PLANNING TO EXPLORE VIA SELF-SUPERVISED WORLD MODELS

Ramanan Sekar

A THESIS

in

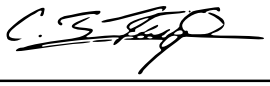
Robotics

Presented to the Faculties of the University of Pennsylvania in Partial

Fulfillment of the Requirements for the Degree of of Master of Science in Engineering

2020

Professor Kostas Daniilidis
Supervisor of Thesis Signature



Professor Camillo J. Taylor
Graduate Group Chairperson Signature

PLANNING TO EXPLORE VIA SELF-SUPERVISED WORLD MODELS

Dedicated to my father Sekar, mother Raji and sister Indhu

ACKNOWLEDGEMENT

I would like to thank my advisor Professor Kostas Daniilidis and Oleh Rybkin for their mentorship, and patiently advising me and helping me throughout. Their mentorship has truly shaped my thought process and perspectives, and has instilled a strong work ethic in me that I will carry forward. I would also like to thank Deepak Pathak for this opportunity by involving me in the project, and for giving access to compute resources at UC Berkeley and running experiments at Facebook AI, and for assisting me in finishing this project. Being able to work with him and seeing him make decisions has had a great impact on the way I approach problems. I also want to thank Bernadette Bucher, for giving me the opportunity to work with her in the curiosity project for my Independent study, and for helping me find my thesis topic when I was unsure. Being able to interact with these phenomenally talented people, seeing them work and solve problems, and learning from them has been a joy and a tremendous privilege, and is something that I will cherish and reflect upon in the future. I also want to thank my friends who've been understanding when I devoted a majority of my time to this project, and gave me company when I needed it.

ABSTRACT

PLANNING TO EXPLORE VIA SELF-SUPERVISED WORLD MODELS

Ramanan Sekar

Professor Kostas Daniilidis

To solve complex tasks, intelligent agents first need to explore their environments. However, providing manual feedback to agents during exploration can be challenging. This work focuses on task-agnostic exploration, where an agent explores a visual environment without yet knowing the tasks it will later be asked to solve. While current methods often learn reactive exploration behaviors to maximize retrospective novelty, we learn a world model trained from images to plan for expected surprise. Novelty is estimated as ensemble disagreement in the latent space of the world model. Exploring and learning the world model without rewards, our approach, Plan2Explore (P2E), efficiently adapts to a range of control tasks with high-dimensional image inputs. Video results can be accessed at: `https://sites.google.com/view/planning-to-explore/`

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. Introduction

The dominant approach in sensorimotor learning is to train the agent on a fixed set of tasks specified apriori either via rewards using reinforcement learning, or from demonstrations using imitation learning. We are interested in the case where an agent is consistently confronted with new tasks it has never experienced before. In this case, learning every task from scratch can be inefficient. How can an agent solve many tasks quickly in either zero or few-shot manner? Instead of training the agent on each task as it becomes available, one approach is let the agent first explore the environment in a task-agnostic manner. To explore complex environments with high-dimensional inputs in the absence of rewards, the agent needs to follow a form of intrinsic motivation or curiosity.

Intrinsic motivation can be defined as aiming to learn about the environment, by striving for novelty. For example, this can be implemented by seeking out parts of the world the agent cannot yet predict accurately (Schmidhuber (1991b); Oudeyer et al. (2007); Pathak et al. (2017)), encouraging diversity in agent's trajectories (Klyubin et al. (2005); Eysenbach et al. (2018)), or by aiming to visit rare states (Poupart et al. (2006); Lehman and Stanley (2011); Bellemare et al. (2016); Burda et al. (2018b)). Most of these self-supervised approaches internally build a model-free policy to explore, which can later be finetuned to specialize to unseen tasks given as a reward function. However, finetuning often relies on model-free agents that require large amounts of interaction with the environment to learn each downstream task.

Model-free approaches are not only slow in adapting to downstream tasks, they are also inefficient during the exploration phase. They optimize for retrospective novelty, which conflicts with the non-stationary nature of curiosity. The agent usually first

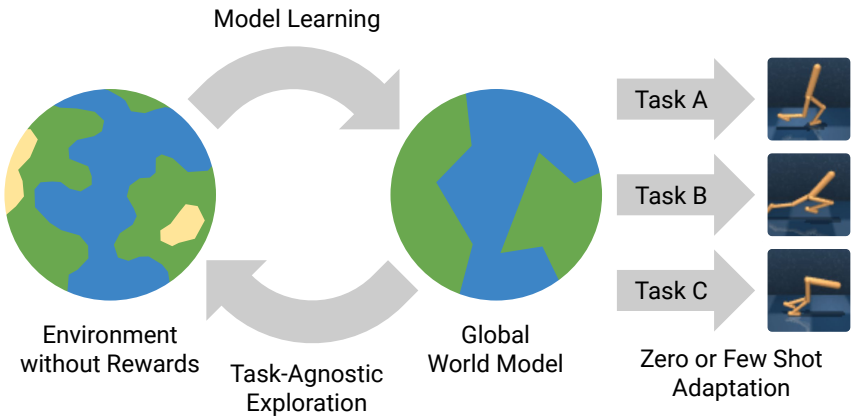


Figure 1: The problem setting. We train an agent to explore to efficiently learn the dynamics in an unsupervised way, and later adapt to the task at hand, such as walking, running, flipping, or standing

interacts in the environment, collect some samples and then these intrinsic rewards are calculated via agent's current estimate of density model or prediction model. These model-free exploration policies only maximizes what was novel in the past, and hence keep searching aimlessly hoping to stumble upon interesting states.

In this work, we address both of these challenges — inefficient exploration as well as adaptation — within a common framework while learning directly from high-dimensional image inputs. Instead of maximizing intrinsic rewards in retrospect, our agent, Plan2Explore (P2E), explicitly plans for expected novelty in a forward-looking manner. For this, P2E first learns a world model from its past interaction, and uses its current estimate to imagine future trajectories and learn a strategy for exploring novel transitions. The policy is optimized purely from imagined trajectories to maximize the intrinsic rewards computed by the model itself. Planning to explore using the learned model lets the agent execute only actions it deems optimal for its current intrinsic reward. Moreover, when faced with downstream tasks, the same model can be used to plan for goal states in a zero-shot manner.

The key challenge in planning to explore lies both in training accurate world models from high-dimensional inputs, as well as in the choice of exploration objective that explores the environment efficiently enough for successful adaptation to downstream tasks. We focus on world models that predict ahead an a compact latent space. Predicting future compact representations facilitates accurate long-term predictions and lets us efficiently predict thousands of future sequences in parallel for policy learning.

An ideal objective for exploration encourages actions that lead to states with high uncertainty due to missing knowledge (epistemic uncertainty) while avoiding the inherent stochastic parts of environment (aleatoric uncertainty). Exploring such states allows the agent to obtain samples that are maximally informative of the environment, leading to improved planning ability. We measure epistemic uncertainty as the disagreement of an ensemble of one-step predictors trained alongside the world model. The disagreement reduces to zero after seeing enough samples even if the environment is stochastic because over time, their predictions approach the mean of the distribution. We further show that this procedure is closely related to maximizing the expected amount of information obtained by executing the plan.

We evaluate P2E on 11 control tasks with image inputs, without access to low-dimensional states nor any task rewards during exploration, where it achieves state-of-the-art zero-shot adaptation performance. Moreover, we empirically quantify the following scientific questions:

- How does planning to explore via latent disagreement compare to a supervised oracle and other model-free and model-based intrinsic reward objectives?
- How much task-specific experience is enough to finetune a task-agnostic model

to reach the task performance of task-specific agent?

- To what degree does a task-agnostic model generalize to unseen tasks compared to a task-specific model trained on a different task in the same environment?
- What is the advantage of maximizing expected novelty in comparison to retrospective novelty?

2. Background and Related Works

We can now go into the background material and prior work done that our work is building on top of. This specifically includes ideas on model-based reinforcement learning methods, ideas on exploration and using intrinsic motivation as an objective for exploration, and finally, some ideas on active-learning.

2.1. Model-based Reinforcement Learning

We proceed to take a look into Model-Based Reinforcement Learning methods. In general, the reinforcement learning setting is laid out in Figure 2, wherein we have an agent interacting with the world, and the agent acts based on a series of observations it receives from the world, and the actions executed by the agent on the world results in the transition of the world to a new state, along with a scalar value called the reward. This is described in general by a Partially Observed Markov Decision Process (POMDP), $\mathcal{M} = \{\mathcal{S}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \mathcal{E}, r\}$, where $\mathcal{S}$ denotes the state space, $\mathcal{A}$ denotes the action space, $\mathcal{O}$ denotes the observation space, $\mathcal{T}$ denotes the transition operator which model the transition probabilities of moving the state of the world from s_t to s_{t+1} given an action a_t, $\mathcal{E}$ the emission probability $p(o_t|s_t)$, and $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function. The MDP is said to be partially observed when ground truth states $s_t \in \mathcal{S}$ are not directly accessible by the agent, and the only things that the agent sees are the observations $o_t \in \mathcal{O}$. The subscript t denotes the timestep in the episode, which could start from $t = 1$. The states $\mathcal{S}$ are a sufficient statistic to describe the system completely, and when the agent has access to these, then the system is said to be a completely observable MDP. The transition functions on the states are said to follow the Markov property if the transition to the future state is conditionally independent of the past states given the current state and action.

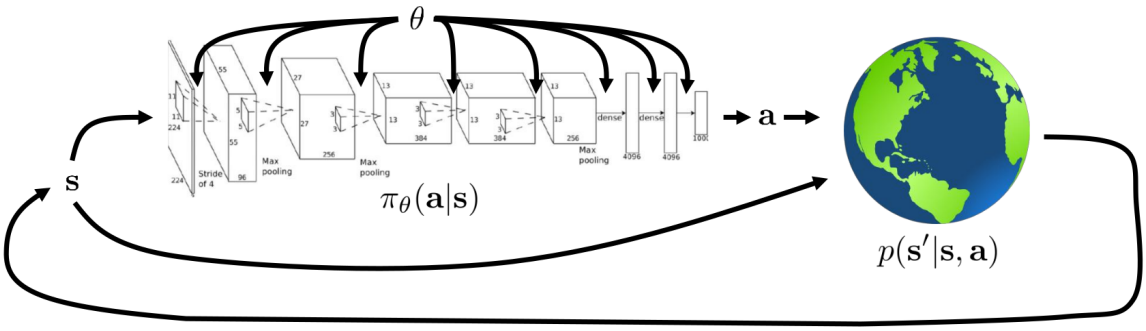


Figure 2: The standard setup of a reinforcement learning problem, where an agent (represented by a parametrized policy π_θ) is acting on a world, which transitions to a new state, giving back the agent a reward for the action it just took (Levine (2019))

The agent is represented as a conditional probability distribution $\pi(\mathbf{a_t}|\mathbf{s_t})$, which represents the action that should be taken given a state at a specific timestep. As represented in Figure 2, this is parametrized by a deep neural network with weights θ. This entire setup is represented as a graphical model in in Figure 3.

