Acting in the Face of Uncertainty

Pessimism in Offline Model-Based Reinforcement Learning

Sten van Wolfswinkel

Supervisor(s): Frans A. Oliehoek, Jinke He

[1]**EEMCS, Delft University of Technology, The Netherlands**

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
June 23, 2024

Name of the student: Sten van Wolfswinkel
Final project course: CSE3000 Research Project
Thesis committee: Frans A. Oliehoek, Jinke He, Mathijs de Weerdt

Abstract

Offline model-based reinforcement learning uses a model of the environment, learned from a static dataset of interactions, to guide policy generation. Sub-optimal planning decisions can be made when the agent explores states that are out-of-distribution, as the world model will have more uncertainty. This paper explores the use of pessimism, the tendency to avoid uncertain states, in the planning procedure. We evaluate Lower Confidence Bound, ensembles, and Monte Carlo dropout in the MinAtar breakout environment. Results indicate that ensemble methods yield the highest performance, with a significant performance gain over the baseline, while LCB also shows varying degrees of improvement. MC dropout is generally shown to not yield a performance improvement.

1 Introduction

Reinforcement learning (RL) is a field with diverse applications in areas such as self-driving cars, healthcare and, recently, natural language processing. Standard online RL methods require the agent to sample interactions directly from the environment. However, this approach can be impractical for many applications, such as self-driving cars, without a highly accurate simulator. This necessitates RL in the offline setting, where the agent learns its policy from a dataset collected a priori. Model-based RL (MBRL) methods aim to further tackle this problem by creating a model of the environment and its interactions [1], which is then used to generate a policy. As the model of the environment is learned from a finite set of interactions, the agent can explore regions of the state-space that were not sufficiently covered in the dataset. In these situations, we expect our model of the environment to be less accurate, which could lead the agent to make sub-optimal decisions based on an imperfect world model. A key idea to compensate for this uncertainty is pessimism: to disincentivise actions that lead the agent to out-of-distribution (OOD) states.

Uncertainty estimation lies at the core of most pessimistic RL algorithms. The most trivial approach to pessimism is the lower confidence bound (LCB) approach [2, 3], which bases its uncertainty directly on how often each data point appears in the dataset. Other approaches use ensembles of neural networks to derive an estimate of uncertainty to enact either a soft penalty [4] or a fixed hard penalty [5]. Additionally, many studies focus on other Bayesian or Bayesian-approximation techniques such as Monte Carlo (MC) dropout [6, 7, 8] or bayesian neural network reparameterisation [9]. Previous research primarily focuses on the model-free Q-learning based setting, thus the goal of this paper is to transfer these ideas and evaluate their performance concretely in the offline model-based setting.

This paper aims to answer the research questions:

- **RQ1: How does incorporating pessimism in the planning loop affect the performance of agents in model-based reinforcement learning?**
- **RQ2: What is the effect of the training data distribution on the performance of pessimism?**

To this end, penalty based pessimism methods, based on LCB, ensemble networks and MC dropout are evaluated in the MinAtar [10] breakout learning environment. Additionally, the effect of the degree of pessimism, as well as the effect of different data distributions in the training procedure are further investigated.

Section 2 covers some preliminaries for the experiments, including a description of the pessimism approaches considered. Section 3 gives an overview of the experimental setup, and Section 4 presents the results. A discussion of the results and findings is found in Section 5. Finally, a discussion of responsible research is found in Section 6, followed by the conclusion and future work in Section 7.

2 Preliminaries

This section covers the relevant background for the problem setting, the different approaches to pessimism that are investigated in this paper, as well as some other related works that are not directly relevant to this paper. Section 2.1 goes over the reinforcement setting relevant to this work, followed by Section 2.2 which details the three pessimism techniques used in this paper. Finally, Section 2.3 briefly goes over other interesting approaches to uncertainty-aware reinforcement learning.

2.1 Reinforcement Learning Setting

RL problems can be framed as Markov Decision Processes (MDPs) defined by (S, A, P, R, γ). Here, S denotes the set of possible states, A denotes the set of possible actions, P denotes the transition dynamics, R denotes the reward function and γ denotes the discount factor. We consider the simplified version of this problem formulation in which the environment is deterministic, i.e. $\exists s_t, a, s_{t+1}, P(s_t, a, s_{t+1}) = 1$. The goal in RL is then to find the policy, $\pi : S \to A$, such that the sum of discounted rewards is maximised.

In MBRL, we aim to learn a dynamics model $S \times A \to S \times R$, which is used in planning to look ahead and estimate future rewards. This is in contrast to the model-free setting, where the cumulative discounted rewards are immediately predicted from the current state, without an intermediate model of the environment. The model architecture and planning procedure used in this paper is based on EfficientZero [11] and Dreamer [12], with several components removed for simplicity. For example, all recurrent components were removed as we assume a fully observable environment where the Markov property holds. Additionally, there is no value function prediction and planning is purely based on immediate reward predictions.

The model architecture consists of a representation network, a dynamics model, and a reconstruction network that drive the planning procedure. The representation network takes a state from the environment and encodes it into a lower dimensional latent space to be used by the dynamics network. The dynamics model is further divided into a dynamics network, which predicts the next latent state from a latent state

and action, and the rewards network, which predicts the immediate reward from a latent state and action. The reconstruction network learns to decode the latent state representation back into the original state space. These networks are then used in a Monte Carlo tree search (MCTS) planning loop to determine the policy. At each decision point, N simulations are performed as part of the MCTS search and the action from the root node which was selected the most is then selected. The model architecture and the relevant algorithms are further detailed in Section 3.

