN (Stanić et al., 2022)	12.1 ± 0.8	-
DreamerV2 (Hafner, 2021)	10.0 ± 1.2	9.0 ± 1.7
MB Scratch	4.4 ± 0.4	8.5 ± 0.1
MF Scratch	2.6 ± 0.1	4.4 ± 0.1

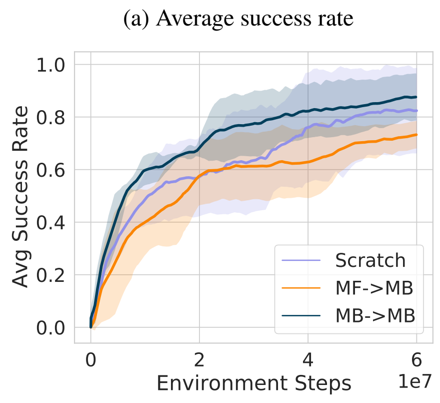
Figure 3. Agent performance on Crafter. (a) Return as a function of the environment steps; (b) Comparison of score and final reward at 1M steps across different fine-tuned agents and published results for agents without fine-tuning. In all our experiments, we used 3 seeds. In line with the literature (Stanić et al., 2022) we report mean and standard deviation across seeds when computing the score.

fine-tuning phase; the third, Meta-World, (Yu et al., 2020)) has fine-tuning environments which share some similarities with the pre-training environment, but are nevertheless different (and therefore out of distribution).

Crafter (Hafner, 2021) is a survival game inspired by the popular game Minecraft. In Crafter, the agent inhabits a two-dimensional procedurally generated world. The environment is multi-task, and these tasks are hierarchical. In order to achieve certain tasks, the agent must complete other tasks first. These include gathering resources, building tools, and defending against potential threats. There are 22 potential achievements. The reward signal comprises $+1$ for each task achieved for the first time in an episode, and, in addition, the agent needs to maintain a given health level to survive by e.g. eating and drinking. As in the original paper, we report the success rate as the fraction of all training episodes up to 1M environment steps where the agent has achieved the task at least once. There are 17 discrete actions. As in Hafner (2021), we also report the score, which is the geometric mean of the success rates: $S = \exp\left(\frac{1}{N}\sum_{i=1}^{N}\log(1+s_i)\right) - 1$. Unlike the return, the score favors unlocking difficult achievements over repeatedly completing easier ones, and being an aggregate measure, it also reflects data-efficiency of the agent.

Although we constrain the agent to use 1M environment steps for fine-tuning, we do not impose such constraint on the data consumption during unsupervised exploration. Our focus is on a relative analysis of model-based transfer and not necessarily to benchmark the efficacy of exploration techniques.

RoboDesk (Kannan et al., 2021) is a control environment simulating a robotic arm interacting with a table and a fixed set of objects. The benchmark features nine core tasks with consistent dynamics but randomized object locations. During exploration, the agent can learn to interact with the objects via optimising intrinsic rewards. Then, we fine-tune the agent individually on the nine different task-rewards.



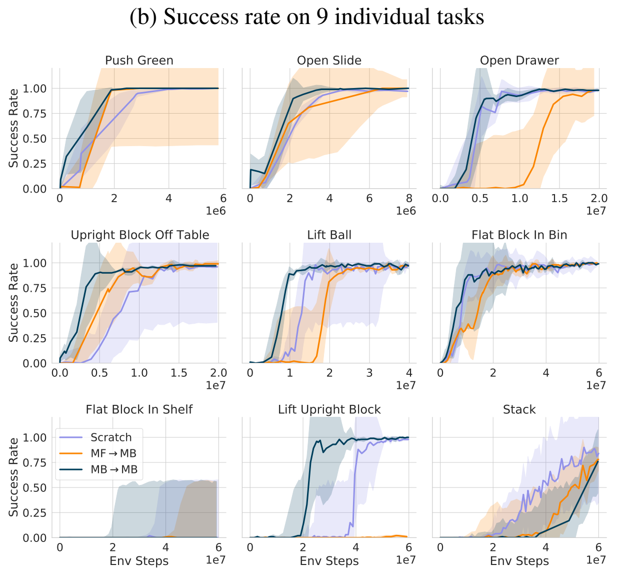


Figure 4. Success rate on RoboDesk. (a) Average success rate over 9 tasks and 3 seeds; (b) Success rate on individual tasks by taking the median over 3 seeds. We also report the standard deviation in both plots. Scratch: MuZero trained from scratch on individual tasks with extrinsic reward; MF $\rightarrow$ MB: model-free exploration transferred to MuZero agent fine-tuned on individual task; MB $\rightarrow$ MB: model-based exploration transferred to MuZero agent fine-tuned on individual task.

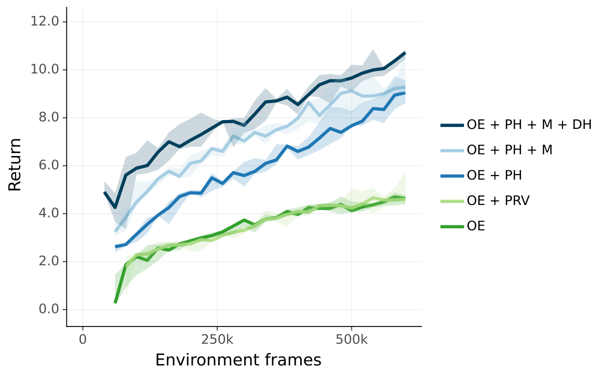


Figure 5. Here we perform an ablation analysis of which transferred components contribute to improved performance on Crafter. OE represents transferring only the observation encoder, OE + PRV additionally adds the prior reward and value heads, OE + PH adds prior reward, value, and policy heads, OE + PH + M adds the dynamics function, and OE + PH + M + DH adds all (equivalent to MB$\rightarrow$MB). For ablations, we use finetuning parameters specified in the appendix in Table 1.

