08 - You Play Ball, I Play Ball:
Bayesian Multi-Agent Reinforcement Learning for Slime Volleyball

New Jun Jie, Stephen Tan Hin Khai, Lim Jun Hup, Aakanksha Rai, Chandrasekaran Hema, Raivat Shah
Email: {e0389098, e0389094, e0325362, e0313710, e0273869, e0315854} @u.nus.edu
School of Computing, National University of Singapore

Abstract
In Slime Volleyball, a two-player competitive game, we investigate how modelling uncertainty improves AI players' learning in 3 ways: against an expert, against each other and against itself, in the domain of multi-agent reinforcement learning (MARL). We show that by modelling uncertainty, Bayesian methods improve MARL training in 4 ways: performance, training stability, uncertainty and generalisability, at the relatively small trade-off of sample efficiency, and through experiments using TensorFlow Probability and Stable Baselines, we present interesting differences in agent behaviour. We contribute 3 code functionalities: Bayesian methods using Flipout integrated into Stable Baselines, multi-agent versioned learning framework for Stable Baselines (previously with only single-agent support) and uncertainty visualisation using agent clones for Slime Volleyball Gym, available at: *github.com/jetnew/slimerl*.

1 Introduction
Deep reinforcement learning (RL) has recently shown promising results in multi-agent gameplay [1,2]. However, several challenges remain open research problems, such as the generalisability of RL agents across unseen environments [3]. Bayesian methods are increasingly applied to many machine learning domains because of its capacity to quantify uncertainty, including reinforcement learning [4]. In the context of improving generalisability, it seems a natural complement to multi-agent reinforcement learning (MARL). Therefore, the aim of the project is to investigate the extent to which Bayesian methods can alleviate the difficulties in multi-agent reinforcement learning. Our research project shows that Bayesian methods, in particular the Flipout layer, improves multi-agent training of PPO in 4 ways: performance, training stability, uncertainty and generalisability, at the relatively small trade-off of sample efficiency.

2 Related Work
Reinforcement Learning in Games
Reinforcement learning in games has shown much development and potential in achieving new advancements in the fields of decision making and problem solving in controlled environments, opening new applications and research areas for study, shown in the improved generalisability of artificial agents in solving game environments of high complexity [5]. Aside from games, reinforcement learning has proven successful in real world applications, such as the design of algorithms to generate robust trading signals in financial markets [6] and learning behaviour in robotics [7].

Bayesian Reinforcement Learning
Bayesian methods in reinforcement learning are known to provide evaluation metrics to quantify action-selection as a function of uncertainty in learning, and acts as a way to incorporate a priori information into the algorithm [4].

$$P(\theta|x) = \frac{P(x|\theta)}{P(x)}P(\theta)$$

The Bayesian posterior provides a principled way to capture the full state of knowledge, and defines measures of uncertainty. Through explicit modeling of the distribution over unknown system parameters, Bayesian methods can be applied to handle parameter uncertainty. Moreover, the probabilistic nature of Bayesian RL gives it a benefit to be able to function and work well with relatively much less information, and therefore more uncertainty. Hence the Bayesian approach offers a compelling approach to RL problems, providing new tools to analyse neural network models in model-free Bayesian reinforcement learning.

Multi-Agent Reinforcement Learning
MARL problems address the sequential decision-making problem of multiple autonomous agents operating in a common environment, each aiming to optimise its own long-term returns by interacting with the environment and other agents. MARL is involved with many challenges, such as optimising an agent's objective with respect to other agents optimising their own objectives. As multiple agents are optimising for their own objectives concurrently, the environment is nonstationary, and theoretical analyses in classical reinforcement learning settings may no longer apply [8].

3 Background
Competitive Multi-Agent Reinforcement Learning
Multi-agent reinforcement learning addresses sequential decision making problems involving more than one agent, a generalization of the Markov decision process (MDP) to multiple agents as a Markov game, where each agent optimizes its own value function $V^{i}_{\pi^{i},\pi^{-i}} = E[\sum_{t\geq 0}\gamma^{t}R^{i}(s_t, a_t, s_{t+1})|a^{i}_{t} \sim \pi(\cdot|s_t), s_0 = s]$. The competitive setting is modelled as a zero-sum Markov game [8], where $\sum_{i\in N} R^{i}(s,a,s') = 0$ for any (s,a,s') where N is the number of agents.

