SELF-SUPERVISED EXPLORATION VIA LATENT BAYESIAN SURPRISE

Anonymous authors
Paper under double-blind review

ABSTRACT

Training with Reinforcement Learning requires a reward function that is used to guide the agent towards achieving its objective. However, designing smooth and well-behaved rewards is in general not trivial and requires significant human engineering efforts. Generating rewards in self-supervised way, by inspiring the agent with an intrinsic desire to learn and explore the environment, might induce more general behaviours. In this work, we propose a curiosity-based bonus as intrinsic reward for Reinforcement Learning, computed as the Bayesian surprise with respect to a latent state variable, learnt by reconstructing fixed random features. We extensively evaluate our model by measuring the agent's performance in terms of environment exploration, for continuous tasks, and looking at the game scores achieved, for video games. Our model is computationally cheap and empirically shows state-of-the-art performance on several problems. Furthermore, experimenting on an environment with stochastic actions, our approach emerged to be the most resilient to simple stochasticity. Further visualization is available on the project webpage.[1]

1 INTRODUCTION

Agents can be trained with Reinforcement Learning (RL) to successfully accomplish tasks by maximising a reward signal, which should likely encourage correct behaviours and penalize wrong actions. For instance, agents can learn to play video games by maximizing the game score (Mnih et al., 2015) or achieve robotic manipulation tasks, such as solving a Rubik's cube (OpenAI et al., 2019), by following human-engineered rewards. However, how to correctly define reward functions to develop general skills remains an unsolved problem (Amodei et al., 2016).

In contrast to RL agents, humans can learn behaviours without any external rewards, due to the intrinsic motivation that naturally drives them to be active and explore the environment. The process of skill acquisition can then be expressed as a selection process over successful exploratory behaviours (Corbetta et al., 2018). The design of similar mechanisms for RL agents opens to the possibility of training and evaluating agents without external rewards (Matusch et al., 2020), potentially leading to more general strategies of learning.

The idea of instilling intrinsic motivation, or 'curiosity', into artificial agents has raised a large interest in the RL community (Oudeyer et al., 2007; Schmidhuber, 1991). One common approach consists of learning a model of the environment that is used to generate intrinsic rewards, which can replace or complement the external reward function. However, what is the best approach to generate such intrinsic bonuses is still unknown.

Several successful approaches modeled intrinsic rewards as the 'surprisal' information of the model. In layman's terms, this can be described as the difference between the agent's belief about the environment state and the ground truth, and can be implemented as the model's predictions error (Achiam & Sastry, 2017; Pathak et al., 2017). Alternatively, Bayesian surprise (Itti & Baldi, 2006) may also be used as a bonus to incentive exploration, albeit existing work has mostly focused on computationally-expensive approaches, which evaluate surprise in the model's parameters space (Houthooft et al., 2016).

[1] https://doubleblind-repos.github.io/

Contributions. In this work, we present an intrinsic bonus based on the concept of Bayesian surprise, efficiently computed with respect to a latent state variable in the dynamics. The dynamics learning process is obtained by optimizing a lower bound objective that reconstructs compact features, obtained from the environment's observations, using a fixed feature model.

The main contributions of this work are as follows: (*i*) a latent dynamics model, which captures the unobserved state of the environment and allows to reconstruct observations in a compact feature space, (*ii*) a new latent Bayesian surprise exploration bonus, representing the information gain with respect to the latent state variable of the dynamics, (*iii*) evaluation of the approach on several continuous-actions robotic simulation tasks and on discrete-actions video games, and comparison with other exploration strategies (*iv*) preliminary assessment of resilience to stochasticity in actions, by comparing to baselines and ablations on an image-prediction environment.

The results empirically show that our **L**atent **B**ayesian **S**urprise (**LBS**) model performs on par and often outperforms state-of-the-art methods, while also being more resilient to stochasticity.

2 Intrinsic Motivation for Reinforcement Learning

We focus on bonus-based methods as intrinsic motivation to incentivize exploration in RL. To foster reader's understanding, we first introduce standard notation and common practices in the area.

Markov Decision Processes. The RL setting can be formalized as a Partially Observable Markov Decision Process (POMDP), which is denoted with the tuple $\mathcal{M} = \{\mathcal{S}, \mathcal{A}, T, R, \Omega, \mathcal{O}, \gamma\}$, where $\mathcal{S}$ is the set of unobserved states, $\mathcal{A}$ is the set of actions, T is the state transition function, also referred to as the dynamics of the environment, R is the reward function, which maps transitions into rewards, Ω is the set observations, $\mathcal{O}$ is a set of conditional observation probabilities, and γ is a discount factor. The objective of the RL agent is to maximize the discounted sum of rewards, or return, $G_t = \sum_{k=t+1}^{T} \gamma^{(k-t-1)} r_k$, where t indicates the time step at which the return is computed.

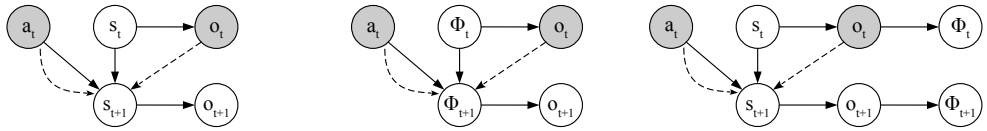
Policy Optimization. In order to maximize the return function, the agent should condition its actions on the basis of the observations coming from the environment. The policy function $\pi(a_t|o_t)$ is used to represent the probability of taking action a_t when observing o_t. Several policy-optimization algorithms also evaluate two value functions, $V(o_t)$ and $Q(o_t, a_t)$, that are used to estimate and predict future rewards with respect to a certain observation or observation-action pair, respectively.