Note that there are no novel objects introduced in the fine-tuning phase; the only differences between pre-training and fine-tuning are the rewards. As the action space for Robodesk is continuous, we adapted our model-based agent according to sampled MuZero (Hubert et al., 2021) for this environment. Similar to the experimental setup of Dadashi et al. (2021), the episode length is 2000 environment steps with action repeat of 5. However, we use the pixel observations only and sparse rewards for our study. The action space consists of five continuous dimensions in (-1, 1).

Meta-World (Yu et al., 2020) is a robotic control suite of up to 50 tasks with a SAWYER arm. We focus on a Meta-World v2 benchmark intended for task generalization, ML-10. In this benchmark, the training suite consists of ten tasks that differ from the testing suite of five tasks. Meta-world provides different challenges for model-based transfer compared to Crafter and RoboDesk: while the robotic arm is shared among all environments, the test environments include unseen objects and configurations which allows us to repurpose this benchmark to study model-based transfer in an out-of-distribution configuration. Note that unlike in the original ML-10 benchmark, our agents do not receive reward observations in the training environments. The action space consists of four continuous dimensions in (-1, 1).

We use pixels from the `corner3` angle as observations and do not use state information from the robot arm. We use sparse rewards which are the average episodic task success rates. We did not find any substantial differences in performance between approaches when using dense reward as these likely made the tasks relatively easy to solve. We explore and fine-tune on about 85 million frames. We also use sampled MuZero (Hubert et al., 2021) in this environment.

4. Results

Through our experiments, we generally find that model-based exploration and fine-tuning outperforms model-free approaches. In what follows, we investigate the questions from Section 3 in detail. We restate each question, give a summary of our answer, and then explain the results in more depth.

Q1: *Is there an advantage to an agent being model-based during unsupervised exploration and/or fine-tuning?*

A1: *Yes. Compared to model-free exploration, model-based* *exploration is more performant and transfers more effectively. Model-based fine-tuning also outperforms model-free fine-tuning.*

To answer this question, we compare different combinations of model-based and model-free pre-training and fine-tuning (i.e., MB→MB, MB→MF, MF→MB, MF→MF, and Scratch described in Section 3).

In Crafter (Figure 3), we find that model-based pre-training with model-based fine-tuning (MB→MB) substantially outperforms all other agent variations including training from scratch, achieving a score of 16.4 ± 1.5 and reward of 12.7 ± 0.4 at 1M environment steps. In fact, this result improves performance over state-of-the-art model-based agents without a pre-training phase, namely DreamerV3 (Hafner et al., 2023), and demonstrates the benefits of combining model-based learning with unsupervised pre-training. We find that the fully model-free agent (MF→MF) performs worst, consistent with other results demonstrating inferior model-free performance in other generalization settings (Anand et al., 2021). However, these results raise the question: is the improved performance of MB→MB due to pre-training, fine-tuning, or both? The performance of the MF→MB agent suggests that a good enough agent can leverage knowledge acquired during pre-training, even if the pre-training was with an inferior algorithm. However, the performance of the MB→MF agent—which provides no improvement over MF→MF—suggests that the fine-tuning algorithm plays an important role, too. It also suggests that additional knowledge may be encoded in the dynamics components, which the model-free agent cannot use.

As Crafter is, in part, designed to evaluate agents' ability to explore efficiently in the absence of reward, we have also quantified differences between the model-based and model-free exploration agents during pre-training. While both agents aim to maximise RND intrinsic rewards, model-based optimisation leads to significantly more effective exploration as measured by the average and individual success rates across achievements (see Figure 2).

In RoboDesk, we fine-tuned the pre-trained agents separately on nine core tasks and see a similar pattern of results as in Crafter. Due to resource limitations, we focused our comparisons on the agent trained from scratch and the two agents that provided the best transfer in Crafter, namely MB→MB and MF→MB. We find that fine-tuning from model-based exploration (MB→MB) consistently outperforms fine-tuning from the model-free counterpart (MF→MB) (Figure 4). Transferring from model-free also appears to result in a larger variance during fine-tuning. This, again, suggests model-based pre-training is superior to model-free pre-training. Additionally, compared to MuZero training from scratch (4a), warm-starting from the pre-trained weights of the model-based exploration agent shows benefits in terms of sample-efficiency but not in final performance. This is more evident on tasks such as *upright block off table*, *lift ball*, *lift upright block*, and *flat block in shelf* (4b). Interestingly, these are relatively difficult tasks to solve because they involve multiple steps or more challenging object manipulation. The improvements are less obvious on easy tasks (e.g. *push green*, *open slide*, *open drawer*). Similarly to Crafter, we found that model-based exploration outperformed its model-free counterpart when evaluating success-rates across different tasks during pre-training (Section B.2). When looking across tasks, we did not observe a correlation between success rates during pre-training and positive transfer during fine-tuning, suggesting that transfer is not mediated by the alignment of intrinsic and task rewards.

Q2: *What are the contributions of each component of a model-based agent for downstream task learning?*

A2: *Both the dynamics components (model and dynamics heads) and the prior heads (in particular, the prior policy) play an important role in transfer performance.*

To answer this question, we return to the Crafter environment and perform an ablation analysis on MB→MB where different parts of the fine-tuning agent are initialized from the model-based exploration agent. These ablations involve the observation encoder (OE), the prior heads (PH), the model (M) and the dynamics heads (DH), as described in Figure 1 and Section 3. In Figure 5, we show the performance of the fine-tuning agent when incrementally removing these various elements. First, removing the dynamics heads from the full agent results in a small drop in performance. Additionally removing the dynamics model results in a further drop, indicating that the dynamics components (both DH and M) do encode useful knowledge that is leveraged during transfer. However, it is not the *only* knowledge that can be transferred. When we remove the prior heads, we find performance further deteriorates. Moreover, this seems to be driven mostly by the policy prior (PP), as performance appears to be the same regardless of whether we include the prior reward and value heads (PRV) or not. In our model-based agent based off of MuZero, the policy prior plays an important role in guiding action selection in the search tree and, consistent with that, these ablations show that transferring both the dynamics model and policy prior contribute most to positive transfer. This result suggests that other model-based approaches to transfer which only transfer the model (e.g. Sekar et al., 2020) could benefit from transferring policy components as well.