Policy Gradients
Policy gradients is a policy optimization method by gradient ascent of $\theta_{k+1} = \theta_k + a\nabla_\theta V(\pi_\theta)|_{\theta_k}$, where the expected return $V(\pi_\theta) = E[R(\tau)]$ is maximised, using the gradient of policy performance $\nabla_\theta V(\pi_\theta)$ [9]. Therefore, the value function that measures the value of an agent's policy that maps a state to an action is given by:

$$\nabla_\theta V^{\pi}(s) = \sum_{a\in A}(\nabla_\theta \pi_\theta(a|s)Q^{\pi}(s,a) + \pi_\theta(a|s)\sum_{s'}P(s'|s,a)\nabla_\theta V^{\pi}(s'))$$

Proximal Policy Optimization (PPO)
Proximal Policy Optimization (PPO) is a policy gradient method that takes the largest update to improve performance while satisfying a constraint that defines how close new and old policies are. The PPO-Clip variant uses a clipping of the objective function in place of the Kullback Leibler divergence constraint. PPO-Clip updates policies with $\theta_{k+1} = arg\,max_\theta E_{s,a\sim\pi_{\theta_k}}[L(s,a,\theta_k,\theta)]$, where $L(s,a,\theta_k,\theta) = min(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a), clip(\frac{\pi_\theta(a|s)}{\pi_{\theta_k}(a|s)}A^{\pi_{\theta_k}}(s,a), 1-\epsilon, 1+\epsilon)A^{\pi_\theta}(s,a))$ enforcing $r(\theta)$ within a interval around $1 \pm \epsilon$ to control policy updates. PPO is an effective policy-based method to achieve good benchmark results [10].

Bayesian Neural Networks and Variational Inference.
Bayesian neural networks (BNN), in contrast to deep neural networks (DNN), determines neural network weights using maximum likelihood estimation to compute the maximum a priori estimate of weights using variational inference by minimizing the Kullback-Leibler divergence between two probability distributions, deriving the evidence lower bound:

$$F(D|\theta) \approx \frac{1}{N}\sum_{i=1}^{N}[log\,q(w^{(i)}|\theta) - log\,p(w^{(i)}) - log\,p(D|w^{(i)})],$$

serving a regularizing effect and reducing tendency to overfit [11].

Flipout for Bayesian Neural Networks

Bayesian neural networks can be trained with variational inference by perturbing the weights. Weight perturbation samples weights of a neural network stochastically. Flipout implements the Bayesian neural network by applying a base perturbation $\Delta\hat{W}$ and multiplying it by a sign matrix $\Delta W_n = \Delta\hat{W} \cdot r_n s_n^T$ [12]. Flipout computes weight perturbations quasi-independently using mini-batches, reducing the high variance of the gradient estimation which many other weight perturbation algorithms suffer. Flipout estimators approximate the posterior distribution by integrating over the kernel and bias, enabling variational inference to approximate the posterior distribution [12].

Slime Volleyball Gym

The Slime Volleyball gym is an environment for testing single and multi-agent reinforcement learning algorithms [13]. Each agent begins with 5 lives, and its goal is to get the ball to land on the opponent's side, scoring a reward of +1, and -1 when the agent loses, until one of the agents loses all 5 lives or after 3000 timesteps, with a reward range of $[-5, 5]$ per episode. The 12 dimensional observation space of an agent consists of the x,y-coordinates and x,y-velocities of the agent, ball and its opponent.

The Slime Volleyball gym is an environment with relatively straightforward dynamics with multi-agent support, allowing us to compare between agents with a DNN and BNN policy, investigating the influence of Bayesian methods on multi-agent reinforcement learning training.

4 Experiment Methodology

To evaluate how Bayesian methods influence RL training, we run the Expert experiment, where the agent trains against a stationary, pre-trained expert, a tiny 120-parameter neural network baseline agent. To evaluate how various training methods influence multi-agent training, we run the Self and Multi experiments, where the agent trains by self-play with the previous version of itself in the former, and the agent trains against the previous version of its opponent in the latter.