Dynamics Model. Exploration bonuses as intrinsic rewards can be generated using the surprisal of a model of the environment's dynamics, which is formalized as $p(s_{t+1}|o_t, a_t)$. In curiosity-driven RL, it has been common practice to compute predictions on a compact feature space. For instance, features $\phi_t = f(o_t)$ can be computed with the mapping $f : \mathbb{R}^n \to \mathbb{R}^m$, with $m \leq n$, turning the model dynamics into $p(\phi_{t+1}|o_t, a_t)$. To distinguish it from the true state dynamics of the POMDP, we refer to this formulation as Feature Dynamics.

Intrinsic Rewards. Intrinsic motivation is instilled into the agent by combining the generated intrinsic rewards with the external rewards of the environment. The combined reward at time step t is represented as: $r_t = \eta_e r_t^{(e)} + \eta_i r_t^{(i)}$, where $r^{(e)}$ is the external reward, $r^{(i)}$ is the intrinsic bonus, and η_e and η_i are factors adopted to balance external and intrinsic rewards. How to optimally balance between exploration with intrinsic motivation and exploitation of external rewards is still an unresolved question. To prevent this issue from affecting our experiments, we assign any external rewards to zero. This is implemented by setting $\eta_e = 0$, focusing the agents solely on the exploratory behaviours inspired by the curiosity mechanism.

3 Latent Bayesian Surprise

Our model provides intrinsic motivation through a Bayesian surprise signal, which is computed with respect to a latent state variable. First, we describe how the latent dynamics inference works and how our model is trained by optimizing a variational lower bound on the log-likelihood of future features, then explain how the intrinsic reward signal is obtained, and finally discuss the intuition behind our work and connections with other methods.



(a) POMDP Model (b) Feature Dynamics Model (c) LBS Model (ours)

Figure 1: **Dynamics graphical models.** In all examples, the model observes o_t and a_t. Solid lines indicate generative processes and dashed lines indicate the inference ones.

Latent Dynamics. In a typical POMDP, the model dynamics should predict next states from current observations and actions. This is represented in Figure 1a, where the agent, after having adopted action a_t, reaches a state of the environment that is observed as o_{t+1}. The dynamics model infers s_{t+1} as $p(s_{t+1}|o_t, a_t)$. Generally, in the curiosity-driven RL literature, hidden true states of the POMDP are overlooked and the state s_t is replaced by features ϕ_t that are computed as point estimates. In this setup, the dynamics of the environment is modeled with $p(\phi_{t+1}|o_t, a_t)$, as for Figure 1b.

Using fixed deterministic features for exploration, either random or pre-trained, has shown empirically better results than training features concurrently with the dynamics model (Burda et al., 2019a;b), likely because of their greater stability. However, a Feature Dynamics model based on fixed deterministic features can be dangerous for several reasons: 1) deterministic features do not allow to model uncertainty e.g. stochasticity in observations and transitions; 2) feature reduction may always generate inconsistencies and ambiguities, which may lead to fit fictitious transitions.

To exploit fixed features, and contemporary alleviate the above problems, we introduce the latent random variable s_t in the Feature Dynamics model. As for Figure 1c, the succession of states s_t is used to capture the dynamics, in a POMDP fashion, while features are treated as the result of a generation process from their respective observations. In this setup, the environment dynamics can be modeled with respect to the latent state variable, and denoted as $p(s_{t+1}|o_t, a_t)$, and future features can be reconstructed from the latent state, as $p(\phi_{t+1}|s_{t+1})$, rather than reconstructing observations.

As illustrated in Figure 2, our LBS model is made of the following components:

Latent Dynamics:	$p(s_{t+1}\|o_t, a_t)$
Latent Posterior:	$q(s_{t+1}\|o_t, a_t, o_{t+1})$
Feature Reconstruction:	$p(\phi_{t+1}\|s_{t+1})$

where the latent dynamics model represents prior beliefs over next states, $q(s_{t+1})$ is an approximation of the dynamics posterior with respect to the latent state variable, and the features reconstruction module allows to regenerate compact features from the corresponding latent state.

All the components are jointly trained by maximizing the following variational lower bound on future features log-likelihood:

$$\mathcal{J} = \mathbb{E}_{s_{t+1}\sim q(s)}[\log p(\phi_{t+1}|s_{t+1})] - \beta D_{\mathrm{KL}}[q(s_{t+1}|o_t, a_t, o_{t+1}) \| p(s_{t+1}|o_t, a_t)] \tag{1}$$

where β is introduced to incentivize disentanglement in the latent state representation, as in (Higgins et al., 2017). The derivation of the objective is available in Appendix A.

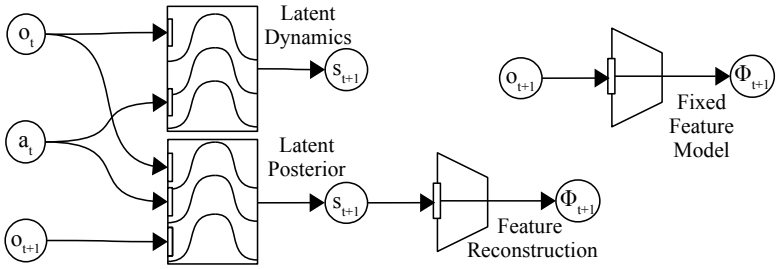


Figure 2: **LBS architecture.** The modules of LBS, with input and output variables. Latent Dynamics and Posterior output distributions, while the Feature Reconstruction outputs point estimates.

Reconstructing features ϕ_{t+1}, rather than observations, has several advantages. First, it has been demonstrated by numerous experiments that most of the information contained in the high-dimensional observations may be neglectable for exploration (Burda et al., 2019a). Thus, features reconstruction may allow faster computation, without compromising performance. Secondly, the use of features in the reconstruction objective avoids any shortcuts in the posterior model.