2.2 Pessimism techniques

We now cover the pessimism techniques that are investigated in this paper. The techniques considered mostly fall under the category of penalty-based techniques, where a scalar value, quantifying the uncertainty in the prediction, is subtracted from the predicted reward. Penalty-based pessimism techniques were considered due to their ease of integration with the base architecture. Other pessimistic algorithms exist, such as Conservative Q-Learning [13] and Ensemble-Diversified Actor Critic [14], but are not investigated due to the challenges posed by adapting their frameworks to our setting. Generally, the reward used in planning follows Equation 1, where $p(s, a)$ is the penalty term given state, s, and action, a.

$$\hat{r}(s, a) = r(s, a) - p(s, a) \tag{1}$$

2.2.1 Lower Confidence Bound

The first approach to pessimism that is considered is derived from the LCB approach from Rashidinejad et al. [2]. In this approach, the uncertainty estimate of the dynamics model is derived directly from the count of state-action pairs in the training dataset. This requires the state-action space to be finite, such that the counts of state-action pairs can be practically stored in memory. The premise behind this approach is that we expect state-action pairs that are frequently covered in the training set to be learned more accurately by the dynamics model and thus have lower uncertainty. There are two techniques through which this penalty can be applied; the soft penalty and the hard penalty approach. For the soft penalty, the reward predicted for each state-action pair is penalised according to Equation 2, with penalty, p, and penalty coefficient, β, which corresponds to the degree to which infrequent pairs are penalised.

$$p(s, a) = \frac{\beta}{\sqrt{count(s, a) \vee 1}} \tag{2}$$

Alternatively, the hard penalty approach uses a threshold to penalise state-action pairs with a fixed penalty. This approach uses the penalty term in Equation 3, where k is the fixed penalty term and t is the penalisation threshold.

$$p(s, a) = \begin{cases} k & \text{if } count(s, a) < t \\ 0 & \text{else} \end{cases} \tag{3}$$

2.2.2 Ensembles for Uncertainty Estimation

The ensembles approach to pessimistic reinforcement learning uses an ensemble of N neural networks to aggregate predictions. Each network is designed such that there is diversity between the other networks, so that outputs for any singular input yield different results for each network. This diversity can be achieved in the ensemble through training the networks on different subsets of the training set, using a different order of the training set for each network, or using gradient boosting [15]. Lakshminarayanan et al. have demonstrated the ability of deep ensembles to quantify the uncertainty of their predictions, by utilising the standard deviation of the outputs of the individual networks. Similar to the work in [17], the penalty term for the ensemble approach to pessimism then corresponds to the standard deviation of the outputs, y_{ϕ_i}, of the networks, ϕ_i, as can be seen in Equation 4.

$$p(s, a) = \beta \sqrt{\frac{\sum_{i=1}^{N} (y_{\phi_i} - \bar{y})^2}{N}} \tag{4}$$

2.2.3 Monte Carlo Dropout for Uncertainty Estimation

MC dropout [18] has shown to produce uncertainty estimates that approximate Bayesian methods. This is similar to the uncertainty estimation achieved by deep ensembles, except MC dropout can be implemented trivially with a single neural network with little modification. Usually, dropout is used with neural networks as a regularization technique, where neurons are deactivated with probability, p, replacing their output with 0. This serves to reduce over-fitting by decreasing the reliance on single neurons and diversifying the learning of functions over the whole network. In the context of uncertainty estimation, this idea can be extended by using dropout during inference time as well, referred to as MC dropout, to produce multiple varied outputs for a single input. The same penalty term from the ensemble in Equation 4 is then used during planning.

2.3 Related Works

Various other methods for pessimism and uncertainty quantification exist in addition to the three approaches explored in this paper.

There has been extensive research into pessimism in the model-free RL setting, largely due to Q-learning being prone to overestimation of the value function. Conservative Q-Learning (CQL) [13] is one algorithm that has been proposed to mitigate this bias. This algorithm involves penalizing actions that deviate from what was observed in the training distribution. Ensemble-Diversified Actor Critic (EDAC) [14] uses multiple Q-networks to give pessimistic value estimates. This approach effectively penalizes actions with high variance by using the minimum value from all the networks to update the policy. In addition to these methods, Bootstrapped DQN [19] similarly uses an ensemble of Q-networks trained on bootstrapped subsets of the training set to guide efficient exploration. It utilizes the variability in the outputs of the Q-networks to avoid out-of-distribution actions when learning the policy.

Model-based approaches to pessimistic reinforcement learning also exist. MOReL [20] learns to partition the state-action space into known and unknown regions using an ensemble of models. This partition is then used to heavily penalise actions in the unknown region. This is similar to our hard LCB approach, where we use counts in the training set to split the dataset into known and unknown regions. Uncertainty Weighted Actor-Critic (UWAC) [6] uses dropout to

estimate uncertainty and down-weights uncertain actions during training. This is in contrast to our approach where the uncertainty estimate is directly used to penalise the predicted reward.

The ensemble and MC dropout approaches to uncertainty quantification can be grouped in the class of Bayesian approximation methods. A more concrete quantification of the uncertainty can be achieved with Bayesian Neural Networks (BNNs), which learn weights as a distribution rather than a point value, offering natural uncertainty estimates with predictions. Recent work has explored the application of Bayesian methods in the reinforcement learning setting [21]. Although the use of BNNs for pessimism would be interesting to investigate, they are heavily limited by high training times, thus they will not be explored. Additionally, as ensembles and MC dropout approximate these methods, they are expected to offer similar performance with reduced computation cost. Other methods build on ensemble networks by using bootstrapping for potentially more robust uncertainty estimates [22].