PPO, DNN and BNN Settings

The Multi-Joint dynamics with Contact (MuJoCo) hyperparameters are chosen for PPO in Stable Baselines. The default implementation of PPO in Stable Baselines uses a multi-layer perceptron with 2 layers of 64 units as a feature extractor [14], which we label the DNN version. We implement the BNN variant, which uses 2 TensorFlow Probability Flipout layers of 64 units as the feature extractor, changing only the type of layer in the architecture.

Expert Experiments

In Expert experiments, the agent is trained against a pre-trained expert baseline, for 5M timesteps, evaluating 1K times per 100K timesteps, across 10 trials on different random seeds. The Expert experiments serve as a comparable baseline to other models trained on the Slime Volleyball gym environment.

Self Experiments

In Self experiments, the agent is trained by self-training, against the previous generation of itself (e.g. DNN-v1 vs DNN-v0), for 50M timesteps, evaluating 100 times per 1M timesteps. Each generation is updated when the current agent scores ≥ 0.5 against its previous generation. The Self experiments provide an opponent-agnostic multi-agent training environment (in contrast to training against an expert) so that agents can be better evaluated for generalisability.

Multi Experiments

In Multi experiments, the agent is trained in a multi-agent context, against the previous generation of its opponent (e.g. DNN-v1 vs BNN-v0 and BNN-v1 vs DNN-v0 concurrently), for 50M timesteps, evaluating 100 times per 1M timesteps. In contrast to the Self experiments, in the Multi experiments, each generation is updated every 100K timesteps. The Multi experiments serve the multi-agent training aspect of the project, necessary to investigate the influence of Bayesian methods on multi-agent reinforcement learning training.

Evaluation

We comparatively evaluate between DNN and BNN versions of PPO in the 3 experiments in 5 ways: performance, sample efficiency, training stability, uncertainty and generalisability. Performance of an agent is measured by its score against the expert baseline. Sample efficiency is determined by the number of timesteps required to reach the maximum episode length. Training stability is defined by the variance in reward achieved in the last 100K timesteps. Uncertainty of an agent is quantified by the entropy across action probabilities, given observations at uncertain regions of the state space, $H(X) = -\sum_{i=1}^{n} P(x_i) log_2 P(x_i)$, where $n = 3$ since the action space contains 3 actions. We define uncertain regions as the observations when 1) magnitude of ball velocity exceeds 99th percentile, and 2) ball is near the net, i.e. $|x| < 0.5$ and $y < 1$. Generalisability can be compared between two agents' performance by evaluating against an opponent not seen during training for 100 games.

5 Experiment Results

Expert Experiment

Performance

PPO-DNN and PPO-BNN are evaluated against the baseline expert that they were trained on over 100 games. PPO-DNN and PPO-BNN scored similarly, with a score of 0.02 and 0.05 respectively. There is no significant difference in performance when evaluated against the baseline expert.

Training Stability

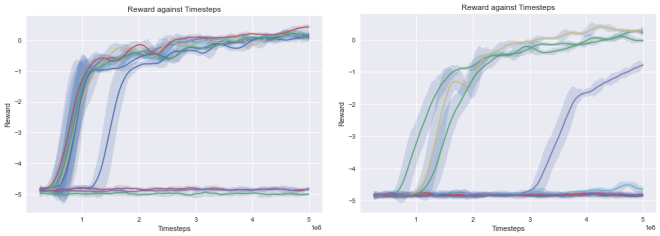

Fig. 1: Reward against timesteps (left: PPO-DNN, right: PPO-BNN)

Variance of reward achieved in the last 100K timesteps by PPO-BNN=2.862 < PPO-DNN=3.996, indicating much higher training stability for PPO-BNN.

Uncertainty

A batch of 100 games is evaluated between PPO-DNN and PPO-BNN, randomly chosen from trials that improved for PPO-BNN. At uncertain regions of the state space, when ball velocity exceeds the 99th percentile, entropy of PPO-DNN=0.623 > PPO-BNN=0.446, and when the ball is near the net, entropy of PPO-DNN=0.774 > PPO-BNN=0.416. indicating much lower uncertainty in PPO-BNN agents.