Bayesian Surprise Bonus. In information theory, Bayesian surprise is defined as the information gain $\mathcal{I}$ obtained about a random variable by observing another random variable. In the context of our method, we are interested in measuring the amount of information that is gained by the LBS model when facing a new environment's transition. After taking action a_t while being in the latent state s_t, the agent would receive a new observation o_{t+1} that completes the transition and brings newer information to the latent dynamics model.

Such information gain can be formulated as the KL divergence between the latent dynamics $p(s_{t+1})$ and its approximate posterior $q(s_{t+1}|o_{t+1})$ and adopted as an intrinsic reward for RL as follows:

$$r_t^{(i)}(o_t, a_t, o_{t+1}) = \mathcal{I}(o_{t+1}, s_{t+1}|o_t, a_t) \approx D_{\mathrm{KL}}[q(s_{t+1}|o_t, a_t, o_{t+1})\|p(s_{t+1}|o_t, a_t)] \tag{2}$$

The above term can be efficiently computed by comparing the predictions of the latent dynamics and of the latent posterior models, in an amortized inference fashion. Such signal should encourage the agent to collect transitions where the latent dynamics predictions are more uncertain or erroneous.

Connections. The intuition behind LBS may be interpreted as the intersection of two known ideas in the exploration landscape: one consists of computing features using a fixed feature network and then using the error prediction of a 'distillation' model as curiosity (Burda et al., 2019b), the other comes from the promising approach of using Bayesian surprise as an exploration signal (Houthooft et al., 2016). The introduction of a latent random variable allows to combine the two ideas, while also allowing to model stochasticity and feature's inadequacies in a probabilistic fashion.

In the LBS model, the latent posterior can be seen as a distillation process from next observations to next features, through the latent state. Instead, the latent dynamics allows to compute prior beliefs, enabling the computation of Bayesian surprise in an efficient way, by exploiting amortized inference.

4 Experiments

The aim of the experiments is to compare the performance of our LBS model and of its latent Bayesian surprise bonus against other approaches for self-supervised exploration in RL.

Environments. Results are presented with respect to two sets of environments: one of continuous control tasks and one with discrete-action games. The continuous control tasks include the classic Mountain Car environment (Moore, 1990), the Mujoco-based Half-Cheetah environment (Todorov et al., 2012), and the Ant Maze environment used in (Shyam et al., 2019). The discrete-action games include 8 video games from the Atari Learning Environment (ALE; Bellemare et al. (2013)) and the Super Mario Bros. game, which is a popular NES platform game. For all tasks, we update the policy using the Proximal Policy Optimization algorithm (PPO; Schulman et al. (2017)).

Implementation Details. The LBS model components are implemented as follows: the latent dynamics and posterior models are deep neural networks with variational outputs, representing multivariate gaussian distributions; the feature reconstruction model is implemented as a multi-layer perceptron (MLP). Parameter sharing is enabled between the posterior and the prior, by using the last layer of the latent dynamics, concatenated with new observations, as an input to the posterior.

To easily obtain fixed features from observations, we follow the trend that empirically found random features to be working well for exploration (Burda et al., 2019a; Pathak et al., 2019; Burda et al., 2019b). Fixed features ϕ_t are obtained by processing observations o_t with a randomly initialised network that is never updated. This network is either represented by a single linear layer, for continuous control tasks, or by a small convolutional network, for vision-based tasks.

Further details on the implementation and the experiments can be found in Appendix B and C.

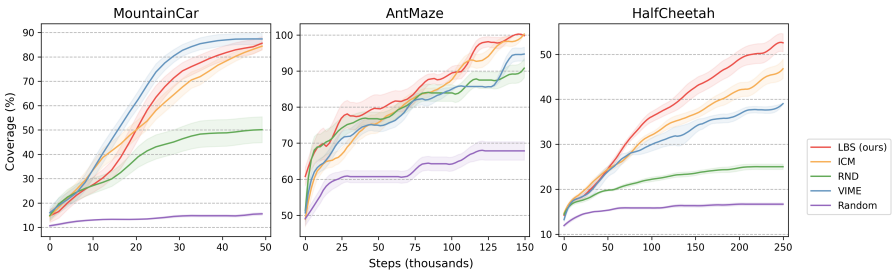


Figure 3: **Continuous Control results.** A comparison of our method against several baselines on continuous control tasks. Lines show the mean state-space coverage (with standard deviation) in terms of percentage of bins visited by the agents, trained only with intrinsic rewards.

4.1 Continuous Control

In our continuous control experiments, we discretize the observation-space into bins and compare the number of bins explored, in terms of coverage percentage, against the following baselines:

- *Intrinsic Curiosity Model* (ICM; Pathak et al. (2017)): intrinsic rewards are computed as the mean-squared error (MSE) between network's predictions in feature space and the true features. Observation are processed into features using a feature network, trained jointly with the dynamics model to optimize an inverse-dynamics objective.
- *Random Network Distillation* (RND; Burda et al. (2019b)): features are obtained with a fixed randomly initialized network. Intrinsic rewards for each transition are computed as the prediction errors between next-observation features and the output of a distillation network, which is trained to match the outputs of the random feature network.
- *Variational Information Maximizing Exploration* (VIME; Houthooft et al. (2016)): the dynamics is modeled as a Bayesian neural network (BNN; Bishop (1997)). Intrinsic rewards for single transitions are shaped as the information gain computed with respect to the BNN's parameters before and after updating the network, using the transition's values.
- *Random*: a naive agent that explores by performing a series of random actions.

The training curves are presented in Figure 3, averaging over runs with eight different random seeds.