If the interaction between the agent and the world goes on for T timesteps with the episode represented by $(\mathbf{s_1}, \mathbf{a_1}, ..., \mathbf{s_T}, \mathbf{a_T})$, then the probability that this episode rollout, occurs is

$$p_\theta(\mathbf{s_1}, \mathbf{a_1}, ..., \mathbf{s_T}, \mathbf{a_T}) = p_\theta(\tau) = p(\mathbf{s_1}) \prod_{t=1}^{T} \pi_\theta(\mathbf{a_t}|\mathbf{s_t}) p(\mathbf{s_{t+1}}|\mathbf{s_t}, \mathbf{a_t}) \tag{2.1}$$

The objective of RL is to come up with an optimal policy π_{θ^*} such that

$$\theta^* = \arg\max_\theta \mathbb{E}_{\tau \sim p_\theta(\tau)}[\sum_t r(\mathbf{s_t}, \mathbf{a_t})] \tag{2.2}$$

This problem setting lends itself to many different ways of solving. If the MDP is fully known, then dynamic programming can be used to solve it. In most real case scenarios, we would not have access to the reward function r or the transition tensor $\mathcal{T}$, giving rise to RL methods with two major bifurcations: model-free and model-based.

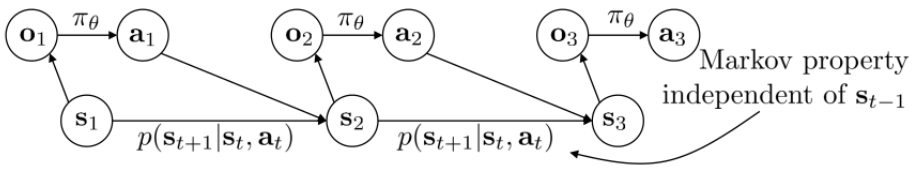


Figure 3: Graphical model of the RL problem setup (Levine (2019))

Model-Free methods usually rely on policy gradients to perform gradient ascent on the policy parameters, without regarding the world model (i.e the transition tensor) (Williams (1992); Mnih et al. (2016); Schulman et al. (2017, 2015)). The other class of methods are the model-based methods, which typically try to learn the world model, on top of learning a policy. After learning this world model, one can take several approaches, of either using model-free optimization to learn a policy (Kaiser et al. (2019); Weber et al. (2017)) or using the model to learn a policy in imagination (Ha and Schmidhuber (2018)) Learning an accurate world model for long-horizon planning is a difficult problem mainly because of modeling inaccuracies, not being able to model multiple modes of policy execution. In low dimensional environments, prior work has been effective in doing this (Chua et al. (2018a); Gal (2016); Amos et al. (2018); Henaff et al. (2019)), but these environments assume a fully observable system with access to states and reward functions. In high dimensional observational space, such as images, learning a compact latent representation for executing a dynamics model has been limited to very simple tasks (Watter et al. (2015); Banijamali et al. (2017)). However, modern advances have been able to succeed to some extent in this. PlaNet (Hafner et al. (2018)) learns the world model jointly and solves visual locomotion tasks by latent online planning. Building on this, Dreamer (Hafner et al. (2019)) uses the same world model learnt by PlaNet but learns a parametric policy, along with a value estimate, in the imagination of the world model by backpropagating the gradients through the model instead of using a policy gradient optimization method.

Algorithm 1 PlaNet

initialize: Dataset D from a few random episodes.
while not converged **do**
  while update step s $\leq$ collectinginterval **do**
    Draw sequence chunks $\{(o_t, a_t, r_t)_{t=k}^{L+k}\}_{i=1}^{B} \sim \mathcal{D}$ uniformly at random from the dataset
    Compute loss $\mathcal{L}(\theta)$ from Equation 2.4
    Update model parameters $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$
  end while
  $o_1 \leftarrow$ env.reset()
  while timestep $t \leq T$ **do**
    Infer belief over current state $q(s_t|o_{\leq t}, a_{\leq t})$ from the history
    $a_t \leftarrow$ planner$(q(s_t|o_{\leq t}, a_{\leq t}), p)$
    Add exploration noise $\epsilon \sim p(\epsilon)$ to the action
    for actionrepeat $k = 1..R$ **do**
      $r_t^k, o_{t+1}^k \leftarrow$ env.step(a_t)
    end for
    $r_t, o_{t+1} \leftarrow \sum_{k=1}^{R} r_t^k, o_{t+1}^R$
  end while
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^{T}\}$
end while

For our work, we use the policy optimization from Dreamer, while retaining the world model learnt by PlaNet.

Learning Latent Dynamics for Planning from Pixels

PlaNet is a Model based reinforcement learning approach that works to solve continuous control tasks in the DeepMind Control Suite (Tassa et al. (2018)) from visual inputs, making this a partially observable MDP. It learns a world model by encoding visual inputs to the latent space and executing the dynamics with a recurrent unit. The setup is as follows:

$$
\begin{aligned}
&\text{Image encoder:} && h_t = e_\theta(o_t) \\
&\text{Posterior dynamics:} && q_\theta(s_t|s_{t-1}, a_{t-1}, h_t) \\
&\text{Prior dynamics:} && p_\theta(s_t|s_{t-1}, a_{t-1}) \\
&\text{Reward predictor:} && p_\theta(r_t|s_t) \\
&\text{Image decoder:} && p_\theta(o_t|s_t).
\end{aligned} \tag{2.3}
$$

With the learnt world model, PlaNet performs model-predictive control (MPC) (Richards (2005)) using the cross-entropy method (CEM) (Rubinstein (1997); Chua et al. (2018b)) as the planner. The planner samples action sequences at a fixed horizon length and chooses actions such that $R = \sum_{\tau=t+1}^{t+H+1} \mathbb{E}[p(r_\tau|s_\tau)]$ is a maximum.

PlaNet learns the transition model, the observation model and the reward predictor from past data collected online using the online planner. PlaNet also learns an encoder $q(s_t|o_{\leq t}, a_{\leq t})$. With this, along with an online planner, PlaNet adapts its plan based on new observations, by replanning at every step. The data is collected in an online fashion. The different models, namely the transition, observation and the reward model, along with the encoder, are gaussian distributions, parametrized by the mean and variance which are modeled as a deep neural networks, using convolutional and MLP architectures. The training objective is derived using variational inference, with

the evidence lower bound (ELBO) being

$$\begin{aligned}\ln p(o_{1:T}|a_{1:T}) &\triangleq \ln \int \prod_t p(s_t|s_{t-1}, a_{t-1}) p(o_t|s_t) ds_{1:T} \\ &\geq \sum_{t=1}^{T} (\mathbb{E}_{q(s_t|o_{\leq t}, a_{\leq t})}[\ln p(o_t|s_t)] \\ &\quad - \mathbb{E}_{q(s_{t-1}|o_{\leq t-1}, a_{\leq t-1})}[\mathrm{KL}[q(s_t|o_{\leq t}, a_{\leq t})||p(s_t|s_{t-1}, a_{t-1})])\end{aligned} \tag{2.4}$$

To model the latent dynamics, PlaNet uses a Recurrent State-Space Model (RSSM), which contains both stochastic and deterministic components in the recurrence relationship, namely

$$\begin{aligned}&\text{Deterministic State Model:} && h_t = f(h_{t-1}, s_{t-1}, a_{t-1}) \\ &\text{Stochastic State Model:} && s_t \sim p(s_t|h_t) \\ &\text{Observation Model:} && o_t \sim p(o_t|h_t, s_t) \\ &\text{Reward predictor:} && r_t \sim p(r_t|s_t, h_t)\end{aligned} \tag{2.5}$$

This relation is shown in the graphical model in Figure 4. Combining this world model with the CEM planner, PlaNet was able to succesfully model the latent dynamics and solve most of the continuous control tasks in DM Control Suite.

Dream to Control: Learning Behaviors by Latent Imagination

A critical missing piece from PlaNet was the lack of an actual policy network. Using a CEM planner limits the planning horizon, and thus necessarily makes the policy short-sighted, not being able to plan for long time-horizons. Dreamer learns long-horizon behaviors with a policy network that trains completely inside the imagination of the world model, by back-propagating the gradients through the world model, rather than running a policy gradient scheme. The world model learnt by Dreamer is the exact

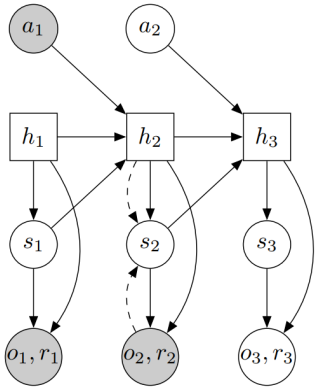


Figure 4: Graphical model of the RSSM in (Hafner et al. (2018)). The model observes the first two time steps and predicts the third. Circles represent stochastic variables, and squares deterministic variables

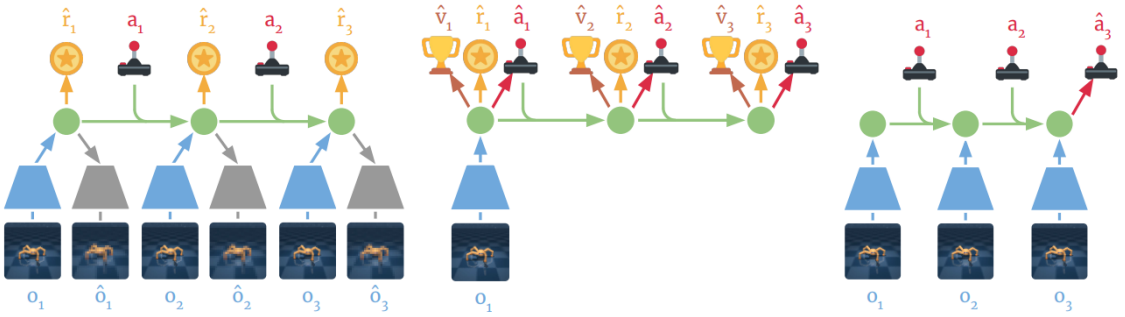


Figure 5: Setup of Dreamer (Hafner et al. (2019)). The world model is learnt from collecting online data from the intermediary policy, and the policy and value estimates are updated inside the imagination of the world model. The policy network then acts on the latent states

same as the world model learnt by PlaNet, except for the fact that the online data collection policy is now the actor network, instead of the CEM planner. To provide a better feedback to the actor network, there's also a value network that is used to compute lambda-returns. This trades off the bias and variance.

The highlight of Dreamer is its ability to efficiently learn a good policy by training only within the world model, and thus not generating any real world samples. This process is outlined in Figure 5.

While the world model setup is the same, Dreamer has the extra actor network $a_\tau \sim q_\phi(a_\tau|s_\tau)$, which is conditioned on the latent states and is parametrized by the network weights ϕ. The policy tries to maximize the expected rewards in the future, where the rewards are sampled from the reward predictor. The value model estimates the expected imagined rewards that the action model achieves from each state $_\tau$, and the state value network is parametrized by weights ψ as $v_\psi(s_\tau) \approx \mathbb{E}_{q(.|s_\tau)}(\sum_{t=\tau}^{t+H} \gamma^{\tau-t} r_\tau)$, where H is the horizon. The values can be estimated in several ways. One can either simply sum up the rewards, thereby not requiring a value network, or using the value network, we can trade-off bias and variance by using exponentially weighted averages, given by the lamda-return. The learning objective for the action model is to essentially maximize the value estimates, and the objective for the value model is to minimize the L_2 error between the calculated estimate and the output of the network. Since all these calculations are through neural-networks, we can update the policy network with backpropagation through the value model, through the imagined states, through the model, which would then depend on the imagined actions. The objective for the actor network is

$$\phi^* = \arg\max_{\phi} \mathbb{E}_{q_\theta, q_\phi}[\sum_{\tau=t}^{t+H} V_\lambda(s_\tau)] \tag{2.6}$$

and the objective for the value network is

$$\psi^* = \arg\max_{\psi} \mathbb{E}_{q_\theta, q_\phi}[\sum_{\tau=t}^{t+H} \frac{1}{2}||V_\lambda(s_\tau) - v_\psi(s_\tau)||^2] \tag{2.7}$$

The world model is learnt with the same loss as was described with PlaNet in Equation 2.4. The algorithm for Dreamer is given in Algorithm 2.