Q3: *How well does the model-based agent deal with environmental shift between the unsupervised and downstream phases?*

A3: *Model-based agents can successfully transfer knowl-*

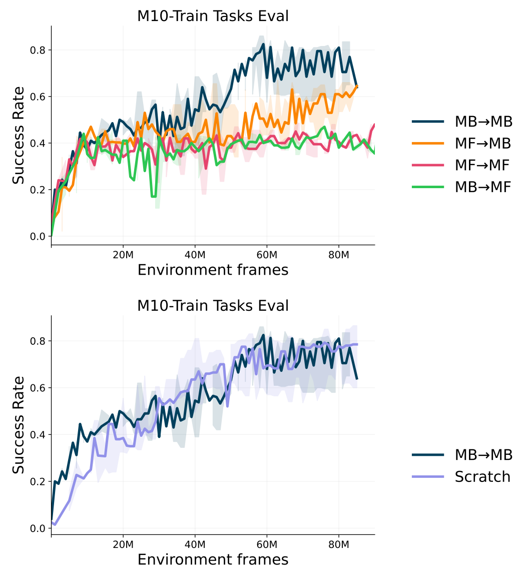


Figure 6. Here we report the return curves (top) of various fine-tuning strategies on Meta-World ML-10 test set.

edge to novel tasks when the dynamics are identical or in-distribution to those observed during training; however, transfer works less well to out-of-distribution dynamics.

Our results on Crafter and, to some extent, Robodesk suggest that model-based agents can successfully transfer knowledge from an unsupervised exploration phase to new reward functions when the pre-training and fine-tuning environments are identical or come from the same distribution. This setting is the most likely to benefit from learning and transferring dynamics models as the train and test environments differ only in the reward function. To better understand the limits of model-based transfer in the case of out-of-distribution environments, we also run a set of transfer experiments on the ML-10 Meta-World benchmark. Similarly to our results on Crafter and Robodesk, we find that that MB$\to$MB outperforms other pre-trained baselines (Figure 6, top). However, when compared with an agent trained from random initialization, MB$\to$MB has a small advantage only early in training, up to $\sim$20 million frames (Figure 6, bottom). Interestingly, model-free baselines appear to have inhibited transfer performance in this domain vs randomly initialized, and model-based transfer only helps mildly. We speculate this difference in results between Meta-World and the other two benchmarks is due to out-of-distribution environment dynamics. Due to the different objects between the train and test environment the agent is able to transfer its knowledge only about the dynamics in the robotic arm, and our empirical results suggest that that alone may be insufficient for positive transfer.

5. Discussion

In this paper we propose to study *when* and *why* model-based learning is beneficial for transferring knowledge. We make use of a framework which consists of an unsupervised pre-training phase in a no reward environment, followed by fine-tuning with task-specific reward functions. We argue that this is a suitable framework to ask (and answer) these questions. We present a specific instantiation of a model-based agent as a strong baseline, as well as a model-free counterpart using the same "backbone". We study a number of key factors that may contribute to successful transfer, namely the agent's ability to *explore* the environment even in the absence of any rewards and to summarize its experience and knowledge about the environment in the form of *representations*, *policies* and *dynamics models* that lend themselves to transfer. By conducting the same experiments on three distinct environments (in-distribution procedural, in-distribution identical, and out-of-distribution), we investigate transfer under environment shift.

Overall, we find that model-based optimization dominates model-free variants both during unsupervised exploration and fine-tuning performance. Our analysis reveals that transferring the dynamics model (and heads) as well as the prior policy learned during exploration contributes most to transfer. We observe that environment shift between unsupervised and downstream phases is detrimental for knowledge transfer. We speculate that the significantly stronger transfer results on Crafter, compared to Robodesk, are enabled partly by the procedural environment variations seen during pre-training.

While the model-based approach has a clear advantage for in-distribution transfer, our result on Meta-World suggests that out-of-distribution (OOD) generalization is still challenging. Within our framework, OOD performance could be potentially be improved if the agent could learn an accurate world model during the pre-training phase. One solution is to design better pre-training tasks—procedural environments with consistency. Better exploration algorithms can also help the world model to uncover the true dynamics rather than learning a biased model induced by a basic policy. Since our investigation suggests that policy also matters for successful transfer, an open research question how to induce the agent to learn more relevant policy during the pre-training phase.

Our study fixes the intrinsic exploration mechanism, this is because we focus on contrasting model-based versus model-free approaches rather than different choices of the intrinsic reward. Since our investigation shows that policy transfer matters, alternative approaches such as empowerment maybe beneficial.

Acknowledgements

We would like to thank Michael Laskin, Loic Matthey, and Simon Osindero for helpful discussions and feedback.

References

Anand, A., Walker, J. C., Li, Y., Vértes, E., Schrittwieser, J., Ozair, S., Weber, T., and Hamrick, J. B. Procedural generalization by planning with self-supervised world models. In *ICLR*, 2021. Cited on pages 1, 4, 7, and 12.

Baker, B., Akkaya, I., Zhokhov, P., Huizinga, J., Tang, J., Ecoffet, A., Houghton, B., Sampedro, R., and Clune, J. Video pretraining (vpt): Learning to act by watching unlabeled online videos. *arXiv preprint arXiv:2206.11795*, 2022. Cited on page 2.

Bellemare, M., Srinivasan, S., Ostrovski, G., Schaul, T., Saxton, D., and Munos, R. Unifying count-based exploration and intrinsic motivation. *Advances in neural information processing systems*, 29, 2016. Cited on page 3.

Bucher, B., Schmeckpeper, K., Matni, N., and Daniilidis, K. An adversarial objective for scalable exploration. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 2670–2677. IEEE, 2021. Cited on page 2.