Mountain Car. In MountainCar, the limited observation space is discretized into 100 bins. As shown in Figure 3, VIME is the best performing approach, with LBS and ICM following, slightly behind. While the first three models could reach around 85% of the visitable bins, RND struggles behind with about 50% of visited bins, which is still 35% better than Random exploration ($\sim$15%).

Ant Maze. In the Ant Maze environment, the agent can explore up to seven bins, corresponding to different aisles of a maze. LBS and ICM perform best in this environment, eventually reaching the end of the maze in all runs. VIME ($\sim$95%) and RND ($\sim$91%) perform comparably, achieving a solid result against the coverage achieved by the Random baseline ($\sim$68%).

Half-Cheetah. In the Half-Cheetah environment, the observation space is discretized into 100 bins. Performance in this task see LBS reaching the highest number of bins, with around 53% of coverage. They follow ICM ($\sim$47%) and VIME ($\sim$39%). RND lacks behind by slightly outperforming the Random baseline ($\sim$25% vs $\sim$17%).

4.2 Arcade Games

For the arcade games, the environments chosen are designed in a way that either requires the player to explore in order to succeed, e.g. Qbert, or to survive as long as possible to avoid boredom, e.g. Pong. For this reason, agents are trained only with curiosity, but evaluated on the game score they achieve in one episode, or on the distance covered from the initial position (only Super Mario Bros.).

We follow the setup of (Burda et al., 2019a) and compare against their baselines, which use MSE prediction error in feature space as the intrinsic motivation signal. The difference between the Vari-

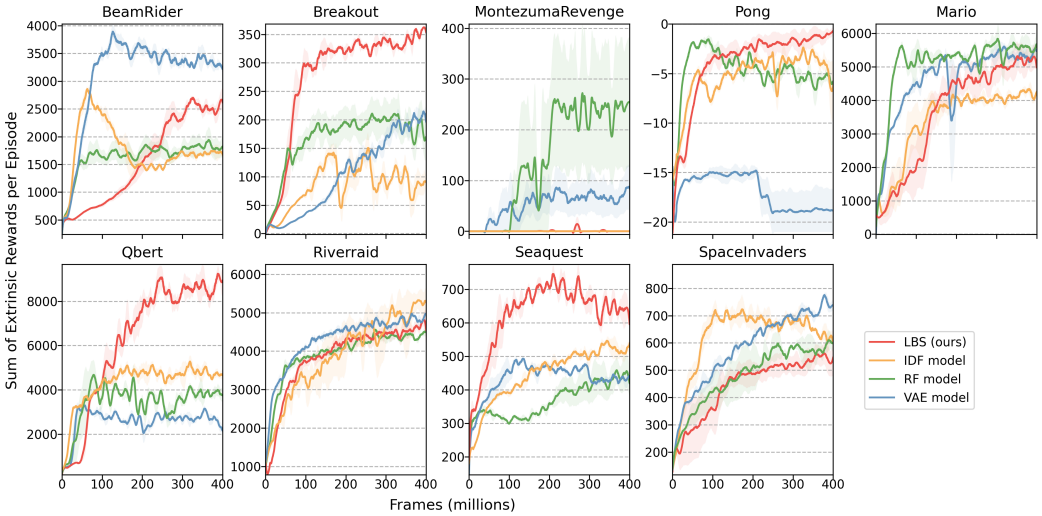


Figure 4: **Arcade Games results.** A comparison of our method against error-based models, using different sets of features, on 8 selected Atari and the Super Mario Bros games. Lines show the mean sums of rewards (and standard deviations) of agents trained with intrinsic rewards, over three seeds.

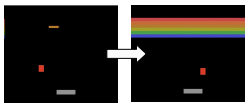
ational Autoencoder, or *VAE model*, the Random Features, or *RF model*, and the Inverse Dynamics Features, or *IDF model*, lies in the features adopted. The VAE model trains an autoencoder, as in (Kingma & Welling, 2014), concurrently with the dynamics model; the RF model uses a randomly initialized network; the IDF model trains features that allows to model inverse dynamics.

A comparison against the above baselines, using different features on high-dimensionality observations, should demonstrate the relevance of our latent Bayesian surprise bonus, compared to error-prediction bonuses, regardless of the features set adopted. The training curves are presented in Figure 4, averaging over runs with three different random seeds.

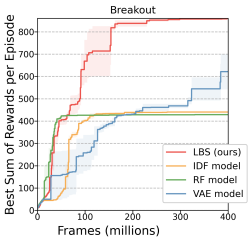
Atari. The results in the selected Atari games are favorable towards the LBS model. It achieves the best final score in 4 out 8 games: Pong, Seaquest, Breakout and Qbert, with a large margin for the latter twos; and performs comparably in all other games, apart from Montezuma Revenge.

Interestingly, in Breakout, the only two levels available have exactly the same game mechanics and graphics, so that when the agent breaks the last brick of Level 1, Level 2 starts exactly as a game reset, as depicted in Figure 5a. For a curiosity-driven RL agent, this is totally disruptive as the model tends to assign low intrinsic rewards to the start of Level 2, which provides well-known observations. This fact is exacerbated by the curves of the best scores, in Figure 5b, which shows that LBS was able to reach high scores, but eventually kept an average score below the Level 1's threshold (432)[2].

[2]Further visual proof is available in our project page, where the Breakout agent is showed to prefer directing the ball against the walls rather than destroying the last bricks of Level 1.