Algorithm 2 Dreamer

initialize: Dataset D from a few random episodes.
while not converged **do**
  while update step s $\leq$ collectinginterval **do**
    \\ `Dynamics Learning`
    Draw sequence chunks $\{(o_t, a_t, r_t)_{t=k}^{L+k}\}_{i=1}^{B} \sim \mathcal{D}$ uniformly at random from the dataset
    Compute loss $\mathcal{L}(\theta)$ from Equation 2.4
    Update model parameters $\theta \leftarrow \theta - \alpha \nabla_\theta \mathcal{L}(\theta)$
    \\ `Behavior Learning`
    Imagine trajectories $\{(s_\tau, a_\tau)\}_{\tau=t}^{t+H}$ from each s_τ
    Predict Rewards $\mathrm{E}(q_\theta(r_\tau|s_\tau))$ and values $v_\psi(s_\tau)$
    Update model parameters $\phi \leftarrow \phi + \alpha \nabla_\phi \mathcal{L}(\phi)$ from Equation 2.6
    Update model parameters $\psi \leftarrow \psi - \alpha \nabla_\psi \mathcal{L}(\psi)$ from Equation 2.7
  end while
  $o_1 \leftarrow$ env.reset()
  while timestep $t \leq T$ **do**
    Compute $s_t \sim p_\theta(s_t|s_{t-1}, a_{t-1}, o_t)$ from history
    Compute $a_t \sim q_\phi(a_t|s_t)$ from the action model
    Add exploration noise $\epsilon \sim p(\epsilon)$ to the action
    $r_t, o_{t+1} \leftarrow$ env.step(a_t)
  end while
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(o_t, a_t, r_t)_{t=1}^{T}\}$
end while

2.2. Exploration and Intrinsic Motivation

Efficient exploration is a crucial component of an effective reinforcement learning agent (Kakade and Langford (2002)). In tabular settings, it is efficiently addressed with exploration bonuses based on state visitation counts (Strehl and Littman (2008); Jaksch et al. (2010)) or fully Bayesian approaches (Duff and Barto (2002); Poupart et al. (2006)), however these approaches are hard to generalize to high-dimensional states, such as images. Recently, several methods were proposed based on generalization of these early approaches, such as using pseudo-count measures of state visitation (Bellemare et al. (2016); Ostrovski et al. (2018)). (Osband et al. (2016)) derived an efficient approximation to the Thompson sampling procedure via ensembles of Q-

functions. (Osband et al. (2018); Lowrey et al. (2018)) use ensembles of Q-functions to track the posterior of the value functions with randomized prior functions. In contrast, our approach neither use reward nor state at training time.

A different line of work on intrinsic motivation considered exploration as an objective on its own (Oudeyer et al. (2007); Oudeyer and Kaplan (2009)). Practical examples of such approaches focus on maximizing prediction error as the intrinsic reinforcement learning objective (Pathak et al. (2017); Burda et al. (2019)). These approaches can also be understood as maximizing the agent's surprise (Schmidhuber (1991a); Achiam and Sastry (2017)). Similar to our work, other recent approaches use the notion of model disagreement to encourage visiting states with the highest potential to improve the model (Burda et al. (2018a); Pathak et al. (2019)), motivated by the active learning literature (Seung et al. (1992); McCallumzy and Nigamy (1998)). However, these approaches are model-free and are very hard to fine-tune to a new task, requiring millions of environment steps for fine-tuning.

Self-Supervised Exploration via Disagreement

Specifically, Pathak et al. (2019) come up with a similar exploration bonus to ours, where the use the disagreement between the predictions of the ensemble to generate intrinsic rewards. Unlike past formulations of the problem, say maximising the visitation count (Poupart et al. (2006); Lopes et al. (2012)) of less frequently visited states, or using the prediction error of the forward dynamics model (Pathak et al. (2017); Schmidhuber (1991a)), this formulation of ensemble disagreement lends itself naturally to attacking the stochasticity in the environment. They train an ensemble of forward dynamics models and incentivize the agent to explore the action space in regions of maximum variance, or maximum disagreement between the models within the ensemble. With this objective, one can effectively show that the agent learns to

avoid stochasticity in the environment. Essentially, as the agent interacts with the environment, it collects trajectories $\{x_t, a_t, x_{t+1}\}$,and the transitions are used to train an ensemble of forward dynamics models $\{f_{\theta_1}, f_{\theta_2}, ..., f_{\theta_k}\}$, which are essentially one step prediction models. The models can be trained in a straightforward manner with the L_2 prediction error, namely $||f(x_t, a_t; \theta) - x_{t+1}||_2^2$ by bootstrapping. States which have not yet been visited would generate a high variance within the ensemble, resulting in a high disagreement about the next state. This metric is used as the intrinsic reward, namely the variance among the next state predictions in the ensemble,

$$r_t^i \triangleq \mathbb{E}_\theta[||f(x_t, a_t; \theta) - \mathbb{E}_\theta[f(x_t, a_t; \theta)]||_2^2] \tag{2.8}$$

Note that the ground truth next state is not really necessary to generate the intrinsic rewards. The paper then shows this method working in Atari games and in robot control tasks, by training the RL agent with PPO (Schulman et al. (2017)). However, while this approach is sound, it lacks several key aspects. Firstly, it still looks only at retrospective novelty and not expected novelty. Secondly, even while estimating the epistemic uncertainty with the disagreement metric, the approach is still model-free, and that still makes it severely lacking in terms of sample efficiency, and task efficiency. The only policy being learnt here is the exploration policy and if one needs to actually get a task policy out of this, then one needs to finetune a task policy, which is definitely not an easy task. The process behind Pathak et al. (2019) is shown in Figure 6

2.3. Active Learning

The idea of actively exploring to collect the most informative data goes back to the formulation of the information gain Lindley (1956). MacKay (1992) described how a learning system might optimize Bayesian objectives for active data selection based on

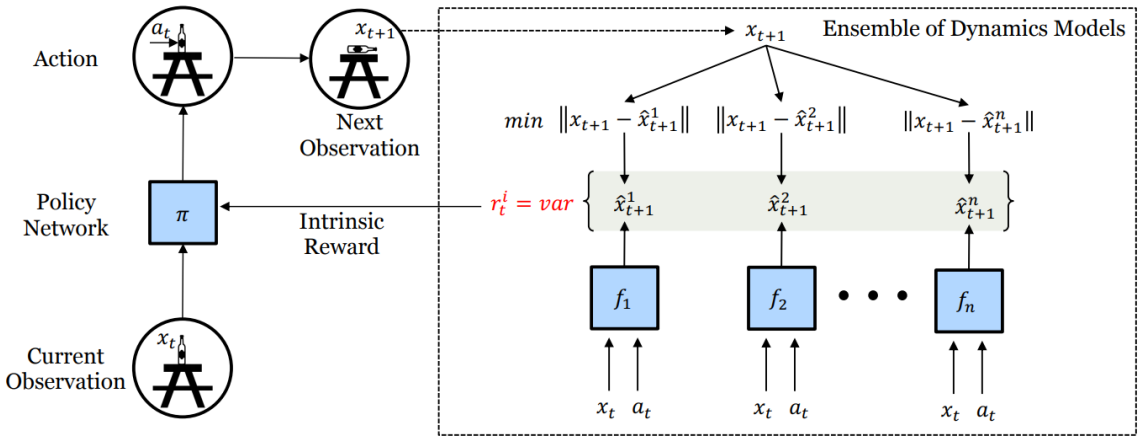


Figure 6: Setup of Pathak et al. (2019)

the information gain. Sun et al. (2011) derived a model-based reinforcement learning agent that can optimize the infinite-horizon information gain and experimented with it in tabular settings. The closest works to ours are Chua et al. (2018b); Shyam et al. (2019), which use a measurement of disagreement or information gain through ensembles of neural networks in order to incentivize exploration. However, these approaches are restricted to setups where low-dimensional states are available, whereas we design a latent state approach that scales to high-dimensional observations. Moreover, we provide a theoretical connection between information gain and model disagreement.

Model Based Active Exploration

The closest work to ours is MAX (Shyam et al. (2019)). MAX also looks into the fact that most exploration methods are reactive and are not active, where the internal model of the agent essentially guides itself to look for places where it isn't confident. To this end, MAX uses an ensemble of dynamics models and uses the disagreement between the next state predictions to come up with a novelty metric. To calculate the disagreement, MAX has a more general formulation by using the Jensen Shannon Divergence (JSD) for discrete environments, and Jesen Renyi Diver-

gence (JRD) for continuous environments. MAX uses this to construct policies that are purely exploratory in nature. If T is the space of all possible transition functions, and if ϕ is one such transition, then the information gain from this transition is $\text{IG}(s, a, s') = \text{IG}(\phi) = D_{\text{KL}}(\mathcal{P}(T|\phi)||\mathcal{P}(T))$. So, for an exploration policy π, the novelty, or the utility from that policy can be given to be

$$\text{IG}(\pi) = \mathbb{E}_{t\sim\mathcal{P}(T)}[\mathbb{E}_{s,a\sim\mathcal{P}(\mathcal{S},\mathcal{A}|\pi,t)}[u(s,a)] \tag{2.9}$$

where

$$u(s,a) = \int_T \int_{\mathcal{S}} \text{IG}(s,a,s')p(s'|a,s,t)p(t)ds'dt = \text{JSD}\{\mathcal{P}(\mathcal{S}|s,a,t)|t \sim \mathcal{P}(T)\} \tag{2.10}$$

where JSD is the Jensen-Shannon Divergence, capturing the disagreement present in a space of distributions. Using this utility, MAX constructs an internal MDP, where an exploration policy tries to maximize the expected utility, thereby, visiting states that are expected to have high novelty. The ensemble is trained with bootstrapping, and to actually be able to calculate the utility tractably with samples, the utility is approximated as

$$u(s,a) \simeq H(\frac{1}{N}\sum_{i=1}^{N}\mathcal{P}(\mathcal{S}|s,a,t_i)) - \frac{1}{N}\sum_{i=1}^{N}H(\mathcal{P}(\mathcal{S}|s,a,t_i)) \tag{2.11}$$

For large continuous spaces, the JSD is replaced by the JRD, with the Renyi entropy, and the renyi entropy has a closed-form solution for a mixture of N Gaussians (Wang