Burda, Y., Edwards, H., Pathak, D., Storkey, A., Darrell, T., and Efros, A. A. Large-scale study of curiosity-driven learning. In *International Conference on Learning Representations*, 2018a. Cited on page 4.

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *International Conference on Learning Representations*, 2018b. Cited on pages 2, 3, and 4.

Byravan, A., Springenberg, J. T., Abdolmaleki, A., Hafner, R., Neunert, M., Lampe, T., Siegel, N., Heess, N., and Riedmiller, M. Imagined value gradients: Model-based policy optimization with tranferable latent dynamics models. In *Conference on Robot Learning*, pp. 566–589. PMLR, 2020. Cited on page 2.

Byravan, A., Hasenclever, L., Trochim, P., Mirza, M., Ialongo, A. D., Tassa, Y., Springenberg, J. T., Abdolmaleki, A., Heess, N., Merel, J., et al. Evaluating model-based planning and planner amortization for continuous control. *arXiv preprint arXiv:2110.03363*, 2021. Cited on page 2.

Campos, V., Sprechmann, P., Hansen, S., Barreto, A., Kapturowski, S., Vitvitskyi, A., Badia, A. P., and Blundell, C. Beyond fine-tuning: Transferring behavior in reinforcement learning. *arXiv preprint arXiv:2102.13515*, 2021. Cited on pages 1 and 2.

Coulom, R. Efficient selectivity and backup operators in Monte-Carlo tree search. In *International conference on computers and games*, 2006. Cited on page 4.

Dadashi, R., Hussenot, L., Vincent, D., Girgin, S., Raichuk, A., Geist, M., and Pietquin, O. Continuous control with action quantization from demonstrations. *arXiv preprint arXiv:2110.10149*, 2021. Cited on page 6.

Dasari, S., Ebert, F., Tian, S., Nair, S., Bucher, B., Schmeckpeper, K., Singh, S., Levine, S., and Finn, C. Robonet: Large-scale multi-robot learning. *arXiv preprint arXiv:1910.11215*, 2019. Cited on page 2.

Dayan, P., Hinton, G. E., Neal, R. M., and Zemel, R. S. The helmholtz machine. *Neural computation*, 7(5):889–904, 1995. Cited on page 1.

Eysenbach, B., Gupta, A., Ibarz, J., and Levine, S. Diversity is all you need: Learning skills without a reward function. *arXiv preprint arXiv:1802.06070*, 2018. Cited on page 3.

Gregor, K., Rezende, D. J., and Wierstra, D. Variational intrinsic control. *arXiv preprint arXiv:1611.07507*, 2016. Cited on page 3.

Grill, J.-B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., et al. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33:21271–21284, 2020. Cited on page 12.

Guo, Z. D., Thakoor, S., Pîslar, M., Pires, B. A., Altché, F., Tallec, C., Saade, A., Calandriello, D., Grill, J.-B., Tang, Y., et al. Byol-explore: Exploration by bootstrapped prediction. *arXiv preprint arXiv:2206.08332*, 2022. Cited on page 4.

Ha, D. and Schmidhuber, J. Recurrent world models facilitate policy evolution. In *NeurIPS*, 2018. Cited on page 1.

Haarnoja, T., Zhou, A., Hartikainen, K., Tucker, G., Ha, S., Tan, J., Kumar, V., Zhu, H., Gupta, A., Abbeel, P., et al. Soft actor-critic algorithms and applications. *arXiv preprint arXiv:1812.05905*, 2018. Cited on page 1.

Hafner, D. Benchmarking the spectrum of agent capabilities. In *International Conference on Learning Representations*, 2021. Cited on pages 2, 3, 4, and 5.

Hafner, D., Pasukonis, J., Ba, J., and Lillicrap, T. Mastering diverse domains through world models. *arXiv preprint arXiv:2301.04104*, 2023. Cited on pages 1, 5, and 7.

Hansen, S., Dabney, W., Barreto, A., Van de Wiele, T., Warde-Farley, D., and Mnih, V. Fast task inference with variational intrinsic successor features. *arXiv preprint arXiv:1906.05030*, 2019. Cited on page 1.

Hazan, E., Kakade, S., Singh, K., and Van Soest, A. Provably efficient maximum entropy exploration. In *International Conference on Machine Learning*, pp. 2681–2691. PMLR, 2019. Cited on page 3.

Hospedales, T., Antoniou, A., Micaelli, P., and Storkey, A. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021. Cited on page 1.

Hubert, T., Schrittwieser, J., Antonoglou, I., Barekatain, M., Schmitt, S., and Silver, D. Learning and planning in complex action spaces. In *ICML*, 2021. Cited on pages 4, 6, and 13.

Kannan, H., Hafner, D., Finn, C., and Erhan, D. Robodesk: A multi-task reinforcement learning benchmark. https://github.com/google-research/robodesk, 2021. Cited on pages 2, 4, and 5.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. Cited on pages 12 and 14.

Kirk, R., Zhang, A., Grefenstette, E., and Rocktäschel, T. A survey of generalisation in deep reinforcement learning. *arXiv preprint arXiv:2111.09794*, 2021. Cited on pages 1 and 2.

Klyubin, A. S., Polani, D., and Nehaniv, C. L. Empowerment: a universal agent-centric measure of control. *2005 IEEE Congress on Evolutionary Computation*, 1:128–135 Vol.1, 2005. Cited on page 3.

Kocsis, L. and Szepesvári, C. Bandit based Monte-Carlo planning. In *European conference on machine learning*, 2006. Cited on page 4.

Lambert, N., Wulfmeier, M., Whitney, W., Byravan, A., Bloesch, M., Dasagi, V., Hertweck, T., and Riedmiller, M. The challenges of exploration for offline reinforcement learning. *arXiv preprint arXiv:2201.11861*, 2022. Cited on page 3.