Generalisability

Evaluation	PPO-DNN (Expert)	PPO-BNN (Expert)
Baseline	0.02 ± 0.990	**0.05** ± 0.805
PPO-DNN (Expert)	-	**0.14** ± 0.762
PPO-BNN (Expert)	-0.14 ± 0.762	-

Since PPO-DNN and PPO-BNN were trained against the expert baseline, generalisability can be evaluated by playing against its policy counterpart. Over 100 games, the score of PPO-BNN=0.14 > PPO-DNN=-0.14, indicating higher generalisability for PPO-BNN agents.

Sample Efficiency

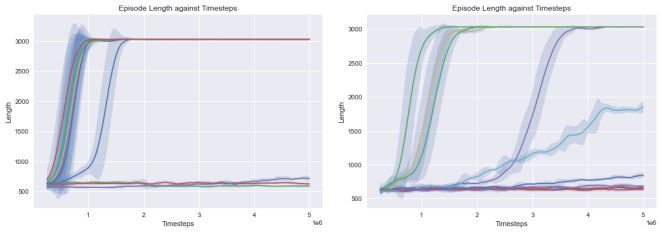

Fig. 2: Episode length against timesteps (left: PPO-DNN, right: PPO-BNN)

As shown in Figure 2, the number of timesteps required before the agent converges to the maximum episode length is much higher for PPO-BNN than PPO-DNN. In more random seeds, PPO-BNN is seen to not improve beyond the random policy, shown as episode length hovers around 600.

Agent Behaviour

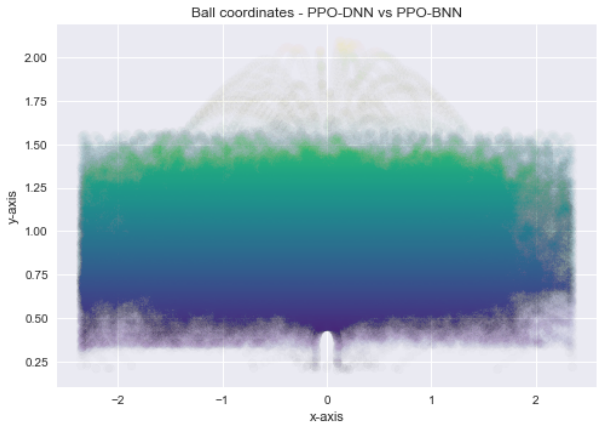

Fig 3: Ball coordinates (left court: PPO-BNN, right court: PPO-DNN)

Figure X shows ball coordinates over 100 games played between PPO-BNN on the left court (negative x-axis) and PPO-DNN on the right court (positive x-axis). The translucent ball coordinate plots visually represent the occurrences of the ball throughout the game.

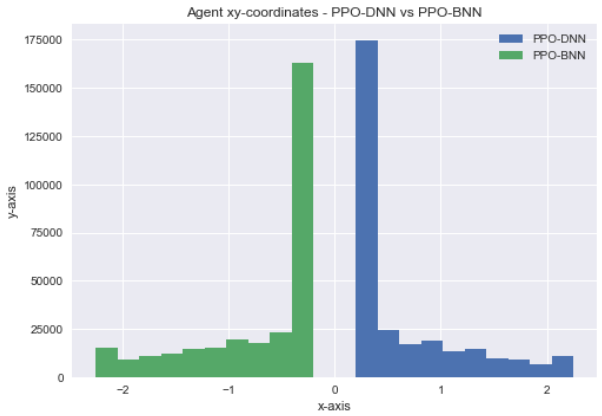

Fig 4: Histogram of agent x-coordinates (left court: PPO-BNN, right court: PPO-DNN)

Plotting a histogram of agent x-coordinates, when PPO-DNN is played against PPO-BNN, there is little notable difference in agent positioning in the court over 100 games.

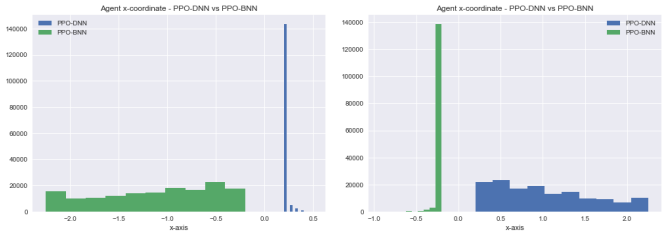

Fig. 5: Agent coordinates when the ball is on the left (left) and right (right)

Figure 5 is a histogram of agent x-coordinates when the ball is on the left and right of the court. When the ball is on the left of the court, PPO-BNN (left court), expectedly, reflects an almost uniform distribution along the x-axis, and vice versa when the ball is on the right of the court.