(a) Breakout Level Reset



(b) Best Scores Breakout

Figure 5: Breakout Insights

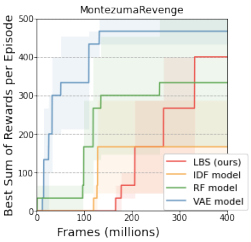
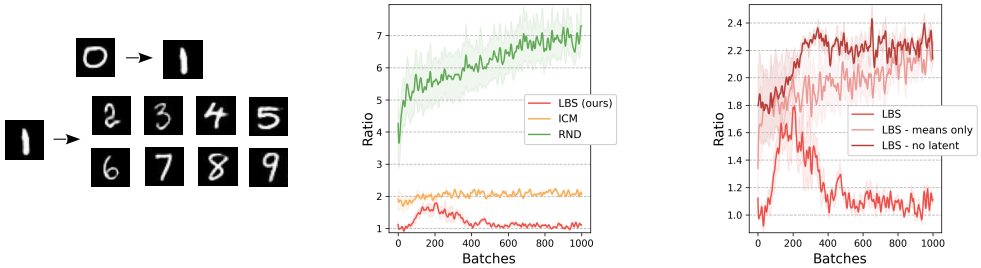


Figure 6: Best Scores on the Montezuma Revenge game



(a) Noisy MNIST task (b) Baselines Comparison. (c) Ablation Comparison..

Figure 7: **Stochasticity Ablation.** (a) Graphical presentation of the fictitious dynamics of the task. (b-c) Intrinsic motivation ratio over training batches achieved by LBS, ICM, RND, and LBS ablations The closer the ratio to the unity, at convergence, the better.

As for Montezuma Revenge, we present in Figure 6 the max scores achieved by all methods, showing that also IDF and LBS models could actually reach non-zero scores during training. It is unclear why the final scores of the latter got to zero, and further analysis should be conducted about this. However, concerns about the use of Montezuma Revenge as a benchmark for exploration methods have already been raised in the past (Taïga et al., 2019), as the games mostly requires other kind of systematic approaches to reach higher scores, rather than vanilla exploration (Badia et al., 2020).

Super Mario Bros. The final distance reached in the Mario's game are comparable for LBS, RF and VAE models, with ICM behind. It may be of interest, that different runs of our models found different paths to solve World 1-1. One model went down from the pipe at the start of the level, reaching the end by using a shortcut, while the other two ran and jumped across all the level.

4.3 Stochastic Environment

In this environment, as in Pathak et al. (2019), we use the Noisy MNIST dataset to perform an experiment on stochastic actions. Taking examples from the test set of MNIST, we built a fictitious dynamics that always starts either from an image of a zero or a one, as in Figure 7a: a 0-image always transitions to a 1-image, while a 1-image transitions into a number between two and nine.

We assess the performance in terms of the ratio between the intrinsic motivation provided for transitions starting from 1-images and transitions starting from 0-images. A well-behaved exploration model should eventually understand that results associated with the more stochastic 1-image transitions belong to the same category, and lose interest with respect to them. Thus, the correct expected behaviour is that the ratio should eventually lean to values close to the unity.

In Figure 7b, we compare LBS to ICM and RND and observe that LBS is the only model that eventually captures the stochasticity in the transitions starting from 1-images, eventually lowering the ratio at convergence. We further benchmark against two ablations of LBS: *LBS - means only*, which takes only the means from the latent dynamics and posterior outputs, computing both the training loss and the intrinsic rewards accordingly; *LBS - no latent*, which gets rid of the latent state and instead uses stochastic random features with fixed variance. In the latter, the training loss and the intrinsic rewards are both collapsed into a KL divergence term between the dynamics model and the true features posterior. As shown in Figure 7c, both the latent state and the use of a random variable as a latent are important components of LBS that help capturing stochasticity in the transitions.

5 Related Work

Reinforcement Learning. Value-based methods in RL use the Q-value function to choose the best action in discrete settings (Mnih et al., 2015; Hessel et al., 2018). However, the Q-value approach cannot scale well to high-dimensional outputs or continuous environments. Policy Optimization techniques solve these problems by directly optimizing the policy, by either learning online, using samples collected from the policy (Schulman et al., 2015; 2017), or offline, reusing the experience stored in a replay buffer (Lillicrap et al., 2016; Haarnoja et al., 2018).

Latent Dynamics. In complex environments, the use of latent dynamics models has proven successful for control and long-term planning, either by using VAEs to model locally-linear latent states (Watter et al., 2015), or by using recurrent world models in POMDPs (Buesing et al., 2018; Hafner et al., 2019; 2020).

Intrinsic Motivation. Several exploration strategies have been presented in Section 4, concerning entropy maximization in the transitions (Pathak et al., 2017), with respect to future observations (Burda et al., 2019b) or model's parameters (Houthooft et al., 2016). In Disagreement (Pathak et al., 2019), they train an ensemble of dynamics models and use the variance in their predictions as intrinsic motivation. In VDM (Bai et al., 2020), they model a latent variable to reconstruct features (using a different reconstruction model from ours) and use the average surprisal over several sampled reconstructions as an exploration bonus.

Models that use alternative approaches to modelling the environment's dynamics are based on pseudo-counts (Bellemare et al., 2016; Ostrovski et al., 2017; Tang et al., 2017), which use density estimations techinques to explore less seen areas of the environment, Randomized Prior Functions (Osband et al., 2018), applying statistical bootstrapping and ensembles to the Q-value function model, or Noisy Nets (Fortunato et al., 2018), applying noise to the value-function network's layers.

Planning Exploration. Recent breakthroughs concerning exploration in RL have also focused on using the learned environment dynamics to plan to explore. This is the case of (Shyam et al., 2019) and (Ratzlaff et al., 2020), where they use imaginary rollouts from their dynamics models to plan exploratory behaviours, and (Sekar et al., 2020), where they combine a model-based planner in latent space (Hafner et al., 2020) with the Disagreement exploration strategy (Pathak et al., 2019).

6 Discussion

In this work, we introduced the LBS model to generate intrinsic rewards as the Bayesian suprise over a latent state variable. Our method has proven successful in several continuous-control and discrete-action settings, outperforming or performing on par with state-of-the-art methods for self-supervised exploration in RL.