Algorithm 3 MAX

initialize: Dataset D from a few random episodes.
        Ensemble, $T' = \{t_1, t_2, ..., t_N\}$
while exploring **do**
  ExplorationMDP $\leftarrow (\mathcal{S}, \mathcal{A}$,Uniform$\{T'\}, u, \delta(s_\tau))$
  $\pi \leftarrow$ Solve(ExplorationMDP)
  $a_\tau \sim \pi(s_\tau)$
  act in the environment: $s_{\tau+1} \sim \mathcal{P}(\mathcal{S}|s_\tau, a_\tau, t*)$
  $\mathcal{D} \leftarrow \mathcal{D} \cup \{(s_\tau, a_\tau, s_{\tau+1}\}$
  Train t_i on $\mathcal{D}$ for each t_i in T'
end while
return Ensemble T'

et al., 2009), given by

$$\begin{aligned} \text{JRD}\{\mathcal{N}_i(\mu_i, \Sigma_i)|i = 1...N\} &= H_2(\sum_i^N \frac{1}{N}\mathcal{N}_i) - \frac{1}{N}\sum_i^N H_2(\mathcal{N}_i) \\ &= -\ln[\frac{1}{N^2}\sum_{i,j}^N \mathcal{D}(\mathcal{N}_i, \mathcal{N}_j)] - \frac{1}{N}\sum_i^N \frac{\ln|\Sigma_i|}{2} - d\ln(2)/2 \end{aligned} \tag{2.12}$$

where $\mathcal{D}(\mathcal{N}_i, \mathcal{N}_j) = \frac{1}{|\Omega|^1}\exp(-\frac{1}{2}\Delta^T\Omega^{-1}\Delta)$ with $\Omega = \Sigma_i + \Sigma_j$ and $\Delta = \mu_j - \mu_i$. The Equation 2.12 measures the divergence among predictions of models, and is used as the utility for practical purposes in continuous state spaces. The algorithm behind max is given in Algorithm 3

While MAX has a much more general formulation of the disagreement objective than ours, MAX still works on state spaces, and does not scale up to high dimensional observations like pixels. Furthermore, MAX still uses a model-free approach to train its exploration policy, namely SAC (Haarnoja et al. (2018)). On the other hand, we scale up to high dimensional observations, without any state information, and we train our policies by backpropagating through the learned model, resulting in a more

efficient optimization. MAX also requires larger computational resources to execute, as the ensemble is made up of the full dynamics model, whereas we have light-weight, one step prediction models for our ensemble, and use a single larger world model to train the exploration policy in imagination.

3. Planning to Explore

We proceed to discuss our contribution. We present a method to efficiently explore given high dimensional pixel observations in a task-agnostic manner by using the disagreement within an ensemble in the latent space. We build exploration policies that collect data based on expected novelty, irrespective of the task at hand. We show that this method builds much more generalizable world models that can transfer well to tasks at test time with very few supervised episodes, even compared to a supervised Oracle, which we take to be Dreamer (Hafner et al. (2019)).

3.1. Control with Latent Dynamics

World models summarize past experience into a representation of the environment that enables predicting imagined future sequences (Sutton, 1991; Watter et al., 2015; Ha and Schmidhuber, 2018). When sensory inputs are high-dimensional observations, predicting compact latent states s_t lets us efficiently predict many future sequences in parallel. The model states s_t are not to be confused with the unknown true environment states. Specificially, we use the latent dynamics model of PlaNet (Hafner et al., 2018), that consists of the following key components that are illustrated in 7,

$$\begin{aligned}
&\text{Image encoder:} && h_t = e_\theta(o_t) \\
&\text{Posterior dynamics:} && {}_\theta(s_t|s_{t-1}, a_{t-1}, h_t) \\
&\text{Prior dynamics:} && p_\theta(s_t|s_{t-1}, a_{t-1}) \\
&\text{Reward predictor:} && p_\theta(r_t|s_t) \\
&\text{Image decoder:} && p_\theta(o_t|s_t).
\end{aligned} \tag{3.1}$$

The image encoder is implemented as a CNN, and the posterior and prior dynamics share an RSSM. The temporal prior predicts forward without access to the cor-

responding image. The reward predictor and image decoder provide a rich learning signal to the dynamics. The distributions are parameterized as diagonal Gaussians. All model components are trained jointly similar to a variational autoencoder (VAE) (Kingma and Welling, 2013; Rezende et al., 2014) by maximizing the evidence lower bound (ELBO).

$$\max_{\theta} \mathbb{E}_{q_\theta(s_{1:T}|o_{1:T},a_{1:T})} \sum_{t=1}^{L} \Big(\ln p_\theta(r_t|s_t) + \ln p_\theta(o_t|s_t) \\ -\mathrm{KL}(q_\theta(s_t|s_{t-1},a_{t-1},o_t)||p_\theta(s_t|s_{t-1},a_{t-1})) \Big) \tag{3.2}$$

Given this learned world model, we need to derive behaviors from it. Instead of online planning, we use Dreamer (Hafner et al., 2019) to efficiently learn a parameteric policy inside the world model that considers long-term rewards. Specifically, we learn two neural networks that operate on latent states of the model. The state-value estimates the sum of future rewards and the actor tries to maximize these predicted values,

$$\text{Actor:} \quad \pi_\phi(a_t|s_t) \qquad \text{Value:} \quad V_\psi(s_t). \tag{3.3}$$

The actor and value learn from the same forward predictions by the learned world model. These are imagined using the transition model and the actor. They start from latent states obtained by encoding images from the replay buffer. The actor efficiently maximizes the predicted values by propagating their gradients through the neural network dynamics model into the actor.

$$\max_{\pi} \mathrm{E}_{p,\pi} \sum_{t=1}^{H} \gamma^t V(s_t) \tag{3.4}$$

$$\min_{V} \mathrm{E}_{p,\pi} \sum_{t=1}^{H} \frac{1}{2} \Big(V(s_t) - \big(r_t + \gamma V(s_{t+1}) \big) \Big)^2 \tag{3.5}$$

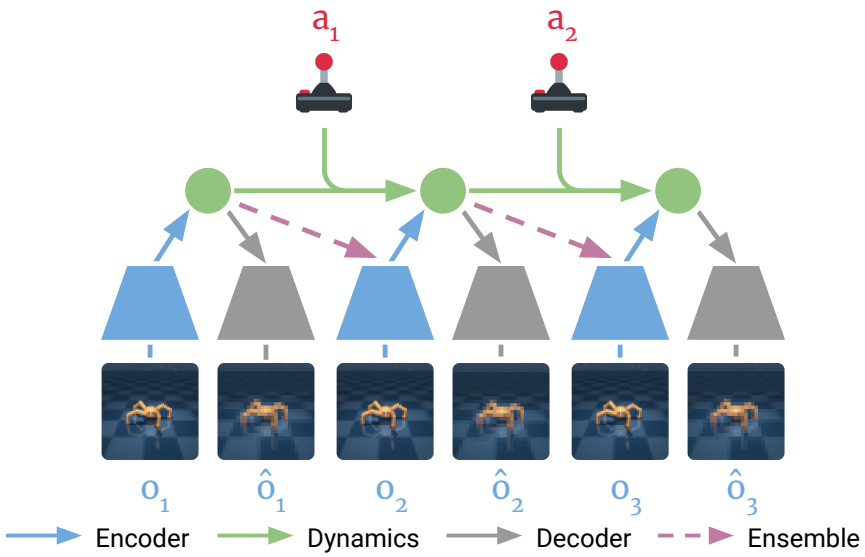


Figure 7: The Latent Dynamics Model. The latent dynamics model first encodes the input images. It then generates a sequence of compact model states conditioned on the image embeddings and actions. The model is learned by reconstructing the images from the model states. Additionally, we train an ensemble that predicts the next image embedding given a model state.

Both actor and value are efficiently optimized using gradients. The actor is updated by propagating the value gradient back through the imagined latent transitions using stochastic backpropagation. The world model is kept fixed while optimizing the actor and value. When collecting experience, the actor is applied to the current latent state of the episode to generate actions.

3.2. P2E

We consider a learning setup with two phases, as illustrated in fig:setting. During task-agnostic exploration, the agent gathers information about the environment and summarizes this past experience in the form of a parametric world model. After exploration, the agent is given a downstream task in the form of a reward function that it should adapt to with no or limited additional environment interaction.

We approach this setup by learning a global world model during exploration, as shown

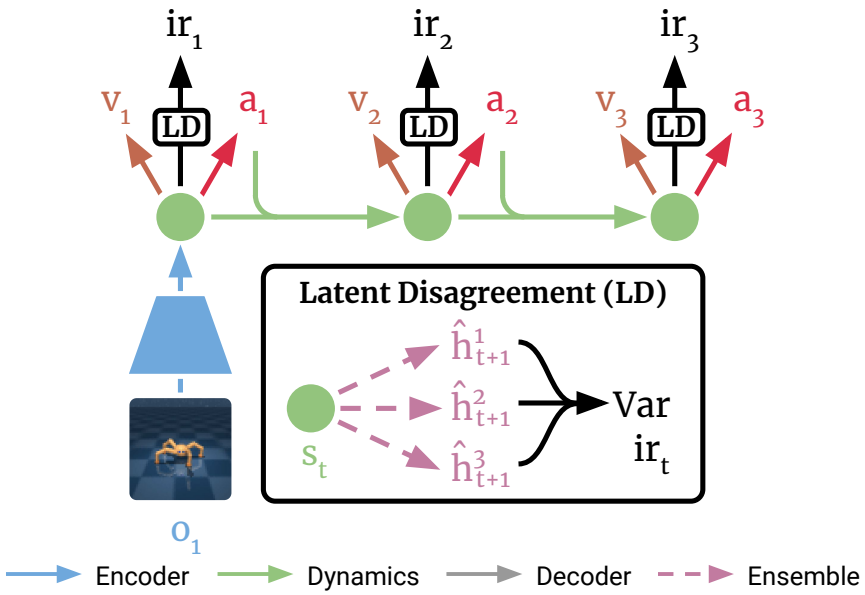


Figure 8: Getting the intrinsic reward: given the world model and ensemble, we learn long-term behaviors by latent imagination. The intrinsic reward for learning the actor and value models is the variance of the ensemble at each time step.

in alg:exploration. This is achieved by training an exploration policy inside of the world model to seek out novel states as estimated by ensemble disagreement in latent space. Crucially, this intrinsic reward can be evaluated without generating images, which allows efficient massive parallel learning to quickly to react changes in the non-stationary intrinsic reward.

During adaptation, the we can efficiently optimize a task policy by imagination inside of the world model, as shown in alg:adaptation. Since this model is trained without being biased toward a specific task, it can only be learned once and then used to adapt to multiple downstream tasks.

3.3. Latent Disagreement

To efficiently learn a world model of an unknown environment, we must select trajectories in a directed manner. A successful strategy should explore the environment such as to collect new experience that improves the model the most. For this, we

Algorithm 4 Planning to Explore via Latent Disagreement

initialize: Dataset D from a few random episodes.
        World model M.
        Latent disagreement ensemble E.
        Exploration actor-critic π_{LD}.
while exploring **do**
  Train M on D.
  Train E on D.
  Train π_{LD} on LD reward in imagination of M.
  Execute π_{LD} in the environment to expand D.
end while
return Task-agnostic D and M

Algorithm 5 Zero and Few-Shot Task Adaptation

input: World model M.
        Dataset D without rewards.
        Reward function R.
initialize: Latent-space reward predictor $\hat{\text{R}}$.
        Task actor-critic π_{R}.
while adapting **do**
  Distill R into $\hat{\text{R}}$ for sequences in D.
  Train π_{R} on $\hat{\text{R}}$ in imagination of M.
  Execute π_{R} for the task and report performance.
  Optionally, add task-specific episode to D and repeat.
end while
return Task actor-critic π_{R}.

quantify the model's uncertainty about its predictions for different latent states. An exploration policy then seeks out states with high uncertainty. Once executed in the environment, the model is trained on the newly acquired trajectories and reduces its uncertainty in these and the process is repeated.