3 Experimental Setup

This section gives an overview of the reinforcement learning environment, the experimental setup, and the results of the experiments. Section 3.1 gives a description of the MinAtar reinforcement learning environment. Section 3.2 details the architecture of the world model used for the policy generation. Section 3.3 gives some details about the training of the model, followed by Section 3.4 which describes the training data used in the experiments. The steps of the policy generation are then outlined in Section 3.5. Section 3.6 notes key details regarding the pessimism approaches implementation.

3.1 MinAtar

The experiments are conducted within the MinAtar breakout environment [10]. This is due to the lower computational demand from the smaller state-space size, allowing for more rapid experimentation. Additionally, the open source nature of MinAtar allows for easier reproducibility of the experiments within this paper. In the experiments, sticky actions are not used to maintain a deterministic environment without stochasticity. Figure 1 shows an example image from the MinAtar environment.

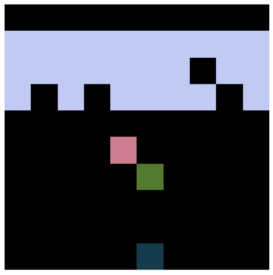

Figure 1: A frame from the MinAtar breakout environment.

3.2 World Model Architecture

The world model aims to learn the interactions in the environment to ultimately predict the next sequences of rewards given a state and actions. To this end, the model consists of a representation network, dynamics model, and reconstruction network. The representation network is a convolutional neural network which takes a state and outputs its lower dimensional latent representation. The dynamics model consists of the dynamics, reward and done sub-networks. The dynamics network predicts the next latent state given a latent state and action, while the reward and done networks predict the reward and termination given a latent state. The termination value is used to prune actions that are expected to end the game. The reconstruction network then takes a latent state and reconstructs the original state. Note that the reconstruction network is never actually used in the policy generation, yet is used to ensure the world model learns a meaningful latent state representation which maps to the original state space. This is also useful for visualising the latent states when debugging. This can optionally be left out of the model architecture, as this does not guarantee performance benefits. A detailed diagram of the model architecture including the individual network architectures can be found in Figure 2.

The objective loss function that is learned is shown in Equation 5. Let z_ϕ be the representation network, f_ϕ be the dynamics network, d_ϕ be the done network, r_ϕ be the reward network, and x_ϕ be the reconstruction network. The coefficients in front of the the individual loss components are tunable parameters, as some of the components are conflicting objectives.

$$\begin{aligned}\mathcal{L} = E\,[&\text{MSE}(s_t, x_\phi(z_\phi(s_t))) + \text{MSE}(r_t, r_\phi(z_\phi(s_t)))\\ &+\text{BCE}(d_t, d_\phi(z_\phi(s_t)))\\ &+0.5 \cdot \text{MSE}(z_\phi(s_{t+1}), f_\phi(z_\phi(s_t), a_t))\\ &+0.1 \cdot \text{MSE}(f_\phi(z_\phi(s_t), a_t), z_\phi(s_{t+1}))]\end{aligned} \tag{5}$$

3.3 Training and Performance

The training and model evaluation were performed on the Delft High Performance Computing (DHPC) cluster [23]. Model training was performed over 15 epochs with mini batches of size 128 and took four hours on an NVIDIA A100 GPU. Model evaluation was parallelized and performed on 16 cores on an Intel Xeon E5-6248R CPU, taking roughly an hour for each evaluation. For each method, three models were trained with different random seeds, and the results were averaged over the different seeds. For the experiments on the effect of the data distribution, only one model was trained for each approach and distribution due to time constraints.

3.4 Data Generation

The world model was trained on 5M frames of interactions in the MinAtar environment. The trajectories are tuples containing the state, action, reward, and whether the action terminated the game, referred to as the "done value". These trajectories were collected using a near optimal Deep Q-Network (DQN) following an epsilon-greedy action selection strategy. The strategy uses an annealed epsilon value which starts at

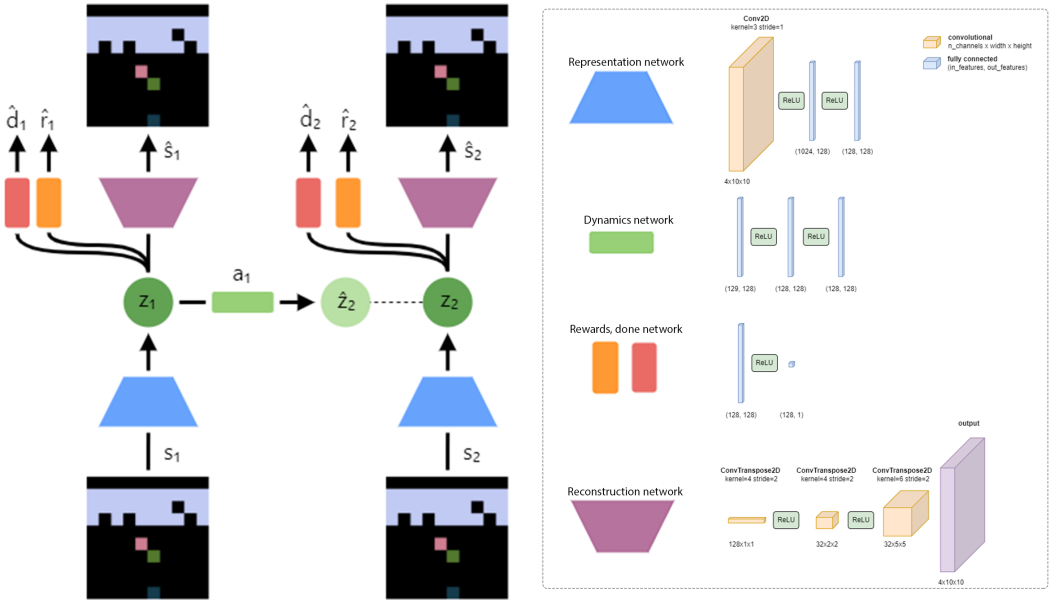

Figure 2: The architecture of the world model. The blue encoder represents the representation network. The encoded states, z_i, are then used to predict $\hat{r}_i$ by the reward network, the termination $\hat{d}_i$ by the done network and the reconstructed $\hat{s}_i$ by the reconstruction network. The dynamics network takes z_i and predicts the next z_{i+1}. The reconstruction network takes z_i and predicts the corresponding state $\hat{s}_i$ in the original state space. On the right, the individual model architectures can be seen.