Laskin, M., Yarats, D., Liu, H., Lee, K., Zhan, A., Lu, K., Cang, C., Pinto, L., and Abbeel, P. Urlb: Unsupervised reinforcement learning benchmark. *arXiv preprint arXiv:2110.15191*, 2021. Cited on pages 2, 3, and 4.

Lowrey, K., Rajeswaran, A., Kakade, S., Todorov, E., and Mordatch, I. Plan online, learn offline: Efficient learning and exploration via model-based control. *arXiv preprint arXiv:1811.01848*, 2018. Cited on page 3.

Lutter, M., Hasenclever, L., Byravan, A., Dulac-Arnold, G., Trochim, P., Heess, N., Merel, J., and Tassa, Y. Learning dynamics models for model predictive agents. *arXiv preprint arXiv:2109.14311*, 2021. Cited on page 2.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. Human-level control through deep reinforcement learning. *nature*, 518(7540): 529–533, 2015. Cited on page 1.

Nagabandi, A., Kahn, G., Fearing, R. S., and Levine, S. Neural network dynamics for model-based deep reinforcement learning with model-free fine-tuning. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 7559–7566. IEEE, 2018. Cited on page 2.

Osband, I., Blundell, C., Pritzel, A., and Van Roy, B. Deep exploration via bootstrapped dqn. *Advances in neural information processing systems*, 29, 2016. Cited on page 3.

Ostrovski, G., Bellemare, M. G., Oord, A., and Munos, R. Count-based exploration with neural density models. In *International conference on machine learning*, pp. 2721–2730. PMLR, 2017. Cited on page 3.

Oudeyer, P.-Y., Kaplan, F., and Hafner, V. V. Intrinsic motivation systems for autonomous mental development. *IEEE transactions on evolutionary computation*, 11(2): 265–286, 2007. Cited on page 3.

Pathak, D., Agrawal, P., Efros, A. A., and Darrell, T. Curiosity-driven exploration by self-supervised prediction. *CoRR*, abs/1705.05363, 2017. URL http://arxiv.org/abs/1705.05363. Cited on page 3.

Pathak, D., Gandhi, D., and Gupta, A. Self-supervised exploration via disagreement. In *International conference on machine learning*, pp. 5062–5071. PMLR, 2019. Cited on page 3.

Schmidhuber, J. Curious model-building control systems. In *Proc. international joint conference on neural networks*, pp. 1458–1463, 1991a. Cited on page 1.

Schmidhuber, J. A possibility for implementing curiosity and boredom in model-building neural controllers. In *Proc. of the international conference on simulation of adaptive behavior: From animals to animats*, pp. 222–227, 1991b. Cited on page 3.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 2020a. Cited on page 4.

Schrittwieser, J., Antonoglou, I., Hubert, T., Simonyan, K., Sifre, L., Schmitt, S., Guez, A., Lockhart, E., Hassabis, D., Graepel, T., et al. Mastering atari, go, chess and shogi by planning with a learned model. *Nature*, 588(7839): 604–609, 2020b. Cited on pages 1, 2, and 4.

Schrittwieser, J., Hubert, T., Mandhane, A., Barekatain, M., Antonoglou, I., and Silver, D. Online and offline reinforcement learning by planning with a learned model. *arXiv preprint arXiv:2104.06294*, 2021. Cited on pages 4 and 12.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. Cited on page 1.

Schwarzer, M., Anand, A., Goel, R., Hjelm, R. D., Courville, A., and Bachman, P. Data-efficient reinforcement learning with self-predictive representations. In *International Conference on Learning Representations*, 2020. Cited on pages 1 and 2.

Sekar, R., Rybkin, O., Daniilidis, K., Abbeel, P., Hafner, D., and Pathak, D. Planning to explore via self-supervised world models. In *International Conference on Machine Learning*, pp. 8583–8592. PMLR, 2020. Cited on pages 1, 2, 3, and 7.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018. Cited on page 4.

Stanić, A., Tang, Y., Ha, D., and Schmidhuber, J. Learning to generalize with object-centric agents in the open world survival game crafter. *arXiv preprint arXiv:2208.03374*, 2022. Cited on page 5.

Sutton, R. S. Dyna, an integrated architecture for learning, planning, and reacting. *ACM Sigart Bulletin*, 2(4):160–163, 1991. Cited on page 1.

Tang, H., Houthooft, R., Foote, D., Stooke, A., Xi Chen, O., Duan, Y., Schulman, J., DeTurck, F., and Abbeel, P. # exploration: A study of count-based exploration for deep reinforcement learning. *Advances in neural information processing systems*, 30, 2017. Cited on page 3.

Tolman, E. C. Cognitive maps in rats and men. *Psychological review*, 55(4):189, 1948. Cited on page 1.

Vinyals, O., Babuschkin, I., Czarnecki, W. M., Mathieu, M., Dudzik, A., Chung, J., Choi, D. H., Powell, R., Ewalds, T., Georgiev, P., et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575 (7782):350–354, 2019. Cited on page 2.

Watters, N., Matthey, L., Bosnjak, M., Burgess, C. P., and Lerchner, A. Cobra: Data-efficient model-based rl through unsupervised object discovery and curiosity-driven exploration. *arXiv preprint arXiv:1905.09275*, 2019. Cited on page 2.

Yarats, D., Fergus, R., Lazaric, A., and Pinto, L. Reinforcement learning with prototypical representations. In *International Conference on Machine Learning*, pp. 11920–11931. PMLR, 2021. Cited on page 3.

Yu, T., Quillen, D., He, Z., Julian, R., Hausman, K., Finn, C., and Levine, S. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pp. 1094–1100. PMLR, 2020. Cited on pages 2, 4, 5, and 6.

Zhang, J., Li, S., and Zhang, C. Cup: Critic-guided policy reuse. In *Advances in Neural Information Processing Systems*. Cited on page 3.

Zhu, Z., Lin, K., and Zhou, J. Transfer learning in deep reinforcement learning: A survey. *arXiv preprint arXiv:2009.07888*, 2020. Cited on page 2.