Quantifying uncertainty is a long standing open challenge in deep learning (MacKay, 1992; Gal, 2016). In this paper, we use ensemble disagreement as one of the empirically successful methods for quantifying uncertainty Lakshminarayanan et al. (2017); Osband et al. (2018). As shown in fig:method, we train an ensemble to predict, from

each model state, the next encoder features. The variance of the ensemble serves as the estimate of uncertainty.

Intuitively, because the ensemble models have different initialization and observe data in a different order, their predictions differ for unseen inputs. Once the data is added to the training set, however, the models will converge towards more similar predictions and the disagreement decreases. Eventually, once the whole environment is explored, the models should converge to identical predictions.

Formally, we define an ensemble of one-step predictive models with parameters $\{w_k \mid k \in [1; K]\}$. Each of these models takes a model state s_t and action a_t as input and predicts the next image embedding h_{t+1}. The models are trained with the mean squared error, which is equivalent to Gaussian log-likelihood,

$$\begin{aligned} &\text{Ensemble predictors:} \quad q(h_{t+1}|w_k, s_t, a_t) \\ &q(h_{t+1}|w_k, s_t, a_t) \triangleq \mathcal{N}(\mu(w_k, s_t, a_t), 1). \end{aligned} \tag{3.6}$$

We quantify model uncertainty as the variance over predicted means of the different ensemble members and use this disagreement as the intrinsic reward $\text{ir}_t \triangleq \text{D}(s_t, a_t)$ to train the exploration policy,

$$\begin{aligned} \text{D}(s_t, a_t) &\triangleq \text{Var}\left(\{\mu(w_k, s_t, a_t) \mid k \in [1; K]\}\right) \\ &= \frac{1}{K-1} \sum_k \left(\mu(w_k, s_t, a_t) - \mu'\right)^2, \\ \mu' &\triangleq \frac{1}{K} \sum_k \mu(w_k, s_t, a_t). \end{aligned} \tag{3.7}$$

The intrinsic reward is non-stationary because the world model and the ensemble predictors change throughout exploration. Indeed, once certain states are visited by

the agent and the model gets trained on them, these states will become less interesting for the agent and the intrinsic reward for visiting them will decrease.

We learn the exploration policy using the Dreamer algorithm (3.1). Since the intrinsic reward can be computed in the compact representation space of the latent dynamics model, we can optimize the learned actor and value from imagined latent trajectories without generating images. This lets us efficiently optimize the intrinsic reward of the current model and ensemble without additional environment interaction.

3.4. Expected Information Gain

Latent disagreement has an information-theoretic interpretation. This subsection derives our method from the amount of information gained by interacting with the environment, which has its roots in optimal Bayesian experiment design (Lindley, 1956; MacKay, 1992).

Because the true dynamics are unknown, the agent treats the optimal dynamics parameters as a random variable W. To explore the environment as efficiently as possible, the agent should seek out future states that are informative of our belief over the parameters.

Mutual information formalized the amount of bits that a future trajectory provides about the optimal model parameters on average. We aim to find a policy that shapes the distribution over future states to maximize the mutual information between the image embeddings $H_{1:T}$ and parameters W,

$$I(H_{t+1}; W | s_t, a_t) \tag{3.8}$$

We operate on image embeddings rather then images themselves to save computa-

tion. To select the most promising data during exploration, the agent maximizes the expected information gain,

$$a_t^* \triangleq \arg\max_{a_t} I(H_{t+1}; W | s_t, a_t). \tag{3.9}$$

This expected information gain can be rewritten as conditional entropy of trajectories subtracted from marginal entropy of trajectories, which correspond to, respectively, the aleatoric and the total uncertainty of the model,

$$\begin{aligned} &I(H_{t+1}; W | s_t, a_t) \\ &\quad = \mathrm{H}(H_{t+1} | s_t, a_t) - \mathrm{H}(H_{t+1} | W, s_t, a_t). \end{aligned} \tag{3.10}$$

We see that the information gain corresponds to the epistemic uncertainty, i.e. the reducible uncertainty of the model that is left after subtracting the expected amount of data noise from the total uncertainty.

Trained via squared error, our ensemble members are conditional Gaussians with means produced by neural networks and fixed variances. The ensemble can be seen as a mixture distribution of parameter point masses,

$$\begin{aligned} p(w) &\triangleq \frac{1}{K} \sum_k \delta(w - w_k) \\ p(h_{t+1} | w_k, s_t, a_t) &\triangleq \mathcal{N}(h_{t+1} | \mu(w_k, s_t, a_t), \sigma^2). \end{aligned} \tag{3.11}$$

Because the variance is fixed, the conditional entropy does not depend on the state

or action in our case (D is the dimensionality of the predicted embedding),

$$\begin{aligned} \mathrm{H}(H_{t+1}|W, s_t, a_t) &= \frac{1}{K}\sum_k \mathrm{H}(H_{t+1}|w_k, s_t, a_t) \\ &= \frac{D}{K}\sum_k \ln \sigma_k(s_t, a_t) + \text{const.} \end{aligned} \tag{3.12}$$

Maximizing information gain then means to simply maximize the marginal entropy of the ensemble prediction. For this, we make the following observation: the marginal entropy is maximized when the ensemble means are far apart (disagreement) so the modes overlap the least, maximally spreading out probability mass. While the marginal entropy has no closed-form expression suitable for optimization, one heuristic to measure how far the ensemble means are apart is the empirical variance over ensemble means,

$$\begin{aligned} \mathrm{D}(s_t, a_t) &\triangleq \frac{1}{K-1}\sum_k \big(\mu(w_k, s_t, a_t) - \mu'\big)^2, \\ \mu' &\triangleq \frac{1}{K}\sum_k \mu(w_k, s_t, a_t). \end{aligned} \tag{3.13}$$

To summarize, our exploration objective defined in 3.3, which maximizes the variance of ensemble means, approximates the information gain and thus should find trajectories that will efficiently reduce the model uncertainty.

4. Experiments and Results

Our experiments focus on evaluating whether our proposed Plan2Explore agent is able to fulfill the original motivation: how to efficiently explore and build a model of the world that allows it to quickly adapt and solve tasks in zero or few-shot manner. The rest of the subsections are organized in terms of the key scientific question we would like to investigate as discussed in the introduction.

4.1. Experimental Setup

We use the DM Control Suite (Tassa et al. (2018)) tasks, a standard benchmark for continuous control. All experiments are performed with only visual observations. We use RGB visual observations with 64×64 resolution. We have selected a diverse set of 8 tasks, that feature sparse rewards, high dimensional action spaces, and environments with unstable equilibria and environments that require a long planning horizon. We use episode length of 1000 steps and a fixed action repeat of $R = 2$ for all the tasks.

Specifically, we will be working with visual inputs in continuous control tasks from the environments in DeepMind Control Suite (Tassa et al. (2018)) to demonstrate viability of our methods scaling up to high-dimensional inputs. The visual input sizes that we handle are $64 \times 64 \times 3$. We have selected a diverse set of 8 tasks which individ-

(a) Cup (b) Acrobot (c) Hopper (d) Walker (e) Quadruped

Figure 9: A subset of the environments in DeepMind Control Suite (Tassa et al. (2018)), some of which we use for our experiments

ually have different challenges associated with them. Reacher and Cartpole Swingup Sparse have sparse rewards, while Cheetah and Walker have higher dimensional state and action spaces. Hopper and all the Swingup tasks have unstable equilibria, with Swingup tasks also requiring a longer planning horizon. The tasks have action spaces between -1 and 1, with episode length of 1000 steps. We use a fixed action repeat parameter of $R = 2$ for all the tasks. In addition to these task, we desgined new tasks for the Cheetah environment. The new tasks are Cheetah Run Backwards, which tries to maximize the backward moving velocity, and Cheetah Flipping Forward and Backward tasks, which try to maximize the angular velocity about the torso of Cheetah in the corresponding directions.

Reward details: To test the generalization performance of the our agent, we define three new tasks in the Cheetah environment: Cheetah Run Backwards, Cheetah Flip Forward, Cheetah Flip Backward.

The Cheetah Run Backwards task is defined analogously to Cheetah Run as follows: The reward r is linearly proportional to the backwards velocity v_b up to a maximum of 10m/s, which means $r(v_b) = \max(0, \min(v_b/10, 1))$, where $v_b = -v$ and v is the forward velocity of the Cheetah.

The Cheetah Flip Backward task is defined as follows: The reward r is linearly proportional to the backwards angular velocity ω_b up to a maximum of 5rad/s, which means $r(\omega_b) = \max(0, \min(\omega_b/5, 1))$, where $\omega_b = -\omega$ and ω is the angular velocity about the positive Z-axis, as defined in DeepMind Control Suite.

The Cheetah Flip Forward task is defined as follows: The reward r is linearly proportional to the forward angular velocity ω up to a maximum of 5rad/s, which means $r(\omega) = \max(0, \min(\omega/5, 1))$.

Implementation We use Hafner et al. (2019) with the original hyperparameters unless specified otherwise to optimize both exploration and task polices of P2E. We found that additional capacity provided by increasing the hidden size of the GRU in the latent dynamics model to 400 and the deterministic and stochastic components of the latent space to 60 helped performance. For a fair comparison, we maintain this model size for Dreamer and other baselines. For latent disagreement, we use an ensemble of 5 one-step prediction models implemented as 2 hidden-layer MLP. Full details are in the supplementary material.

Baselines We compare our agent to a state-of-the-art task-oriented agent that receives reward information throughout training, Dreamer Hafner et al. (2019). We also compare to two unsupervised agents, a random data collection policy that uniformly samples from the action space of environment, and model-based curiosity, a version of our method that uses the the model prediction loss as intrinsic reward, as proposed in ICM Pathak et al. (2017). While ICM uses the L_2 error of predicting encoded features, we use the image-specific full model loss of the encoder, decoder and the RSSM as a metric for intrinsic rewards. Note that this reward can only be computed when ground truth data is available, and needs a separate reward predictor to optimize it in a model-based fashion. All compared methods share the same model hyperparameters.

In addition to these model-based agents, we also use the model-free agents from Pathak et al. (2019) and Pathak et al. (2017). While adapting these methods to a new task would be very data-inefficient, we instead use the exploration data produced by these methods to train a Dreamer agent. We then test the zero or few shot performance of this Dreamer agent on a new task.

Number of Runs for Error Bars: First, we discuss about the error bars in the plots and the number of seeds being used. We run every experiment with three different random seeds. The shaded area of the graphs shows the standard deviation in performance. Second, all the plots in the main paper, and in the supplementary, are smoothed with a rolling mean that takes into account the past 8 data points. This was done so as to provide cleaner looking plots that indicate the general trend. Low variance in all the curves consistently across all figures suggests that our approach is very reproducible.

4.2. Zero-shot Transfer

How good is zero-shot transfer to new tasks?

To test whether P2E has learned a global model of the environment that can be used to solve new tasks, we evaluate zero-shot performance of our agent. Our agent learns a model without using any task-specific information. After that, a separate task agent is trained, which optimizes the task reward using only the model learned in an unsupervised way and no new data. To specify the task, we provide the agent with the reward function that is used to label its replay buffer with rewards. This process is described in the Algorithm 5, with the step 10 omitted.

In 10 we plot the zero-shot performance of our task agent with respect to different amounts of exploration data by training the task agent continuously. The results indicate that the zero-shot performance of P2E is competitive to Dreamer, even outperforming it in the hopper hop task. Moreover, P2E overall performs better than other exploration strategies, sometimes being the only successful unsupervised method.