1.0 and ends at 0.1, linearly transitioning over the first 100k frames. For the later experiments, which investigate the effect of different data distributions on pessimism, this same DQN was used to collect 5 million frames with varying epsilon values without an annealing strategy.

3.5 Model Evaluation

The learned world model must then be used to guide the action selection of the agent, such that the world model can be evaluated in terms of cumulative undiscounted return. To this end, a Monte Carlo Tree Search (MCTS) is performed from each environment state, using the dynamics model to determine future latent states and expected rewards. This process is shown in Algorithm 1. A standard MCTS implementation is used with UCB1 [24], 128 simulations, an exploration factor of 1.0, a discount factor of 0.97 and a planning horizon of 32 to avoid accumulating errors. Additionally, a form of early branch pruning is performed to avoid states which have likely led to the end of a game. A cumulative continuation probability value is maintained for each state, corresponding to the multiplication of the state's last cumulative continuation probability with the compliment of the predicted done value in [0,1]. When the cumulative continuation probability goes below 10^{-6}, the branch is pruned. Finally, the action from the root node with the highest visit count, rather than the highest expected value, is chosen by the agent according to [25]. Iteratively performing the MCTS results in a cumulative return value for each world model. The model evaluation is then performed 100 times with different randomisation seeds, and the returns are averaged for an aggregate result.

This is also the stage where pessimistic penalties are enacted. In the MCTS, predicted rewards returned by the world model are augmented with the penalty associated with each state and action, as shown in Equation 1.

Algorithm 1 Model Evaluation

Input: env, model
$return \leftarrow 0$
$done \leftarrow$ **false**
while not $done$ **do**
  $z \leftarrow$ model.representation_net(env)
  $best\_action \leftarrow$ MCTS(z, model)
  $reward, done \leftarrow$ env.act($best\_action$)
  $return \leftarrow return + reward$
  return return
end while

3.6 Pessimism Experiment Details

The following section provides important details of the experiments conducted for each of the pessimism techniques.

3.6.1 LCB

For the LCB method, counts of state-action pairs in the training dataset must be retrievable. For this, an in-memory hash map is maintained containing the counts. This bring rise to multiple important considerations. Firstly, the entire training set of states and actions must be stored in memory; for large training sets or larger state spaces, this approach may no longer be feasible. Secondly, the state-action pairs must be discrete for a hash map of counts to be plausible. Lastly, the MCTS used for model evaluation encodes states using their latent representation, rather than the original representation which the counts are associated with. There are 2 con-

sidered approaches to retrieving the original representation for each latent state. The first approach is to use the reconstruction network to transform the latent state into the original state space. As the reconstruction network is not perfect, this leads to losses due to imperfect reconstructions and transitions of latent spaces, which is an even larger problem at deeper branches of the MCTS due to compounding errors. A solution to this is using the original environment to simulate the actions and retrieve the original states that would be reached by taking actions. In a real RL setting, the agent would not have access to this environment, however the latter approach is used to give an upper bound on the performance of LCB. We experiment with both a soft and a hard penalisation approach. The former uses Equation 2 to calculate the penalty while the latter uses Equation 3. Additionally, we experiment with a variation of the hard approach where state-action pairs below the threshold are not able to be visited at all.

3.6.2 Ensemble

In the ensembles approach, we use an ensemble of network and combine their results to produce uncertainty estimates. The experiments for the ensemble approach use a configuration with an ensemble size of 5. Network diversity is important for ensembles to have robust learning and meaningful uncertainty estimates. The networks are each trained on the entire train set, differing only in their random initialisations and order of mini-batches. This is contrast to the common method of splitting the dataset into even and independent partitions, as experiments showed that training on the entire dataset gave higher performance. This is also the approach used by [16]. The MCTS algorithm must be modified here to maintain a list of the latent states for each network in the ensemble. The results are only aggregated when the reward is predicted from the latent states. The uncertainty estimate used as the penalty then corresponds to the standard deviation in the different rewards predicted by each of the ensembles.

3.6.3 MC Dropout

In the MC Dropout approach, dropout layers are used in the network to perform multiple stochastic forward passes during inference to estimate uncertainty. Specifically, dropout rates of 0.01 and 0.05 are used to explore the effect of different dropout rates on model performance. Unlike the ensemble method, the uncertainty estimation for MC dropout can also use the standard deviation in the latent space rather than the reward predictions. This could improve performance based on the hypothesis that applying dropout in the dynamics network offers more meaningful and informative uncertainty estimates than the reward network. An explanation for this is that the latent space is expected to contain more information than a single reward estimate. Therefore, we test MC dropout with uncertainty estimates for both the latent space and reward prediction. The standard deviation of 10 forward passes is then used as the uncertainty measure.

4 Results

This section presents the results of the experiments. Section 4.1 presents the results of the different pessimism approaches on the standard dataset, corresponding to **RQ1**. Section 4.2 then shows the results of the experiments repeated over multiple data distributions collected over different epsilon-greedy policies, which corresponds to **RQ2**.

4.1 Evaluating Pessimism

This section covers the results for **RQ1**:

How does incorporating pessimism in the planning loop affect the performance of agents in model-based reinforcement learning?.