The experiments in low-dimensional continuous-control tasks, which evaluate performance over an artificial metrics on the coverage of the observation space, have shown that our method provides more in-depth exploration than other methods. In particular, VIME seems to shine in low-dimensional environments such as Mountain Car but is struggling with higher-dimensional observations. RND generally performed worst in these low-dimensional experiments.

In our arcade games results, we provided additional insights about exploration in some video games, such as Breakout, Montezuma Revenge and Super Mario Bros., identifying issues to be considered for future works or highlighting interesting exploratory behaviours that emerged from our runs. Future benchmarks using these games would likely benefit from this information.

We also tested LBS to be resilient to stochasticity in the dynamics, in a simple fictitious task. We believe stochasticity is an important limitation that still affects exploration methods and future work should focus on understanding to which extent limitations apply and how to overcome them.

As our Bayesian surprise proved successful in several environments, we are also interested in applying it for planning exploration, by further exploiting our latent dynamics model. A straightforward implementation of this, could for instance adopt recurrent world models and latent overshooting (Hafner et al., 2019) to compute Bayesian surprise with respect to latent state predictions in the more distant future.

REFERENCES

Joshua Achiam and Shankar Sastry. Surprise-based intrinsic motivation for deep reinforcement learning, 2017.

Dario Amodei, Chris Olah, Jacob Steinhardt, Paul Christiano, John Schulman, and Dan Mané. Concrete problems in ai safety, 2016.

Adrià Puigdomènech Badia, Pablo Sprechmann, Alex Vitvitskyi, Zhaohan Daniel Guo, Bilal Piot, Steven Kapturowski, Olivier Tieleman, Martín Arjovsky, Alexander Pritzel, Andrew Bolt, and Charles Blundell. Never give up: Learning directed exploration strategies. In *8th International Conference on Learning Representations, ICLR 2020*, 2020.

Chenjia Bai, Peng Liu, Zhaoran Wang, Kaiyu Liu, Lingxiao Wang, and Yingnan Zhao. Variational dynamic for self-supervised exploration in deep reinforcement learning, 2020.

Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. The arcade learning environment: An evaluation platform for general agents. *J. Artif. Int. Res.*, 47(1):253–279, 2013.

Marc G. Bellemare, Sriram Srinivasan, Georg Ostrovski, Tom Schaul, David Saxton, and Rémi Munos. Unifying count-based exploration and intrinsic motivation. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 1479–1487, 2016.

Christopher M. Bishop. Bayesian Neural Networks. *Journal of the Brazilian Computer Society*, 4, 1997.

Lars Buesing, Theophane Weber, Sebastien Racaniere, S. M. Ali Eslami, Danilo Rezende, David P. Reichert, Fabio Viola, Frederic Besse, Karol Gregor, Demis Hassabis, and Daan Wierstra. Learning and querying fast generative models for reinforcement learning, 2018.

Yuri Burda, Harrison Edwards, Deepak Pathak, Amos J. Storkey, Trevor Darrell, and Alexei A. Efros. Large-scale study of curiosity-driven learning. In *7th International Conference on Learning Representations, ICLR*, 2019a.

Yuri Burda, Harrison Edwards, Amos J. Storkey, and Oleg Klimov. Exploration by random network distillation. In *7th International Conference on Learning Representations, ICLR 2019*, 2019b.

Daniela Corbetta, Abigail DiMercurio, Rebecca F. Wiener, John P. Connell, and Matthew Clark. How perception and action fosters exploration and selection in infant skill acquisition. In *Studying the Perception-Action System as a Model System for Understanding Development*, volume 55 of *Advances in Child Development and Behavior*, pp. 1–29. 2018.

Meire Fortunato, Mohammad Gheshlaghi Azar, Bilal Piot, Jacob Menick, Ian Osband, Alexander Graves, Vlad Mnih, Remi Munos, Demis Hassabis, Olivier Pietquin, Charles Blundell, and Shane Legg. Noisy networks for exploration. In *Proceedings of the International Conference on Representation Learning (ICLR 2018)*, 2018.

Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pp. 1861–1870, 2018.

Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 2555–2565, 2019.

Danijar Hafner, Timothy P. Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, 2020.

Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI*, pp. 3215–3222, 2018.

Irina Higgins, Loïc Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. beta-vae: Learning basic visual concepts with a constrained variational framework. In *5th International Conference on Learning Representations, ICLR 2017*, 2017.

Rein Houthooft, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. Vime: Variational information maximizing exploration. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, pp. 1117–1125, 2016.

Laurent Itti and Pierre Baldi. Bayesian surprise attracts human attention. In *Advances in Neural Information Processing Systems*, volume 18, pp. 547–554, 2006.

Diederik P. Kingma and Max Welling. Auto-Encoding Variational Bayes. In *2nd International Conference on Learning Representations, ICLR 2014, Banff, AB, Canada, April 14-16, 2014, Conference Track Proceedings*, 2014.

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *ICLR*, 2016.

Brendon Matusch, Jimmy Ba, and Danijar Hafner. Evaluating agents without rewards, 2020.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

A. Moore. Efficient memory-based learning for robot control, 1990.

OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. Solving rubik's cube with a robot hand, 2019.

Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 31, pp. 8617–8629, 2018.

Georg Ostrovski, Marc G. Bellemare, Aäron van den Oord, and Rémi Munos. Count-based exploration with neural density models. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 2721–2730, 2017.

P. Oudeyer, F. Kaplan, and V. V. Hafner. Intrinsic motivation systems for autonomous mental development. *IEEE Transactions on Evolutionary Computation*, 11(2):265–286, 2007.