A. Agent Details

A.1. Network Architecture

Both the model-free and model-based agents utilize the same network architectures. It's identical to the ones used in MuZero ReAnalyse (Schrittwieser et al., 2021). The pixel input resolution is 96×96 for Crafter and Meta-World; 64×64 for RoboDesk. The images are first sent through a convolutional stack that downsamples to an 6×6 tensor (for Meta-World) or 8×8 tensor (for RoboDesk and Crafter). This tensor then serves as input to the encoder. Both the encoder and the dynamics model were implemented by a ResNet with 10 blocks, each block containing 2 layers. Each layer of the residual stack was convolutional with a kernel size of 3x3 and 256 planes.

A.2. Model-based Agent

MuZero + SPR To add the SPR as an auxiliary loss, we add a projection and a prediction head, similar to Grill et al. (2020). The projection and prediction networks have an identical architecture: a two convolutional layers with stride 1 and kernel size 3 and relu non-linearity in between; then flatten the output to a vector. The input to the projection network is the output of dynamics function; the input to the prediction network is the output of the projector. Since the dynamics model unrolls n steps into the future, it results in n prediction vectors x_n. The target vector y_n is the output of the projector computed with the target network weights. The target network weights is the same as the one used to compute the temporal difference loss in MuZero. The input for computing the target vectors are the corresponding image observation at that future step n. The encoder of MuZero uses 15 past images as history. However, when we compute the target vectors, our treatment of the encoding history is different from that of the agent; instead of stacking all of the historical images ($n - 15 ... n - 1$) up to the corresponding step n, we simply replace the history stack with 15 copies of the image at the current step n. This is the same treatment as in Anand et al. (2021). We then attempt to match x_n with the corresponding target vector y_n with a cosine distance between x_i and y_i.

Hyper-parameters Table 1 listed the hyper-parameters of our model-based agent for both pre-training and fine-tuning. If no special indication in Table 1, pre-training and fine-tuning use the same value. For all the agents, we use the Adam (Kingma & Ba, 2014) optimizer. The batch size for the training is 1024. The Scratch baseline uses the pre-train hyper-parameters, but always with initial learning rate of 10^{-4} and cosine learning rate schedule.

Table 1. Hyper-parameters for Model-based Pre-training and Fine-tuning

HYPER-PARAMETER	CRAFTER (*Pre-train / Fine-tune*)	ROBODESK (*Pre-train / Fine-tune*)	META-WORLD (*Pre-train / Fine-tune*)
TRAINING			
Model Unroll Length	5	5	5
TD-Steps	5	0	0
ReAnalyse Fraction	0.8 / 0.99	0.925	0.925
Replay Size (in sequences)	50000	2000	2000
MCTS			
Number of Simulations	50	50	50
UCB-constant	1.25	1.25	1.25
Number of Samples	n/a	20	20
SELF-SUPERVISION			
SPR Loss Weight	1.0	1.0	1.0
OPTIMIZATION			
Initial Learning Rate	10^{-4} / 10^{-5}	10^{-4}	10^{-4} / 10^{-5}
Learning Rate Schedule	cosine / constant	constant / cosine	cosine

A.3. Model-free Agent

Q-Learning The Q-Learning setup is identical to Anand et al. (2021). We describe here for completeness of the paper. Our controlled Q-Learning agent has an identical network architecture to the model-based agent, but modifies it in a few key ways to make it model-free rather than model-based.

The Q-Learning baseline uses n-step targets for action value function. Given a trajectory $\{s_t, a_t, r_t\}_{t=0}^{T}$, the target action value is computed as follow

$$Q_{target}(s_t, a_t) = \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \max_{a \in \mathcal{A}} Q_\xi(s_{t+n}, a) \tag{2}$$

Where Q_ξ is the target network whose parameter ξ is updated every 100 training steps.

In order to make the model architecture most similar to what is used in the MuZero agent, we decompose the action value function into two parts: a reward prediction $\hat{r}$ and a value prediction V, and model these two parts separately. The total loss function is, therefore, $\mathcal{L}_{total} = \mathcal{L}_{reward} + \mathcal{L}_{value}$. The reward loss is exactly the same as that of MuZero. For the value loss, we can decompose Equation 2 in the same way:

$$\begin{aligned} Q_{target}(s_t, a_t) &= \hat{r}_t + \gamma V_{target}(s) \\ &= \sum_{i=0}^{n-1} \gamma^i r_{t+i} + \gamma^n \max_{a \in \mathcal{A}} \left(\hat{r}_{t+n} + \gamma V_\xi(s^{'}) \right) \\ \implies V_{target}(s) &= \sum_{i=1}^{n-1} \gamma^{i-1} r_{t+i} + \gamma^{n-1} \max_{a \in \mathcal{A}} \left(\hat{r}_{t+n} + \gamma V_\xi(s^{'}) \right) \end{aligned} \tag{3}$$

Since the reward prediction should be taken care of by $\mathcal{L}_{reward}$ and it usually converges fast, we assume $\hat{r}_t = r_t$ and the target is simplified to Equation 3. We can then use this value target to compute the value loss $\mathcal{L}_{value} = \text{CE}(V_{target}(s), V(s))$.

In RoboDesk and Meta-World, since it has a continuous action space, maximizing over the entire action space is infeasible. We follow the Sampled Muzero approach (Hubert et al., 2021) and maximize only over the sampled actions.

Hyper-parameters Table 2 lists the hyper-parameters for our model-free baselines. For the model-free agent, we only unroll a single-step for computing action and values.