The performance of each of the approaches can be seen in Table 1. The ensembles are shown to have the highest performance of all the approaches, while the LCB method also achieves a score greater than the baseline. The MC dropout method achieves performance lower than the baseline non-pessimistic agent.

Model	Score	Standard Error
Baseline	6.28	0.44
LCB	9.89	0.57
Ensemble	10.59	1.58
MC Dropout	5.61	0.34

Table 1: Performance comparison of different models. The value for LCB is achieved with the hard constraint approach. MC Dropout uses a dropout rate of 0.05 and uncertainty on the reward prediction. The standard errors correspond to three trials with different random seeds.

4.1.1 LCB

The soft penalty LCB approach, from Equation 2 does not improve the baseline model performance. The scores for different penalty coefficients can be seen in Figure 3. A large penalty coefficient is shown to reduce the performance of the agent to 0.

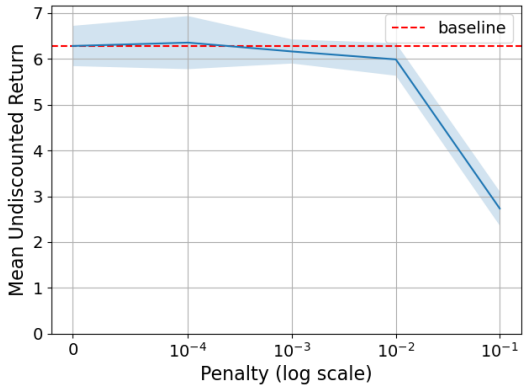

Figure 3: The performance of soft penalty LCB for different penalty coefficients.

Following from the soft penalty LCB approach, we experiment with the hard penalty approach that uses Equation 3. The results for different thresholds and penalties are shown in Figure 4. This approach yields similar performance to the soft penalty approach. These results indicate that the count based

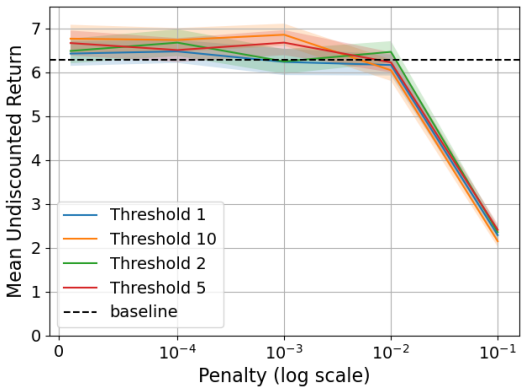

Figure 4: The performance of hard penalty LCB for different penalties and thresholds.

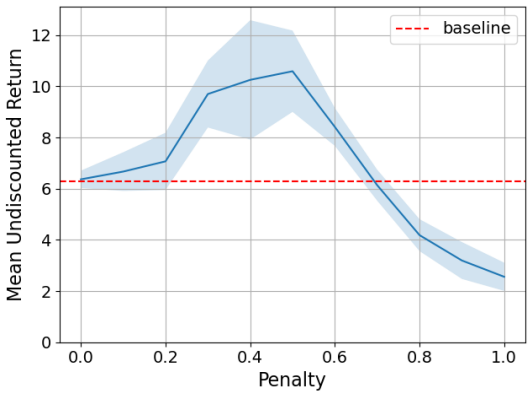

Figure 6: The performance of the ensemble for different penalties.

uncertainty estimates, used by the penalties, do not provide any utility for the agent to explore safer states.

As the performance of the thresholded penalty was shown to not yield better results, an alternate approach was tested with a hard constraint rather than a penalty. In the new approach, state-action pairs with counts below the threshold can no longer visited, restricting the search to state-action pairs in the training set. The results of this method are shown in Figure 5. This method achieves a score of 9.89, which is the best from the LCB approaches, and is significantly higher than the baseline. This indicates that for the dataset used, the counts are able to be used to guide safer action selection. However, this result is highly dependent on the dataset used, as to achieve runs with with high scores, the model must have seen those episodes with high scores in the training set.

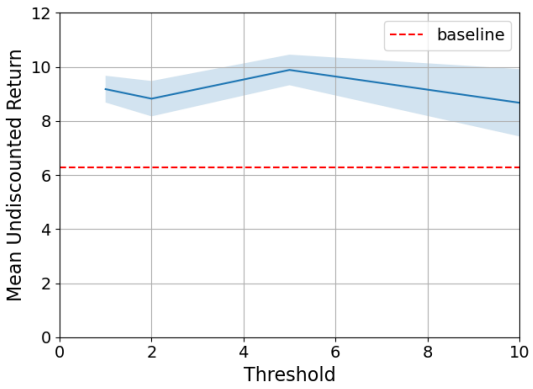

Figure 5: The performance of hard constrained LCB for different thresholds. No penalty is enacted but branches with counts lower than the threshold are not explored.

4.1.2 Ensemble

The results shown in Figure 6 show the mean undiscounted return for various penalty values. As seen in the figure, the ensemble method achieves the highest mean undiscounted reward at a penalty value of 0.5, significantly outperforming the baseline model. The performance substantially decreases as the penalty value varies beyond this point, indicating an optimal penalty value for the ensemble method. Increasing the penalty too high results in a return of 0.

The use of multiple neural networks for uncertainty quantification leads to robust results as the uncertainty is shown to be a Bayesian uncertainty approximation [16], without a trade-off in model performance. As ensembles use multiple diverse networks, it is expected that their expressive capability is higher than standard neural networks, so it is naturally expected that their performance is higher than standard networks. However, the experiments showed that the increase in performance is not attributed to the increase in expressive power, as the increase in performance was only observed after enacting the penalty.