Deepak Pathak, Pulkit Agrawal, Alexei A. Efros, and Trevor Darrell. Curiosity-driven exploration by self-supervised prediction. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, pp. 2778–2787, 2017.

Deepak Pathak, Dhiraj Gandhi, and Abhinav Gupta. Self-supervised exploration via disagreement. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5062–5071, 2019.

Neale Ratzlaff, Qinxun Bai, Li Fuxin, and Wei Xu. Implicit generative modeling for efficient exploration. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 7985–7995, 2020.

J. Schmidhuber. Curious model-building control systems. In *[Proceedings] 1991 IEEE International Joint Conference on Neural Networks*, pp. 1458–1463 vol.2, 1991.

John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1889–1897, 2015.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

Ramanan Sekar, Oleh Rybkin, Kostas Daniilidis, Pieter Abbeel, Danijar Hafner, and Deepak Pathak. Planning to explore via self-supervised world models. In *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 8583–8592, 2020.

Pranav Shyam, Wojciech Jaśkowski, and Faustino Gomez. Model-based active exploration. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pp. 5779–5788, 2019.

Haoran Tang, Rein Houthooft, Davis Foote, Adam Stooke, Xi Chen, Yan Duan, John Schulman, Filip De Turck, and Pieter Abbeel. #exploration : a study of count-based exploration for deep reinforcement learning. In *Advances in Neural Information Processing Systems*, volume 30, pp. 1–18, 2017.

Adrien Ali Taïga, William Fedus, Marlos C. Machado, Aaron Courville, and Marc G. Bellemare. Benchmarking bonus-based exploration methods on the arcade learning environment, 2019.

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

Manuel Watter, Jost Tobias Springenberg, Joschka Boedecker, and Martin Riedmiller. Embed to control: A locally linear latent dynamics model for control from raw images. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'15, pp. 2746–2754, 2015.

A LATENT DYNAMICS LOWER BOUND

The LBS model's loss is obtained by maximizing a variational lower bound (ELBO) on future features log-likelihood. The ELBO derivation employs Jensen's inequality, as shown:

$$\begin{aligned}
\log p(\phi_{t+1}|o_t, a_t) &\triangleq \log \mathbb{E}_{s_{t+1}\sim p(s)}[p(s_{t+1}, \phi_{t+1}|o_t, a_t)] \\
&= \log \mathbb{E}_{s_{t+1}\sim p(s)}[p(s_{t+1}|o_t, a_t)p(\phi_{t+1}|s_{t+1})] \\
&\geq \mathbb{E}_{s_{t+1}\sim q(s)}[\log(p(s_{t+1}|o_t, a_t)p(\phi_{t+1}|s_{t+1})/q(s_{t+1})] \\
&= \mathbb{E}_{s_{t+1}\sim q(s)}[\log p(\phi_{t+1}|s_{t+1})] - D_{\mathrm{KL}}[q(s_{t+1})\|p(s_{t+1})]
\end{aligned}$$

B IMPLEMENTATION DETAILS

Mountain Car. The agent is spawned at the bottom of a valley and can move across two hills, by adopting an adequate velocity action. The observation space is two-dimensional, and limited to the following ranges: [-1.2, 0.6] for the position and [-0.07, 0.07] for the velocity. We divide each range in ten equally-sized ranges and generate 100 bins from their possible combinations.

Ant Maze. A robotic ant should navigate a 'C-shaped' maze, where 7 visitable bins have been identified by heavily discretizing the observation space (Shyam et al., 2019).

Half-Cheetah. Tasks using the HalfCheetah environment usually concern training a bipedal walker robot to learn skills such as walking, running, flipping backward or forward. For this reason, we chose to evaluate exploration performance in terms of x-axis velocity and angular velocity, empirically choosing ranges of [-9.0, 9.0] and [-18.0, 18.0], respectively. Each range is divided into ten smaller subranges and exploration is measured as the percentage of the bins visited among the 100 bins, representing all visitable combinations of subranges.

Model Details. In the LBS modules, the variational layers are implemented as linear layers that output the means and standard deviations of a distribution. Standard deviations are obtained by applying a softplus operator to the outputs of the network. The reparametrization trick is used to propagate gradients through the sampling from the posterior.

For vision-based tasks, observations are first processed by small convolutional networks, which are trained as part of the dynamics and of the posterior models. As for the model's loss, we found $\beta = 0.01$ to be working well across all tasks. We used 512-dimensional hidden layers, latent state and features, for vision-based tasks. For continuous-control experiments, we instead use 32-dimensional hidden layers, latent state and features' dimensionality is the same as the observation's dimensionality.

Normalization. Observation normalization is applied differently for different kinds of environment. In continuous control tasks, observations are normalized by maintaining running statistics. For video games, following the setup in (Burda et al., 2019a), observation's mean and standard deviation values are computed on 10,000 steps collected using a random agent, and are kept fixed during training.

Hyper-parameters. In all experiments, the same learning rate, number of updates and mini-batches have been used both for the policy and the model training. The policy value loss coefficient is fixed to 0.5 and the entropy coefficient to 0.001.

For continuous-control experiments, we used a learning rate of $3e-4$, 10 updates per episode, and 32 minibatches. From one environment, we collected 2048 steps per epoch. For the arcade games experiments, we used a learning rate of $1e-4$, 3 updates per episode, and 8 minibatches. From 128 parallel environments, we collected 128 steps from each, per epoch.

C ADDITIONAL RESULTS

In this Section, we provide the final performance in terms of coverage for the continuous control experiments (Table 1) and the final scores for the video games experiments (Table 2). We also compare to Human performance, reporting the scores in (Mnih et al., 2015), to show that curiosity-driven playing of video games can in some cases be in the same scale with human performance (e.g. BeamRider, Qbert), or even outperform it (e.g. Breakout). It should be also considered that in all Atari games, the max length of an episode was limited to 4500 steps and that performance is shown with respect to 400M frames for training, as in (Burda et al., 2019a). Thus, the scores here shown may likely be increased with longer training or removing the episodic steps limitation.