Fig. 6: Agent coordinates when PPO-BNN scores (left) and PPO-DNN scores (right)

Figure 6 is a histogram of agent x-coordinates when either PPO-BNN or PPO-DNN scores a point. Most of the time, when either side scores a win, the loser is positioned at the net.

Self Experiment

Performance

PPO-DNN and PPO-BNN are evaluated against the baseline expert that was not seen in training over 100 games. A higher score of PPO-BNN=0.38 > PPO-DNN=0.32 indicates better performance of the PPO-BNN agent.

Training Stability

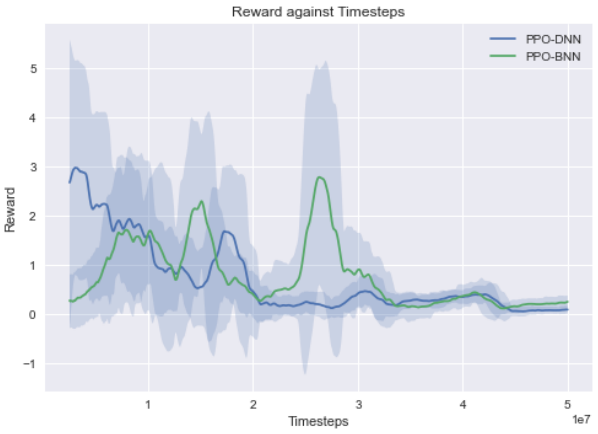

Fig. 7: Reward against timesteps

Nonetheless, variance of reward achieved in the last 100K timesteps by PPO-BNN=0.728 < PPO-DNN=0.848. As we expect reward for self-training to revolve around 0 nearing convergence, a lower reward variance indicates higher training stability of the PPO-BNN agent.

Uncertainty

A batch of 100 games is evaluated between self-trained PPO-DNN and PPO-BNN. At uncertain regions of the state space, when ball velocity exceeds the 99th percentile, entropy of PPO-DNN=0.565 > PPO-BNN=0.434, and when the ball is near the net, entropy of PPO-DNN=0.623 > PPO-BNN=0.446. The lower entropy indicates lower uncertainty in the PPO-BNN agent.

Generalisability

Evaluation	PPO-DNN (Self)	PPO-BNN (Self)

Baseline	0.32 ± 0.691	**0.38** +- 0.903
PPO-DNN (Self)	-	0.00 ± 0.583
PPO-BNN (Self)	0.00 ± 0.583	-
PPO-DNN (Expert)	3.47 ± 1.330	**4.71** ± 0.668
PPO-BNN (Expert)	4.50 ± 0.866	**4.65** ± 0.740

Since PPO-DNN and PPO-BNN were self-trained, generalisability can be evaluated by playing against opponents not seen during training: the expert baseline, the variant counterpart and the agents from the Expert experiments. Across all evaluations, PPO-BNN achieves a higher score than PPO-DNN, indicating that PPO-BNN has higher generalisability than PPO-DNN.

Sample Efficiency

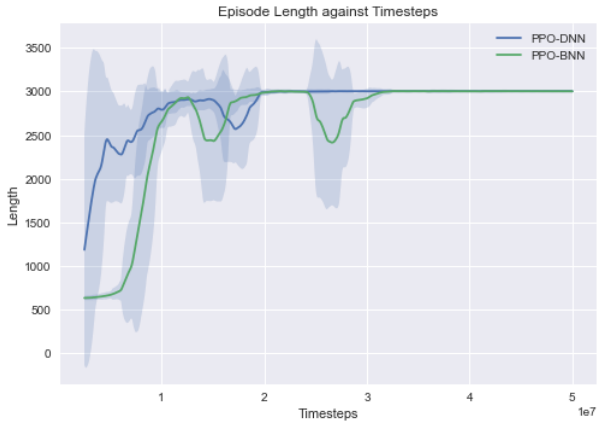

Fig. 8: Episode length against timesteps

However, PPO-BNN is less sample efficient than PPO-DNN as PPO-DNN converges to the maximum episode length in 20M timesteps, fewer timesteps than required by PPO-BNN of 30M timesteps.