4.1.3 MC Dropout

Figure 7 shows the results for MC Dropout for dropout rates of 0.01 and 0.05, and uncertainty estimates for both the latent space and reward value. A dropout rate of 0.05 on the dynamics network, corresponding to the uncertainty in the latent space, is shown to significantly reduce performance compared to the other settings. This indicates that the dynamics network contains most of the complexity required for effective planning, and high dropout rates on the dynamic network lead to a high loss in expressive capability. Using dropout on the reward network is shown to have a lower drop in planning performance. However, for all approaches, enacting a penalty based on the MC dropout uncertainty estimates is not shown to improve performance, in contrast to the ensemble method.

4.2 Effect of Data Distribution

This section covers the results for **RQ2**:

What is the effect of the training data distribution on the performance of pessimism?.

The experiments are repeated for different data distributions to examine how effective the approaches are under different conditions. We test the approaches with 6 new datasets, generated by changing the epsilon value in the epsilon-greedy action selection strategy. The datasets range from mostly optimal actions to mostly random actions, corresponding to an epsilon value of 0 and 1 respectively. The results of the experiments are shown in Figure 8.

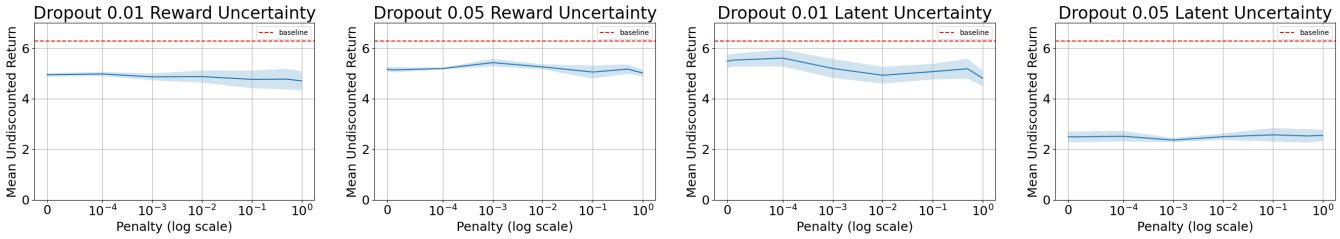

Figure 7: The performance of MC Dropout for different dropout rates and uncertainty sources. The performance in each setting is shown over a range of penalty coefficients.

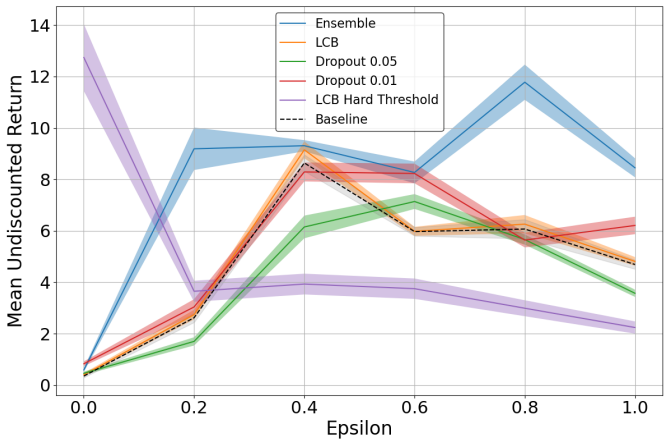

Figure 8: The performance of each of the pessimism approaches for models trained on data distributions with varying epsilons for the epsilon-greedy action selection strategy. The MC dropout method uses uncertainty on the reward value, as this was shown to give the highest return in the previous experiments.

Hard-constrained LCB is shown to perform the best out of all methods when the dataset used is entirely optimal (epsilon 0 in epsilon-greedy). This is expected as this approach effectively memorises the trajectories seen in the dataset, so a dataset of a single perfect trajectory will always result in the maximum score. The performance of this method significantly drops for other values of epsilon. In these cases, the model is forced to stay within the dataset it was trained on. The drop in performance suggests that the baseline model is able to generalise well to unseen states, as limiting the search to seen states leads to a large drop in performance.

The ensemble method is shown to perform the best out of all methods for the other data distributions, with its peak performance at an epsilon of 0.8. This is surprising, as an epsilon of 0.8 corresponds to taking mostly random actions, thus the dataset contains mostly suboptimal trajectories.

MC Dropout also generally under performs compared to the baseline, but it is able to outperform the baseline for some values of epsilon. The dropout rate of 0.01 is also shown to out-perform the dropout rate of 0.05 for all data distributions, due to a loss in expressive capability of the networks for higher dropout rates.

5 Discussion

The findings indicate that ensemble methods provided the highest performance boost, significantly outperforming the baseline model. The hard constrained LCB method also showed improvement but was highly sensitive to the training data distribution. MC dropout generally did not improve performance and often underperformed compared to the baseline.

The LCB method's moderate improvement highlights the utility of count-based penalties for steering the agent towards more promising states. However, it is then surprising that the penalty based LCB approaches did not provide any improvement in performance, as the penalty-based approaches implicitly aim to achieve the same effect as the hard constraint. The need for the entire dataset to be in a tabular form and stored in memory presents practical limitations for LCB's applicability in larger and more complex environments.

The performance of ensemble methods can be attributed to their ability to robustly quantify uncertainty without sacrificing the expressiveness of the network. This is in contrast to MC dropout, where the uncertainty quantification effectively results in a loss in model complexity, as the network is expected to have p neurons deactivated during inference. However, it is still unexpected that using the uncertainty estimates in MC dropout does not result in better performance, even when taking into account the lost model complexity. It is possible that these results would be different for larger models and potentially different dropout rates. Work from Osband et al. has shown that MC dropout can provide a poor approximation of the Bayesian posterior [26]. This is likely why the uncertainty estimates provided by MC dropout do not guide the agent to make better decisions.