Table 1: Continuous Control Final Performance

	LBS	ICM	RND	VIME	Random
MountainCar	85.62 ± 2.69	84.5 ± 1.45	50.12 ± 5.31	**87.38 ± 2.24**	15.5 ± 0.55
AntMaze	**100.0 ± 0.0**	**100.0 ± 0.0**	91.07 ± 2.67	94.64 ± 1.86	67.86 ± 2.54
HalfCheetah	**52.67 ± 2.04**	46.67 ± 2.1	25.0 ± 0.59	39.0 ± 0.0	16.67 ± 0.31

Table 2: Arcade Games Final Performance

	LBS	ICM	RF model	VAE model	Human
BeamRider	2659 ± 286	1710 ± 98	1803 ± 251	**3250 ± 89**	5775
Breakout	**359 ± 10**	92 ± 20	165 ± 51	206 ± 20	31.8
MontezumaRevenge	0 ± 0	0 ± 0	**254 ± 180**	87 ± 62	4367
Pong	**-0.8 ± 0.7**	-6.6 ± 4.1	-6.0 ± 0.3	-18.8 ± 3.1	9.3
Qbert	**8866 ± 794**	4692 ± 286	3775 ± 408	2232 ± 226	13455
Riverraid	4711 ± 517	**5296 ± 465**	4499 ± 146	4958 ± 91	13513
Seaquest	**623 ± 31**	530 ± 20	433 ± 66	439 ± 27	20182
SpaceInvaders	534 ± 78	615 ± 41	599 ± 41	**737 ± 28**	1652
Super Mario Bros.	5168 ± 336	4178 ± 175	**5610 ± 470**	5511 ± 277	-

Finally, we report the training curves for the episode lenghts (Figure 8) and the best scores achieved (Figure 9), in all arcade games experiments.

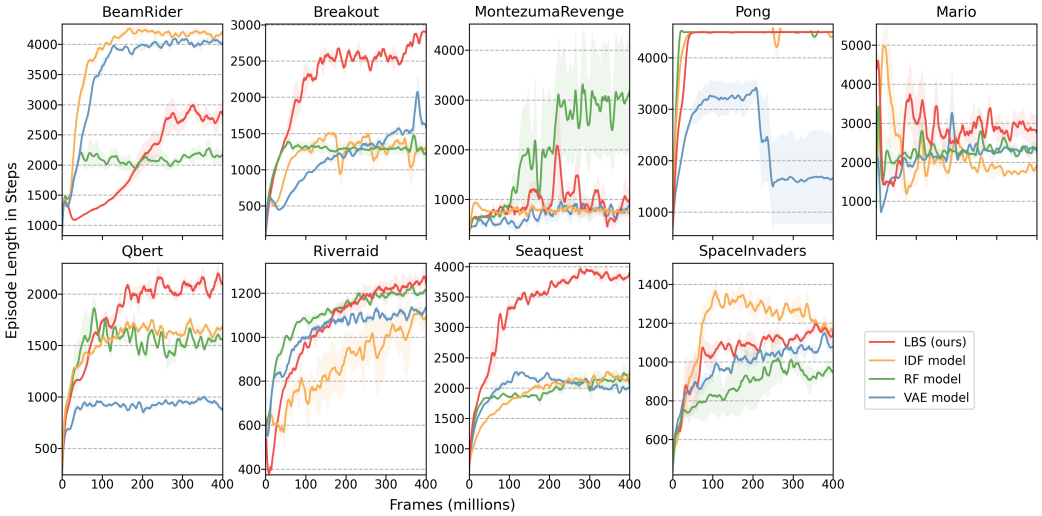


Figure 8: Arcade Games - Episode lengths

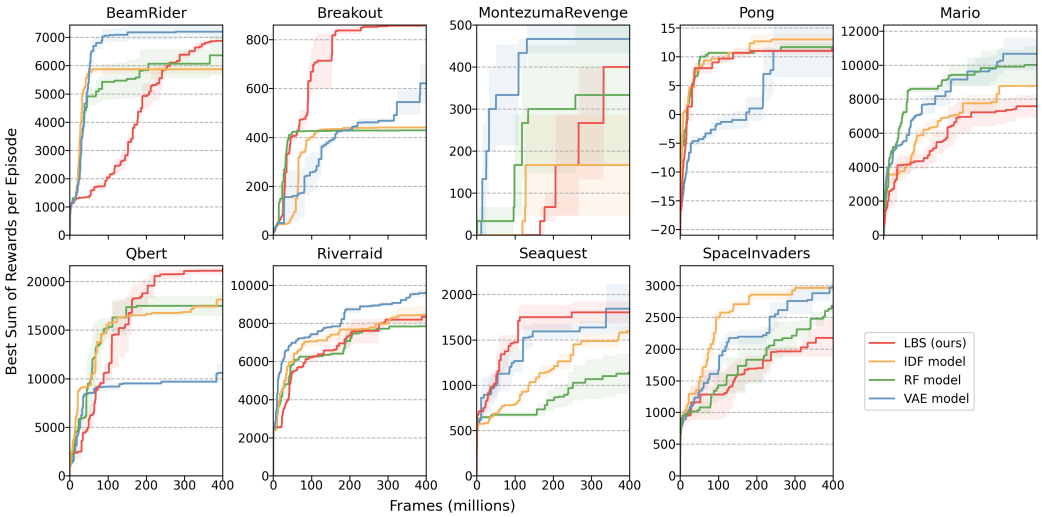


Figure 9: Arcade Games - Best scores