We see that P2E was able to successfully learn a task-agnostic model of the environment, as well as to efficiently derive task-oriented behavior from this model. We

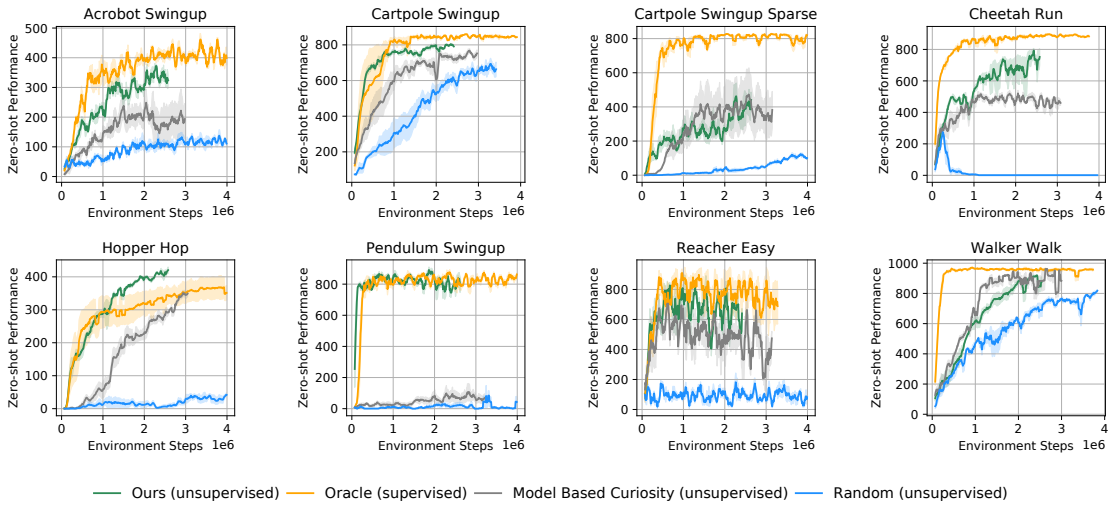


Figure 10: Zero-shot performance from raw pixels without state-space input. After training the agent without reward supervision, we provide it with a task by specifying the reward function. Throughout the exploration, we take snapshots of the agent to train a task policy on the final task and plot its zero-shot performance. Model-based curiosity is a version of our approach trained with the ICM objective Pathak et al. (2017). We see that P2E achieves state-of-the-art zero-shot task performance on a range of tasks, and even demonstrates competitive performance to Dreamer Hafner et al. (2019), a state-of-the-art supervised reinforcement learning agent, on certain tasks. This indicates that P2E is able to explore effectively and learn a global model of the environment that is useful for adapting to new tasks, demonstrating the potential of the task-agnostic reinforcement learning framework. More results and videos are in the supplementary material and the website.

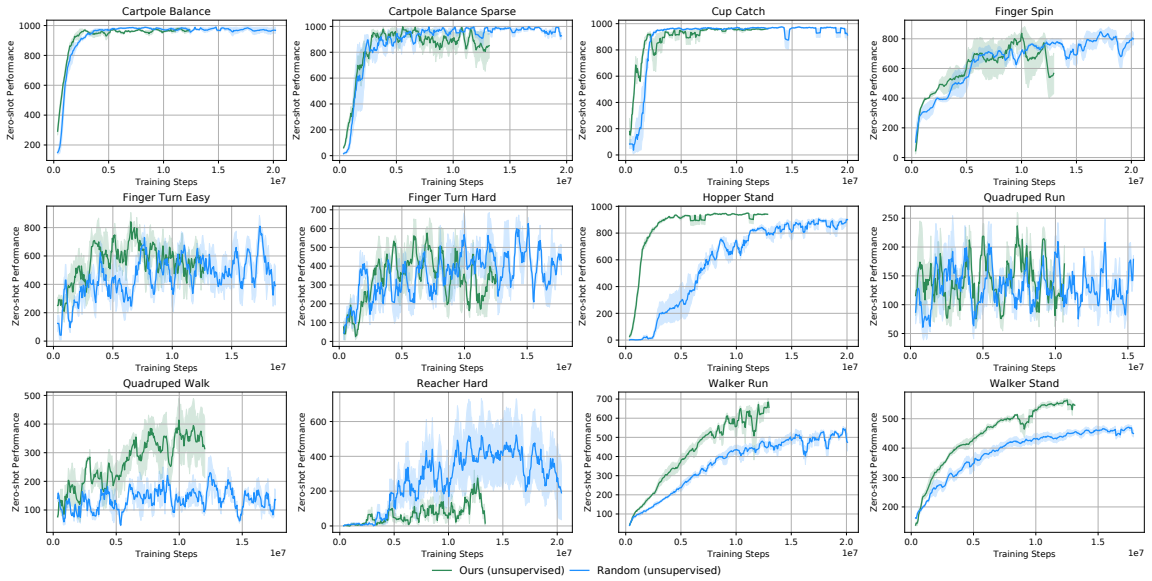


Figure 11: We test the zero-shot performance of our method on an extended set of tasks in the DM Control Suite. We compare our performance to that of the random baseline. It is evident that our method is robust to hyper-parameter changes

emphasize that P2E performs completely unsupervised exploration, and Dreamer has a significant advantage as it is allowed to collect task-oriented data in our experimental setup. Yet, P2E is able to solve all presented tasks, often performing on par or outperforming the supervised agent, Dreamer. Moreover, the latent disagreement reward outperforms other proposed rewards such as model error Pathak et al. (2017) as well as random exploration.

We have run more experiments on other tasks in DM Control Suite, represented in Figure 11. We compare our approach to that of the random baseline. Except for Reacher Hard, our approach always either matches the random baseline outperforms it by a large margin. Noting that our approach maintains the same set of hyper-parameters across the different tasks, it is evident that our approach is robust.

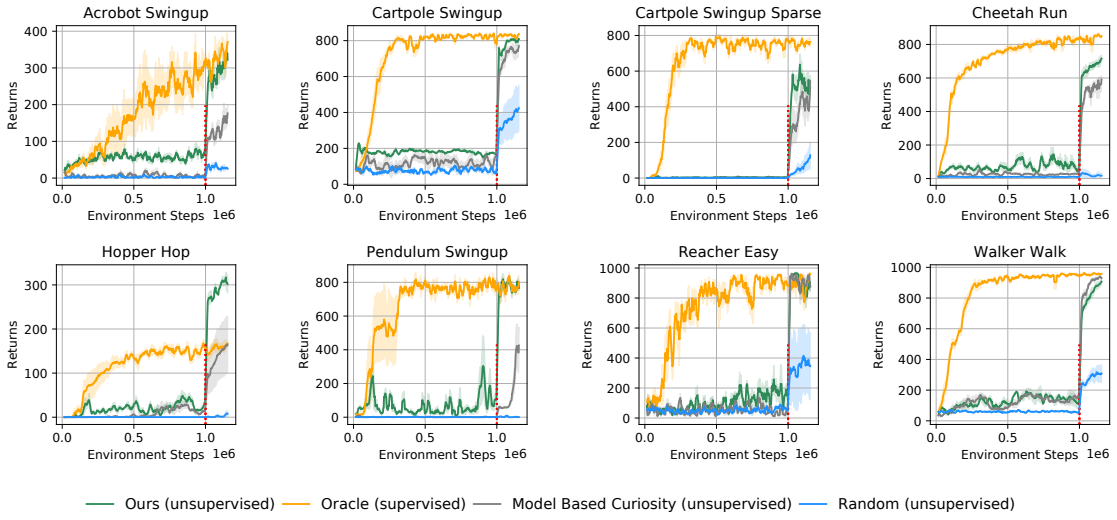


Figure 12: Performance on few-shot adaptation from raw pixels without state-space input. After the exploration phase, during which the agent does not observe the reward and thus does not solve the task, we let the agent collect a small amount of data from the environment. Model-based curiosity is a version of our approach trained with the ICM objective Pathak et al. (2017). We see that P2E is able to explore the environment efficiently in only 1000 episodes, and then adapt its behaviour immediately after observing the reward. P2E adapts rapidly, producing effective behavior competitive to state-of-the-art supervised reinforcement learning in just a few collected episodes.

4.3. Few-Shot Adaptation

How many supervised samples are needed forfinetuning to match the oracle?

While zero-shot learning might suffice for some tasks, in general we will want to adapt our model of the world to task-specific information. In this section, we test whether few-shot adaptation of the model to a particular task is competitive to training a fully supervised task-specific model. To adapt our model, we only add $100 - 150$ supervised episodes which falls under 'few-shot' adaptation. Furthermore, in this setup, to evaluate the data efficiency of P2E we set the number of exploratory episodes to only 1000. In order to efficiently adapt in a few-shot manner, we increase the number of model training steps before collecting new episode by 10 with respect to the default parameter. To provide a fair comparison, we similarly increase the number of training steps for Dreamer by 10 throughout the entire training procedure.

In the exploration phase of Figure 12, i.e., left of the vertical line, our agent does not aim to solve the task, as it is still unknown, however we expect that during some period of exploration it will *coincidentally* achieve higher rewards as it explores the parts of the state space relevant for the task. The performance of unsupervised methods is coincidental until 1000 episodes and then it switches to task-oriented behaviour for remaining 150 episodes, while for supervised, it is task-oriented throughout. That's why we see a big jump to right of vertical line for unsupervised methods. In the few-shot learning setting, P2E eventually performs competitively to Dreamer on all tasks, significantly outperforming it on the hopper task. P2E is also able to adapt quicker or similar to other unsupervised agents on all tasks. These results show that a task-agnostic agent, when presented with a task specification, should be able to rapidly adapt its model to the task information, matching or outperforming the fully

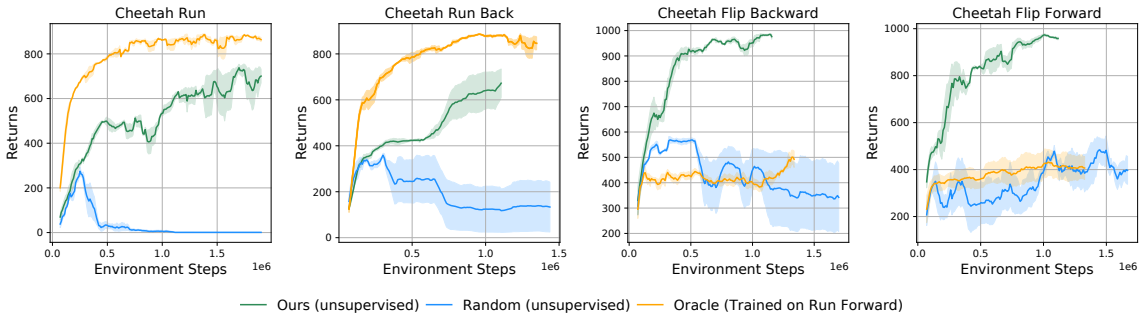


Figure 13: Do task-specific models generalize? We test P2E on zero-shot performance on four different tasks in the cheetah environment from raw pixels without state-space input. Throughout the exploration, we take snapshots of the agent to train a task policy on the four tasks and plot its zero-shot performance. In addition to random exploration, we compare to an oracle agent, Dreamer, that uses the data collected when trained on the run forward task in a supervised way. Although Dreamer trained on 'run forward' is able to solve both running tasks, it struggles on both flipping tasks, indicating that it has not learned a complete model of the environment.

supervised agent trained only for that task. Moreover, P2E is able to learn this general model with a small amount of samples, matching Dreamer, which is fully task-specific, in data efficiency.