The results have shown the potential of ensemble networks to be used for robust planning for MBRL agents. The ensemble networks were shown to perform well under a wide range of data distributions, and the ability of ensembles to perform well with suboptimal data highlights their potential in real life applications where data is rarely optimal.

6 Responsible Research

This section reflects on the reproducibility and integrity of the research conducted in this study. The reproducibility of the research was addressed in multiple ways. Firstly, the MinAtar [10] breakout environment was used as a computationally inexpensive and open-source RL environment, to allow for easy replication. Additionally, key architectural and

algorithmic aspects of the research are well documented in this paper. The model architectures, datasets, planning algorithms, and hyperparameters are thoughtfully reported to allow the experiments to be reproduced as closely as possible. In other areas where standard algorithms such as MCTS are used, only the algorithms parameters are reported for the sake of brevity. Lastly, the code used for this paper can be found in **this github repo.**

The experiments were performed on the Delft High Performance Computing cluster [23], and Section 3.3 outlines the hardware used and time taken for the experiments. This can be used as a reference for other researchers when reproducing the experiments, to help them with the correct allocation of resources.

With respect to the integrity of this research, it is important that the results presented are accurate. To this end, all results are presented with their standard errors. However, this does not eliminate all sources of inaccuracy; there are several sources of stochasticity in training neural networks, such as the order of batches and random initialisation. To further improve the accuracy of the results, the experiments were repeated multiple times with three random seeds. This was not done for the experiments on the effect of the data distribution as this was infeasible due to the high time and resource requirements of training and evaluating the models. Improved integrity of this research could be achieved by also repeating the experiments on the effect of the data distribution.

Any negative ethical implications of the research should also be considered. In this case, any improvements to RL models can also be applied to malicious RL models, making their threat larger. However, the risk here is determined to be limited, as this research focuses on the comparison of already existing methods.

7 Conclusions and Future Work

This research investigated the effect of different pessimism techniques for offline model-based reinforcement learning agents. To mitigate the effect of overestimation bias for states that are not sufficiently covered in the training dataset of the world model, pessimism aims to steer the agent towards more in-distribution states. To this end, three pessimism techniques, Lower Confidence Bound (LCB), ensembles, and Monte Carlo (MC) dropout, were evaluated in the MinAtar breakout environment. An additional goal of this paper was to evaluate the different techniques under different data distributions.

The LCB method showed a moderate increase in performance compared to the baseline. LCB was tested in multiple configurations: using a soft penalty, thresholded penalty and a hard constraint (where low-count state-action pairs are not visited). The soft and thresholded penalty showed no increase in performance, while the hard constraint yielded a more significant increase in peformance. However, the hard constraint approach was highly influenced by the data distribution, with near-optimal datasets required to achieve good performance. Additionally, this method requires the entire dataset to be tabular and to be stored in memory, which is infeasible for many applications. LCB was evaluated using a perfect world model to retrieve state-action pair counts, thus this result is an upper bound on what can be achieved in practical settings.

The ensembles achieved the highest performance among the evaluated techniques. The use of multiple neural networks allowed for robust and useful uncertainty quantification without a trade-off in network expressiveness. Interestingly, ensembles were shown to perform the best with relatively suboptimal datasets (epsilon 0.8).

MC dropout generally underperformed compared to the baseline model. The uncertainty quantification using dropout at inference did not provide any utility during planning.

Future research could focus on new adaptive methods for automated optimisation of the penalty coefficients. All penalty based methods were shown to be highly sensitive to the chosen penalty coefficient. Improvements in this area could yield more robust performance under a wider range of conditions.

These methods could also be applied to more complex environments to provide more insights into their generalisability. Applying these techniques to real-world applications, such as autonomous driving or healthcare, where offline data is common, could provide new insights into their suitability. Considering safety, these experiments should be in simulated environments.

Additionally, the effect of the size of the models could also be investigated. MC dropout provides uncertainty estimates at the expense of worse model capacity. It is possible that increasing the model size improves the performance of the dropout network without pessimism, such that the penalties can be used to achieve better performance than the baseline. The decomposition of the uncertainty estimate into epistemic and aleatoric uncertainty may also provide a more useful uncertainty quantification.

References

[1] Sinan Çalışır and Meltem Kurt Pehlivanoğlu. Model-free reinforcement learning algorithms: A survey. In *2019 27th Signal Processing and Communications Applications Conference (SIU)*, pages 1–4, 2019. doi: 10.1109/SIU.2019.8806389.

[2] Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: A tale of pessimism. (arXiv:2103.12021), July 2023. URL http://arxiv.org/abs/2103.12021. arXiv:2103.12021 [cs, math, stat].

[3] Laixi Shi, Gen Li, Yuting Wei, Yuxin Chen, and Yuejie Chi. Pessimistic q-learning for offline reinforcement learning: Towards optimal sample complexity. (arXiv:2202.13890), June 2022. doi: 10.48550/arXiv.2202.13890. URL http://arxiv.org/abs/2202.13890. arXiv:2202.13890 [cs, stat].

[4] Jordi Smit, C. T. Ponnambalam, M. T. J. Spaan, and F. A. Oliehoek. Pebl: Pessimistic ensembles for offline deep reinforcement learning. *Robust and Reliable Autonomy in the Wild Workshop at the 30th International Joint Conference of Artificial Intelligence*, 2021. URL https://repository.tudelft.nl/islandora/object/uuid%3A2053b579-a663-4def-ad25-4bedad0169be.