Table 2. Hyper-parameters for Model-free Pre-training and Fine-tuning

Hyper-parameter	Crafter (*Pre-train / Fine-tune*)	RoboDesk (*Pre-train*)	Meta-World (*Pre-train / Fine-tune*)
Training			
Model Unroll Length	1	1	1
TD-Steps	5	1	1
ReAnalyse Fraction	0.75 / 0.99	0.945	0.9325
Replay Size (in sequences)	50000	2000	2000
Self-Supervision			
SPR Loss Weight	1.0	1.0	1.0
Optimization			
Initial Learning Rate	10^{-4} / 10^{-5}	10^{-4}	10^{-4} / 10^{-5}
Learning Rate Schedule	cosine / constant	constant / cosine	cosine

A.4. Random Network Distillation

Our modified version of RND utilizes the same encoder architecture as the agent network described previously. A projector network takes the output of the encoder and projects into a vector. The projector network is of the same architecture as the projector/predictor described for SPR. The target vector z is computed with randomly initialized weights. The prediction vector $\hat{z}$ is computed with the learned weights. The observation encoder receives training signals from both the RND loss and the usual MuZero losses. The prediction error is the L2 distance between the prediction and target $e = (z - \hat{z})^2$.

The intrinsic reward is computed as a function of the prediction error e. We keep an exponential moving average of the error with a decay of 0.99 and bias correction technique applied (Kingma & Ba, 2014). This gives us the mean and standard deviation $\hat{e}$ and $\hat{\sigma}_e$. Then the reward is the normalized prediction error $r_{intrinsic} = (e - \hat{e})/\hat{\sigma_e}$. We do not apply any clipping to the reward.

B. Additional Results

B.1. Crafter

Figure 7 reports the success rates of the sub-tasks after fine-tuning. We find that the performance margins of MB$\rightarrow$MB are particularly large on the more advanced tasks—especially those requiring or involving stone. We also trained a model-based agent from scratch with combined extrinsic and intrinsic rewards. We found a weighting of the intrinsic (RND) loss relative to the extrinsic loss around 0.001 to be optimal with a score of 2.9 ± 0.1 and reward of 4.5 ± 0.1 across three seeds.

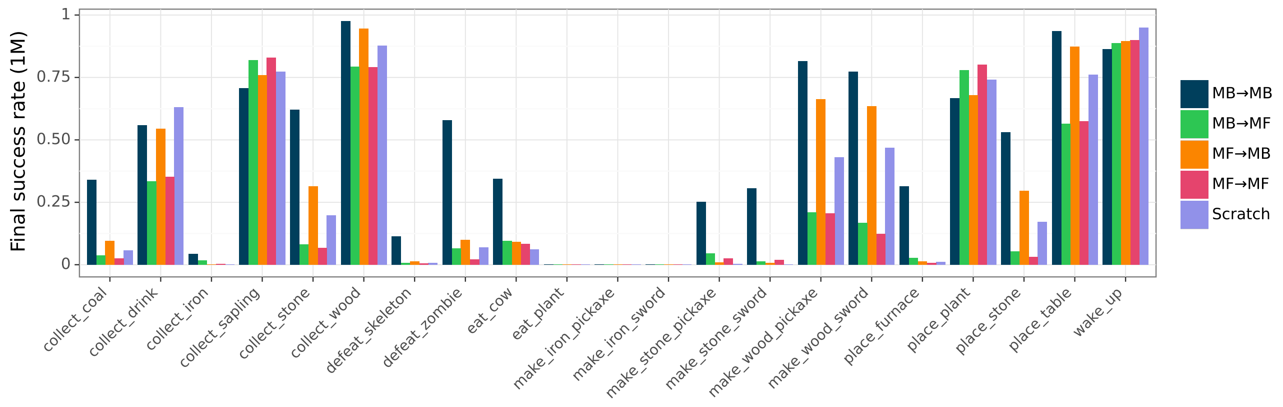


Figure 7. Success rates on Crafter after fine-tuning. Comparison of sub-task success rates of various transfer agents and baselines.

B.2. RoboDesk

Figure 8 shows the success rate on all the 18 tasks of RoboDesk during pre-training. Unlike Crafter, we do not observe significant correlation between exploration and task rewards.

In addition to the success rate reported in Figure 4, we report the return curve of model-based and model-free transfer agents, as well as the training from scratch agent, in Figure 9. The relative comparison between agents judging from return is very similar to that from the success rate we presented in the main paper.