4.4. Multi-task Generalization

Do task-agnostic models generalize better than supervised task-specific models?

If the quality of our learned model is good, it should be transferable to multiple tasks. In this section, we test the quality of the learned model on generalization to multiple tasks in the same environment. We devise a set of three new tasks for the Cheetah environment, specifically, running backwards, flipping forwards, and flipping backwards. We evaluate the zero-shot performance of P2E, and additionally compare to a Dreamer agent that is only allowed to collect data on the running forward task and then tested on zero-shot performance on the three other tasks.

13 shows that while Dreamer performs well on both running tasks, it fails to solve the flipping tasks, performing comparably to random exploration. In contrast, P2E performs well across all tasks, outperforming Dreamer on the flipping tasks. This indicates that the model learned by P2E is indeed global, while the model learned by Dreamer, which is task-oriented, fails to generalize to significantly different tasks.

4.5. Expected vs Retrospective Novelty

Does planning for expected novelty give rise to better exploration than retrospective novelty?

We evaluate whether maximizing expected novelty is advantageous by comparing an approach that maximizes expected novelty by estimating it with model-based rollouts to a model-free approach that uses the likelihood ratio gradient estimator. Our Plan2Explore agent is able to measure expected novelty by imagining future states that were not visited yet. A model-free agent, in contrast, is only trained on the states from the replay buffer, and only gets to see the novelty in retrospect, after the state has been visited. Here, we evaluate the advantages of computing expected versus retrospective novelty by comparing P2E to a corresponding model-free agent Pathak et al. (2019). 14 shows the zero-shot performance of the expected and retrospective computation strategies. Our agent achieves superior performance, indicating that planning to explore is advantageous in our case.

To further support our hypothesis, we also show results of retrospective and expected comparisons with prediction error objective in Figure 15. These results are consistently show that expected is significantly better than retrospective exploration and furthermore reinforce our hypothesis.

The case for training an exploration policy that tries to optimize for expected novelty

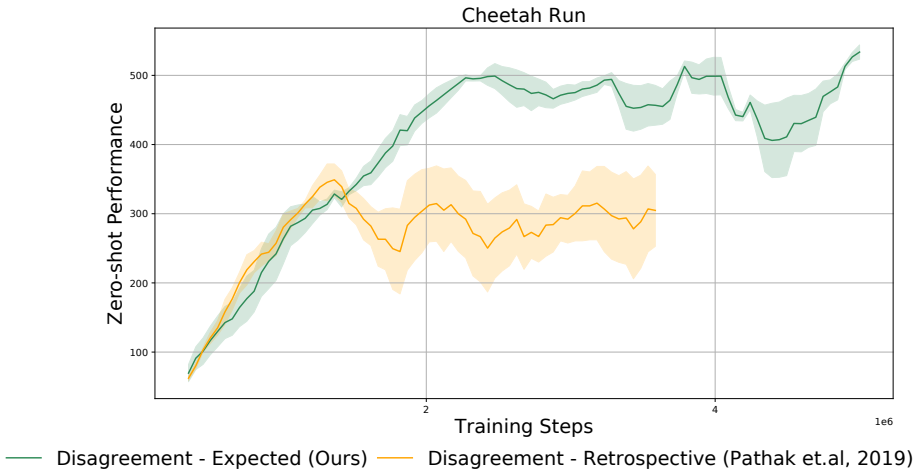


Figure 14: We compare P2E to a corresponding model-free agent Pathak et al. (2019) that uses the disagreement objective. The model-free agent is only trained on already visited states and thus optimizes the retrospective novelty. We adapt Pathak et al. (2019) to zero-shot setting by training a model-based agent on the exploration data collected by Pathak et al. (2019). We see that our agent that maximizes expected novelty achieves superior performance.

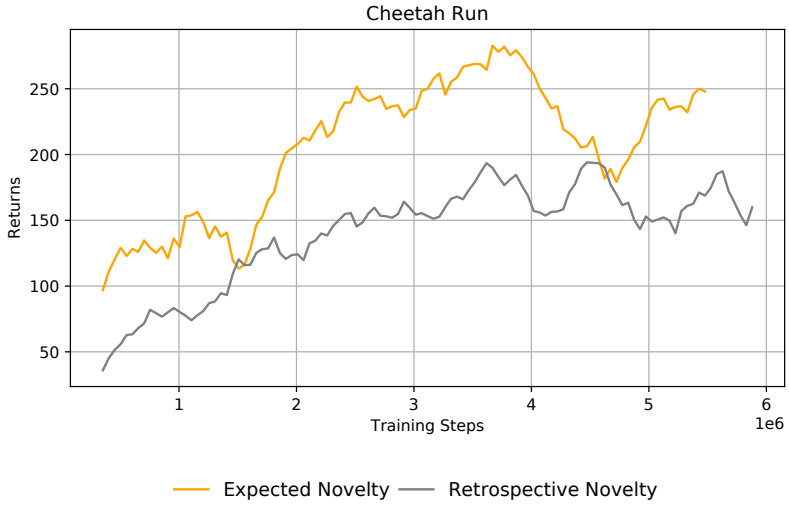


Figure 15: We compare a model-based agent that plans to explore to a model-free agent that optimizes the retrospective novelty, namely the prediction error objective. We evaluate the zero-shot performance of the retrospective approach by training a model-based agent on the exploration data collected by the retrospective approach. We see that the agent that maximizes expected novelty achieves superior performance. This was run with a single seed only.

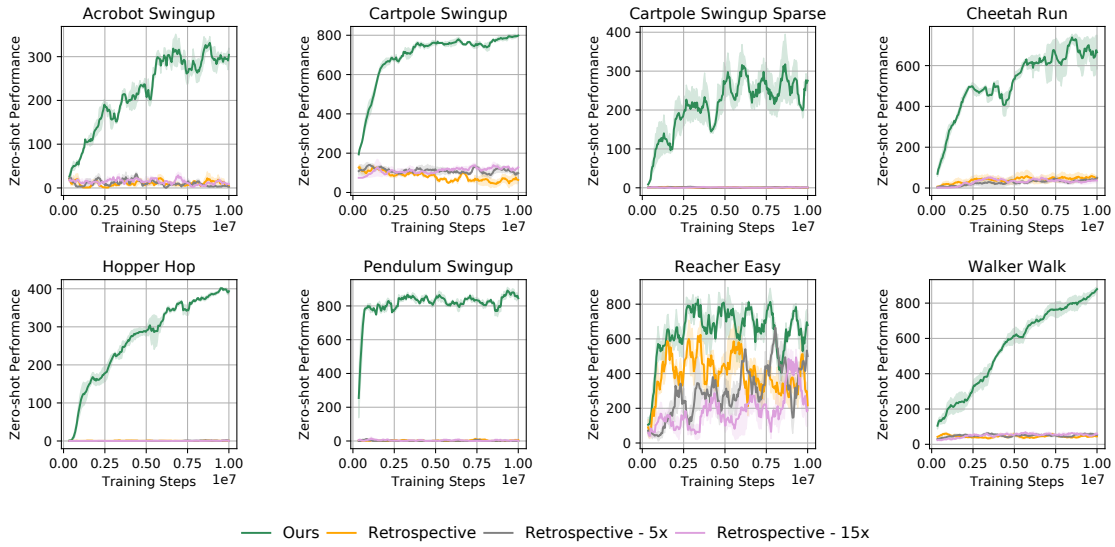


Figure 16: We compare the zero-shot performance of our method to a variant that has a small imagination of horizon of 1, effectively making it a 'reactive' exploration method. This simulates the performance difference between optimizing for expected novelty vs retrospective novelty. As is clearly observed, optimizing for expected novelty vastly outperforms optimizing for retrospective novelty.

can further be made by playing around with the imagination horizon in training the exploration policy. The relevant experiment is represented in Figure 16. Here, we compare our method, which has an imagination horizon of 15, to a variant which has a horizon of 1, effectively not being able to plan for future novelty, therefore simulating an approach that only optimizes for retrospective novelty in a one step rollout. For the retrospective approaches, we have three ablations, where the experiment labeled with 5x is one where we train the exploration policy on old data 5 times that of the world model, and the experiment labeled with 15x is one where we train the exploration policy on old data 15 times that of the world model. This is done so as to give a stronger learning signal to the policy. But as is clearly observable, these short horizon approaches vastly underperform. This once again supports the case for optimizing for expected novelty.

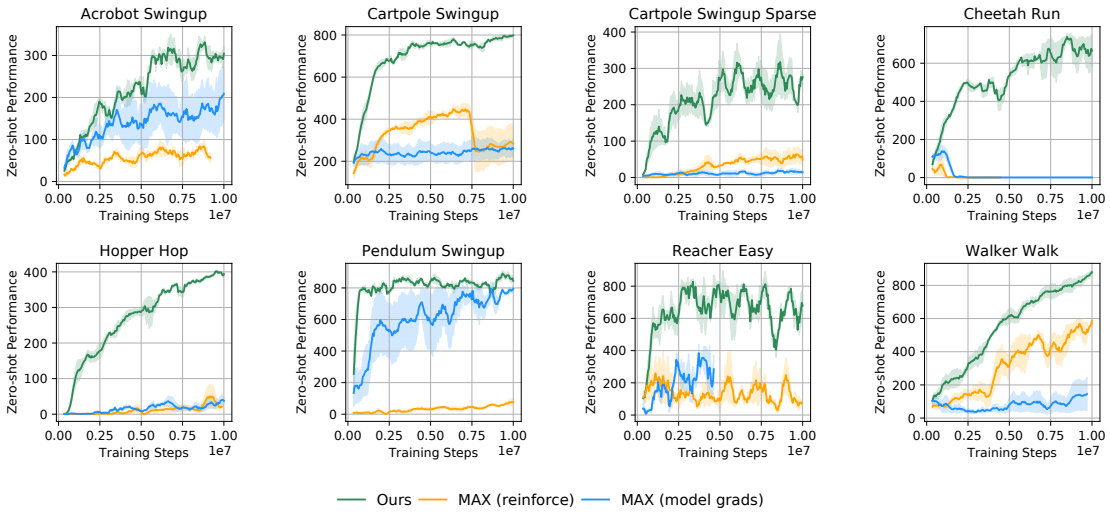


Figure 17: We look at the zero-shot performance of our agent, as compared to the MAX Baselines. We evaluate it against two different variants of MAX, where we train the exploration policy by propagating gradients through the learnt world model, and where we train it in a model-free way by using policy gradients. Our method outperforms both the variants of MAX

4.6. Comparing with MAX

We specifically compare the zero-shot performance of our method as compared to MAX (Shyam et al 2019). This is shown in Figure 17 We first see the performance difference when we change the generation of intrinsic rewards from Latent Disagreement to the JRD metric as designed by MAX, given in Equation 2.12. However, the key differences between MAX and ours is that instead of using the full dynamics model elements in the ensemble, which for us would be too expensive, we use the one step prediction models as the ensemble, where we learn both the mean and variance of the individual Gaussians distributions, as designed by MAX. We look into two different variants of MAX. First, we train the exploration policy just like how we train our exploration policy, by propagating gradients through the dynamics model. This is labeled as MAX (model grads) in the plot. This is not how MAX trains its policies, which uses a model-free approach using policy gradients. To simulate this as