[5] Bo Hu, Yang Xiao, Sunan Zhang, and Bocheng Liu. A data-driven solution for energy management strategy of hybrid electric vehicles based on uncertainty-aware model-based offline reinforcement learning. *IEEE Transactions on Industrial Informatics*, 19(6):7709–7719, 2023. doi: 10.1109/TII.2022.3213026.

[6] Yue Wu, Shuangfei Zhai, Nitish Srivastava, Joshua Susskind, Jian Zhang, Ruslan Salakhutdinov, and Hanlin Goh. Uncertainty weighted actor-critic for offline reinforcement learning. (arXiv:2105.08140), May 2021. doi: 10.48550/arXiv.2105.08140. URL http://arxiv.org/abs/2105.08140. arXiv:2105.08140 [cs].

[7] Takuya Hiraoka, Takahisa Imagawa, Taisei Hashimoto, Takashi Onishi, and Yoshimasa Tsuruoka. Dropout q-functions for doubly efficient reinforcement learning. (arXiv:2110.02034), March 2022. doi: 10.48550/arXiv.2110.02034. URL http://arxiv.org/abs/2110.02034. arXiv:2110.02034 [cs].

[8] Thomas M. Moerland, Joost Broekens, and Catholijn M. Jonker. Efficient exploration with double uncertain value networks. (arXiv:1711.10789), November 2017. doi: 10.48550/arXiv.1711.10789. URL http://arxiv.org/abs/1711.10789. arXiv:1711.10789 [cs, stat].

[9] Charles Blundell, Julien Cornebise, Koray Kavukcuoglu, and Daan Wierstra. Weight uncertainty in neural networks. (arXiv:1505.05424), May 2015. doi: 10.48550/arXiv.1505.05424. URL http://arxiv.org/abs/1505.05424. arXiv:1505.05424 [cs, stat].

[10] Kenny Young and Tian Tian. Minatar: An atari-inspired testbed for thorough and reproducible reinforcement learning experiments, 2019. URL https://arxiv.org/abs/1903.03176.

[11] Weirui Ye, Shaohuai Liu, Thanard Kurutach, Pieter Abbeel, and Yang Gao. Mastering atari games with limited data. *CoRR*, abs/2111.00210, 2021. URL https://arxiv.org/abs/2111.00210.

[12] Danijar Hafner, Timothy P. Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. *CoRR*, abs/2010.02193, 2020. URL https://arxiv.org/abs/2010.02193.

[13] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. (arXiv:2006.04779), August 2020. doi: 10.48550/arXiv.2006.04779. URL http://arxiv.org/abs/2006.04779. arXiv:2006.04779 [cs, stat].

[14] Gaon An, Seungyong Moon, Jang-Hyun Kim, and Hyun Oh Song. Uncertainty-based offline reinforcement learning with diversified q-ensemble. (arXiv:2110.01548), October 2021. doi: 10.48550/arXiv.2110.01548. URL http://arxiv.org/abs/2110.01548. arXiv:2110.01548 [cs].

[15] Jerome H. Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29(5):1189–1232, October 2001. ISSN 0090-5364, 2168-8966. doi: 10.1214/aos/1013203451.

[16] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. (arXiv:1612.01474), November 2017. doi: 10.48550/arXiv.1612.01474. URL http://arxiv.org/abs/1612.01474. arXiv:1612.01474 [cs, stat].

[17] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization. (arXiv:2005.13239), November 2020. doi: 10.48550/arXiv.2005.13239. URL http://arxiv.org/abs/2005.13239. arXiv:2005.13239 [cs, stat].

[18] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. (arXiv:1506.02142), October 2016. doi: 10.48550/arXiv.1506.02142. URL http://arxiv.org/abs/1506.02142. arXiv:1506.02142 [cs, stat].

[19] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. (arXiv:1602.04621), July 2016. doi: 10.48550/arXiv.1602.04621. URL http://arxiv.org/abs/1602.04621. arXiv:1602.04621 [cs, stat].

[20] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel: Model-based offline reinforcement learning. (arXiv:2005.05951), March 2021. doi: 10.48550/arXiv.2005.05951. URL http://arxiv.org/abs/2005.05951. arXiv:2005.05951 [cs, stat].

[21] Mohammad Ghavamzadeh, Shie Mannor, Joelle Pineau, and Aviv Tamar. Bayesian reinforcement learning: A survey. *Foundations and Trends® in Machine Learning*, 8(5–6):359–483, 2015. ISSN 1935-8237, 1935-8245. doi: 10.1561/2200000049. arXiv:1609.04436 [cs, stat].

[22] Aviral Kumar, Justin Fu, George Tucker, and Sergey Levine. Stabilizing off-policy q-learning via bootstrapping error reduction. (arXiv:1906.00949), November 2019. doi: 10.48550/arXiv.1906.00949. URL http://arxiv.org/abs/1906.00949. arXiv:1906.00949 [cs, stat].

[23] Delft High Performance Computing Centre (DHPC). DelftBlue Supercomputer (Phase 2). https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2, 2024.

[24] Edward J. Powley, Daniel Whitehouse, and Peter I. Cowling. Bandits all the way down: Ucb1 as a simulation policy in monte carlo tree search. In *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, pages 1–8, 2013. doi: 10.1109/CIG.2013.6633613.

[25] Cameron B. Browne, Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. A survey of monte carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games*, 4(1):1–43, 2012. doi: 10.1109/TCIAIG.2012.2186810.

[26] Ian Osband, John Aslanides, and Albin Cassirer. Randomized prior functions for deep reinforcement learning. (arXiv:1806.03335), November 2018. doi: 10.48550/arXiv.1806.03335. URL http://arxiv.org/abs/1806.03335. arXiv:1806.03335 [cs, stat].