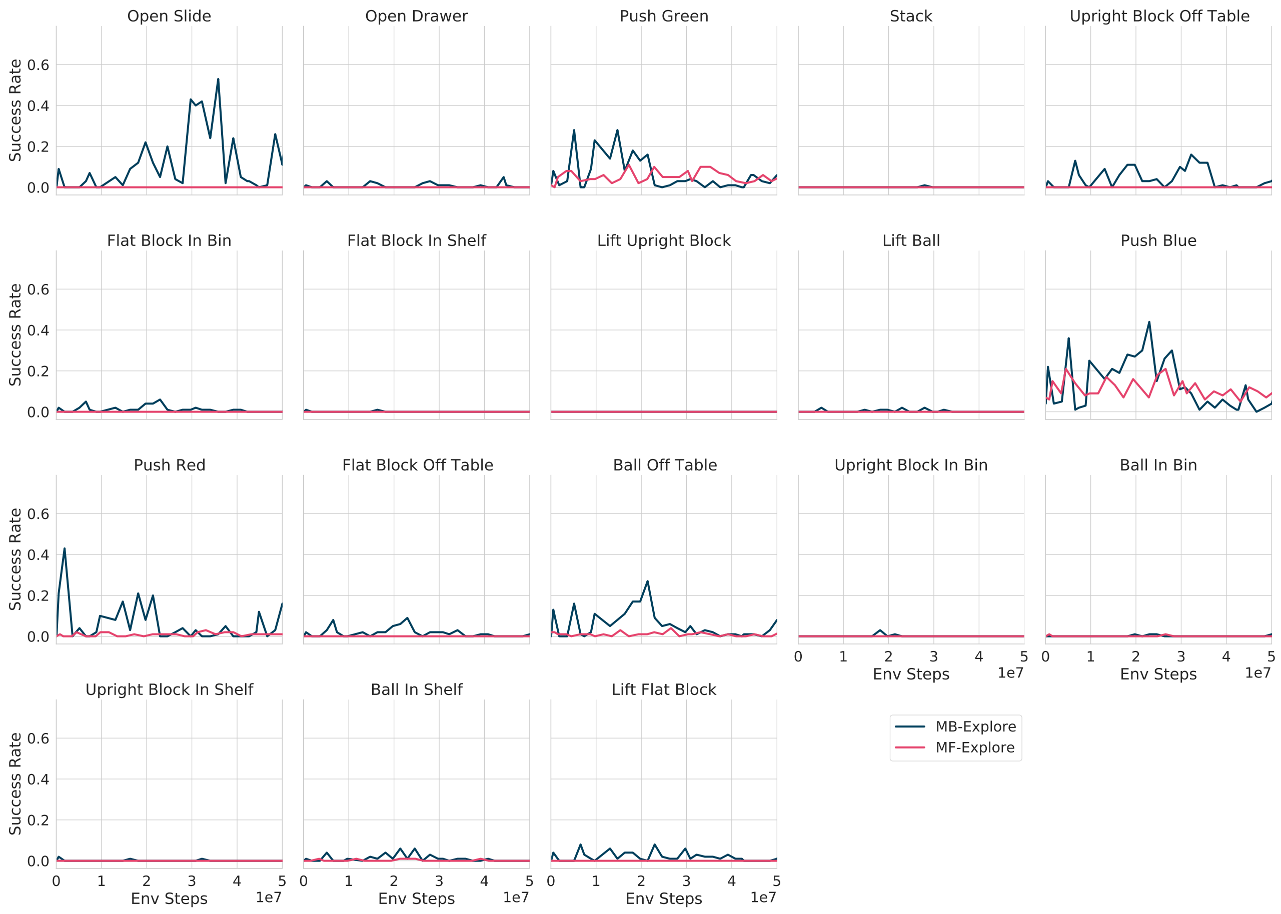


Figure 8. Success rates on all 18 RoboDesk tasks during pre-training (reported for a single seed).

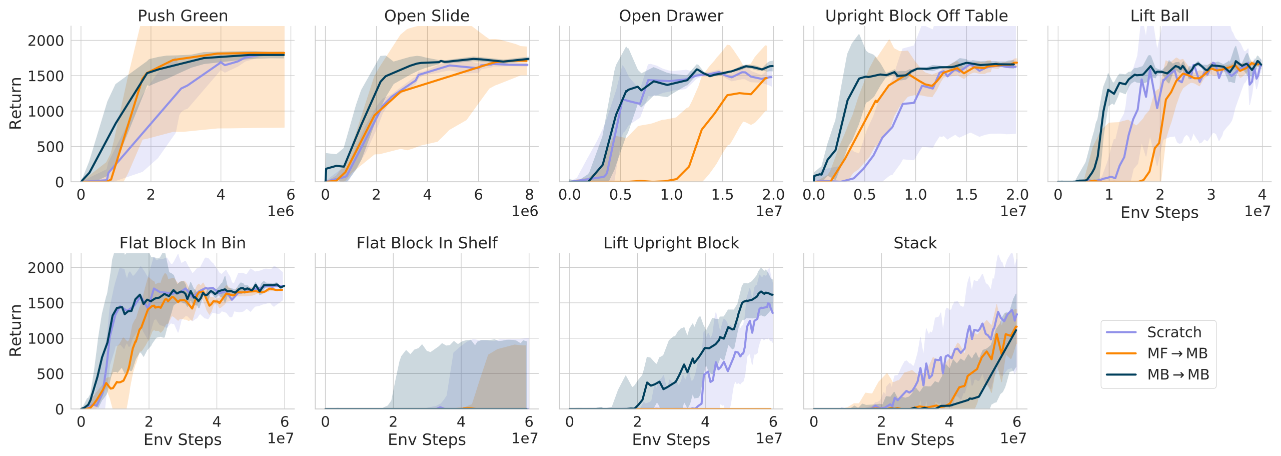
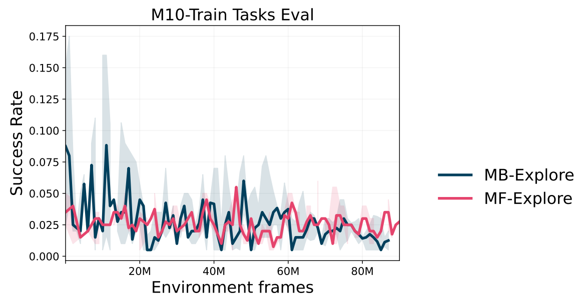


Figure 9. Fine-tuning return curve on 9 RoboDesk core tasks. We take the median of 3 seeds and report the standard deviation.

B.3. Meta-World

Similar to RoboDesk, Figure 10a shows that there is no correlation between exploration and downstream tasks rewards. Figure 10b provides the results for an ablation study on Meta-World. However, since there is no tangible benefits from the transfer in Meta-World, the various transfer setting results in similar performance.

(a) Average success rate on train tasks



(b) Average success rate on test tasks

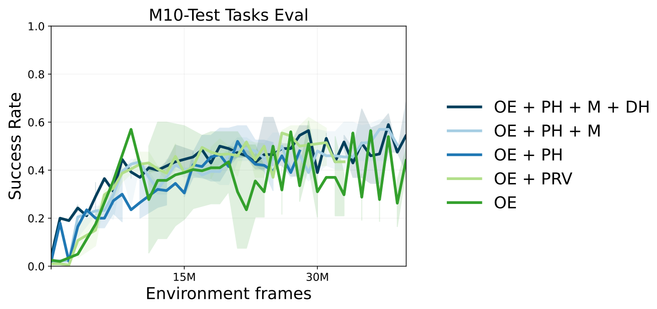


Figure 10. Average success rate on Meta-World. (a) Average success rates on train tasks during pre-training phase. (b) Average success rate on test tasks during fine-tuning phase for the ablation analysis.