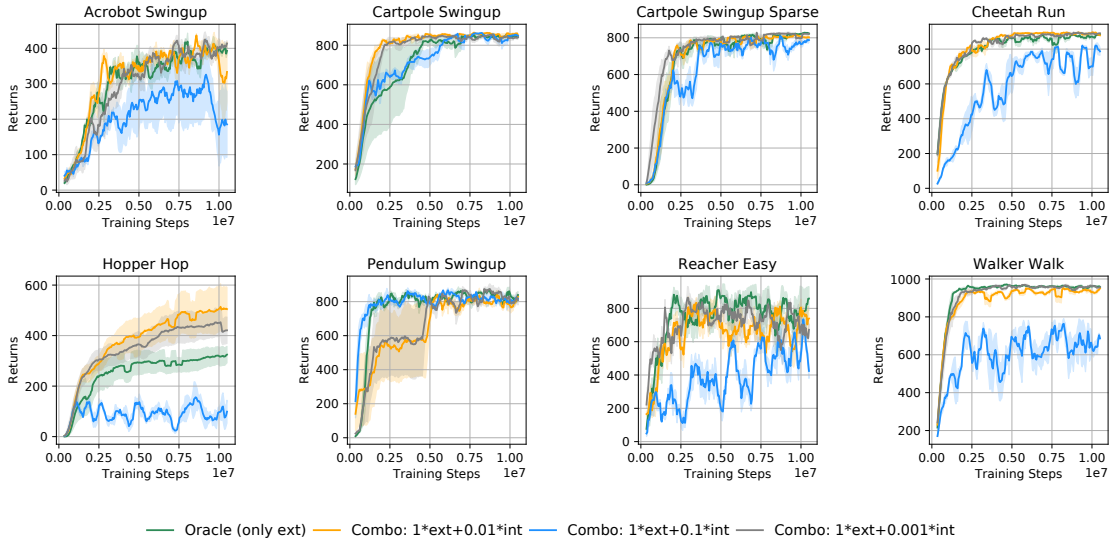


Figure 18: We compare the performance of the supervised oracle with a supervised approach that also has exploration bonuses. We perform ablations with the coefficient of the exploration bonus, and it is clear that the agent with our intrinsic reward as a bonus with a coefficient of 0.01 augments the supervised oracle well in tasks where the oracle struggles

well, we train the exploration policy by using REINFORCE (Williams (1992)) and optimize the exploration policy head $\pi_{\phi'}$ as

$$\phi' \leftarrow \phi' + \alpha \mathbb{E}_{\pi}[\sum_{\tau=t}^{t+H} \Psi_{\tau} \nabla_{\phi'} \log \pi_{\phi'}(a_{\tau}|s_{\tau})] \tag{4.1}$$

where $\Psi_{\tau} = V_{\lambda}(s_{\tau})$ is the lambda-return value estimate. As can clearly be seen, our method outperforms both the variants of MAX. It is however not clear if training MAX policies with policy gradients is any better than training them with model gradients. More experiments need to be made in this avenue.

4.7. Exploration Bonuses

While not being the focus of the method or this thesis, we also look into whether we can use the intrinsic rewards as exploration bonuses to the extrinsic rewards. To

achieve this, we train our ensemble as usual, but we don't have a separate policy or value head for exploration. We simply generate the intrinsic rewards by using the latent disagreement through the rollout in the imagination that the extrinsic task policy head executes. We combine the final rewards to the policy head as some linear combination of the intrinsic and the extrinsic rewards. This experiment is shown in Figure 18. We vary the coefficient of the intrinsic reward and see the performance difference. It appears that with a intrinsic coefficient of 0.01, our method augments the supervised oracle well. The coefficient of 0.1 seems to be too large, meaning the policy can't decide what to optimize between the intrinsic and extrinsic objectives, and a coefficient of 0.001 seems to be too little, meaning the policy is simply optimizing only for the extrinsic rewards. However, the gains here by using exploration bonuses are not high. This might mean that these tasks are not challenging exploration problems, and that since Dreamer already solves most of these tasks, it really isn't a necessity to give a strategic exploration bonus, although in the tasks where Dreamer struggles, like Hopper Hop, our exploration bonus seems to have improved the performance.

5. Conclusions

We presented Plan2Explore, a task-agnostic reinforcement learning method that learns a global model of its environment through unsupervised exploration and uses this model to solve tasks in a zero-shot or few-shot manner. We derived connections of our method to information gain, a principled objective for Bayesian exploration. Building on recent work on learning dynamics models and behaviors from images, we constructed a model-based zero-shot reinforcement learning agent that was able to achieve state-of-the-art zero-shot task performance on a range of DeepMind control suite tasks. Moreover, the agent's zero-shot performance was competitive to Dreamer, a state-of-the-art supervised reinforcement learning agent on some tasks, with the few-shot performance eventually matching or outperforming the supervised agent. By presenting a method that is able to learn effective behavior for many different tasks in a scalable and data-efficient manner, we hope this work constitutes a step towards building scalable real-world learning systems that are able to interact with their environment and solve complex real-world tasks.

BIBLIOGRAPHY

J. Achiam and S. Sastry. Surprise-based intrinsic motivation for deep reinforcement learning. *arXiv:1703.01732*, 2017.

B. Amos, L. Dinh, S. Cabi, T. Rothörl, A. Muldal, T. Erez, Y. Tassa, N. de Freitas, and M. Denil. Learning awareness models. In *International Conference on Learning Representations*, 2018.

E. Banijamali, R. Shu, M. Ghavamzadeh, H. Bui, and A. Ghodsi. Robust locally-linear controllable embedding. *arXiv preprint arXiv:1710.05373*, 2017.

M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos. Unifying count-based exploration and intrinsic motivation. In *NIPS*, 2016.

Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018a.

Y. Burda, H. Edwards, A. Storkey, and O. Klimov. Exploration by random network distillation. *arXiv preprint arXiv:1810.12894*, 2018b.

Y. Burda, H. Edwards, D. Pathak, A. Storkey, T. Darrell, and A. A. Efros. Large-scale study of curiosity-driven learning. *ICLR*, 2019.

K. Chua, R. Calandra, R. McAllister, and S. Levine. Deep reinforcement learning in a handful of trials using probabilistic dynamics models. *arXiv preprint arXiv:1805.12114*, 2018a.

K. Chua, R. McAllister, R. Calandra, and S. Levine. Unsupervised exploration with deep model-based reinforcement learning. 2018b.

M. O. Duff and A. Barto. *Optimal Learning: Computational procedures for Bayes-adaptive Markov decision processes*. PhD thesis, University of Massachusetts at Amherst, 2002.

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

Y. Gal. Uncertainty in deep learning. *University of Cambridge*, 1:3, 2016.

D. Ha and J. Schmidhuber. World models. *arXiv preprint arXiv:1803.10122*, 2018.

T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. Soft actor-critic: Off-policy maxi-

mum entropy deep reinforcement learning with a stochastic actor. *arXiv preprint arXiv:1801.01290*, 2018.

D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. Learning latent dynamics for planning from pixels. *arXiv preprint arXiv:1811.04551*, 2018.

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

M. Henaff, A. Canziani, and Y. LeCun. Model-predictive policy learning with uncertainty regularization for driving in dense traffic. *ICLR*, 2019.

T. Jaksch, R. Ortner, and P. Auer. Near-optimal regret bounds for reinforcement learning. *Journal of Machine Learning Research*, 11(Apr):1563–1600, 2010.

L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, et al. Model-based reinforcement learning for atari. *arXiv preprint arXiv:1903.00374*, 2019.

S. Kakade and J. Langford. Approximately optimal approximate reinforcement learning. In *ICML*, volume 2, pages 267–274, 2002.

D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

A. S. Klyubin, D. Polani, and C. L. Nehaniv. Empowerment: A universal agent-centric measure of control. In *Evolutionary Computation*, 2005.

B. Lakshminarayanan, A. Pritzel, and C. Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. In *Advances in Neural Information Processing Systems*, pages 6402–6413, 2017.

J. Lehman and K. O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th annual conference on Genetic and evolutionary computation*, 2011.

S. Levine. Cs 285 at uc berkeley, deep reinforcement learning. *http://rail.eecs.berkeley.edu/deeprlcourse/*, 2019.

D. V. Lindley. On a measure of the information provided by an experiment. *The Annals of Mathematical Statistics*, pages 986–1005, 1956.

M. Lopes, T. Lang, M. Toussaint, and P.-Y. Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *NIPS*, 2012.

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

D. J. MacKay. Information-based objective functions for active data selection. *Neural computation*, 4(4):590–604, 1992.

A. K. McCallumzy and K. Nigamy. Employing em and pool-based active learning for text classification. In *Proc. International Conference on Machine Learning (ICML)*, pages 359–367. Citeseer, 1998.

V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. P. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *ICML*, 2016.

I. Osband, C. Blundell, A. Pritzel, and B. Van Roy. Deep exploration via bootstrapped dqn. In *NIPS*, 2016.

I. Osband, J. Aslanides, and A. Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 8617–8629, 2018.

G. Ostrovski, M. G. Bellemare, A. v. d. Oord, and R. Munos. Count-based exploration with neural density models. *ICML*, 2018.

P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in neurorobotics*, 2009.

P.-Y. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *Evolutionary Computation*, 2007.

D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *ICML*, 2017.

D. Pathak, D. Gandhi, and A. Gupta. Self-supervised exploration via disagreement. *ICML*, 2019.

P. Poupart, N. Vlassis, J. Hoey, and K. Regan. An analytic solution to discrete bayesian reinforcement learning. In *ICML*, 2006.

D. J. Rezende, S. Mohamed, and D. Wierstra. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082*, 2014.

A. G. Richards. *Robust constrained model predictive control*. PhD thesis, Massachusetts Institute of Technology, 2005.

R. Y. Rubinstein. Optimization of computer simulation models with rare events. *European Journal of Operational Research*, 99(1):89–112, 1997.

J. Schmidhuber. Curious model-building control systems. In *Neural Networks, 1991. 1991 IEEE International Joint Conference on*, pages 1458–1463. IEEE, 1991a.

J. Schmidhuber. A possibility for implementing curiosity and boredom in model-building neural controllers. In *From animals to animats: Proceedings of the first international conference on simulation of adaptive behavior*, 1991b.

J. Schulman, N. Heess, T. Weber, and P. Abbeel. Gradient estimation using stochastic computation graphs. In *NIPS*, 2015.

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

H. Seung, M. Opper, and H. Sompolinsky. Query by committee. *COLT*, 1992.

P. Shyam, W. Jaśkowski, and F. Gomez. Model-Based Active Exploration. In *ICML*, 2019.

A. Strehl and M. Littman. An analysis of model-based interval estimation for markov decision processes. *Journal of Computer and System Sciences*, 2008.

Y. Sun, F. Gomez, and J. Schmidhuber. Planning to be surprised: Optimal bayesian exploration in dynamic environments. In *AGI*, 2011.

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

Y. Tassa, Y. Doron, A. Muldal, T. Erez, Y. Li, D. de Las Casas, D. Budden, A. Abdolmaleki, J. Merel, A. Lefrancq, T. Lillicrap, and M. Riedmiller. DeepMind control suite. Technical report, DeepMind, Jan. 2018. URL `https://arxiv.org/abs/1801.00690`.

M. Watter, J. Springenberg, J. Boedecker, and M. Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *NIPS*, 2015.

T. Weber, S. Racanière, D. P. Reichert, L. Buesing, A. Guez, D. J. Rezende, A. P. Badia, O. Vinyals, N. Heess, Y. Li, et al. Imagination-augmented agents for deep reinforcement learning. *arXiv preprint arXiv:1707.06203*, 2017.

R. J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.