Agent Behaviour

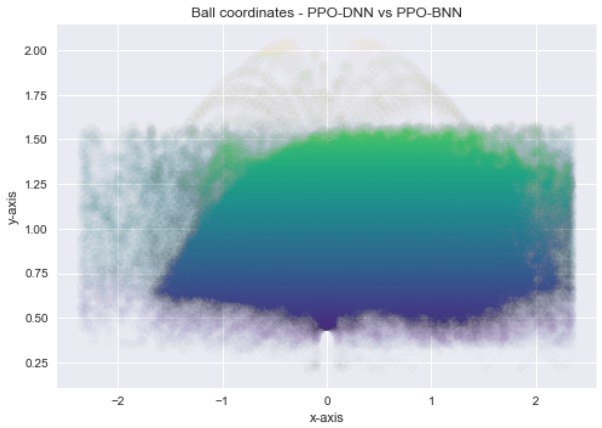

Fig 9: Ball coordinates (left court: PPO-BNN, right court: PPO-DNN)

When PPO-DNN (positive x-axis) is played against PPO-BNN (negative x-axis), from coordinates of the ball (Fig 2), we notice PPO-BNN bouncing the ball at a sharp angle, frequently returning the ball to PPO-DNN in one bounce. As a result, PPO-DNN usually requires more than one bounce to return the ball.

This can be interpreted as an emergent behaviour due to the introduction of the Flipout Bayesian method into the PPO algorithm, as PPO-BNN seems to have a consistent advantage to be able to have better control over the ball such that it can more consistently lob the ball to the side of PPO-DNN.

Fig 10: Histogram of agent x-coordinates (left court: PPO-BNN, right court: PPO-DNN)

In contrast to the Expert experiment, agents stick to the net less and are more spread out across the court.

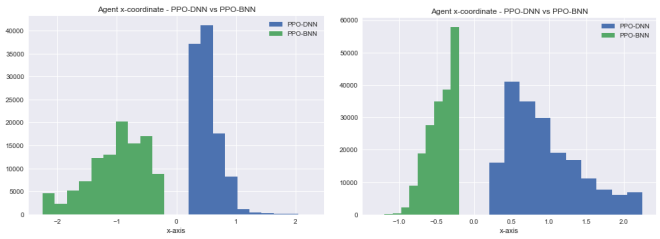

Fig. 11: Agent coordinates when the ball is on the left (left) and right (right)

Figure 11 is a histogram of agent x-coordinates when the ball is on the left and right of the court respectively. As compared to the agent behaviour in the Expert experiment, when the ball is on the opponent's court, both agents reflect a more spread out distribution in the x-axis than sticking closely to the net. When compared directly between PPO-DNN and PPO-BNN, when the ball is on the opponent's side of the court, PPO-BNN reflects a gentler decrease in frequency as distance away from the net increases (right graph) than that of PPO-DNN (left graph).

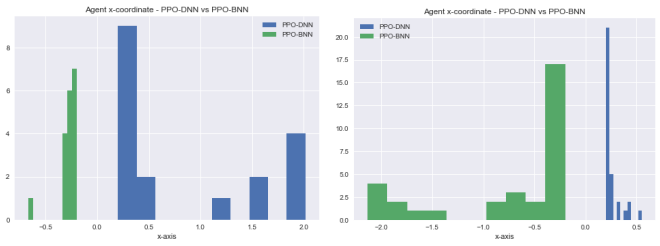

Fig. 12: Agent coordinates when PPO-BNN scores (left) and PPO-DNN scores (right)

Figure X is a histogram of agent x-coordinates when either PPO-BNN or PPO-DNN scores a point. As compared to the agent behaviour in the Expert experiment, when PPO-BNN (left court) scores a point (left graph), PPO-DNN (right court) is more frequently found to be on the extreme right of the court.

Multi Experiment

Performance

PPO-DNN and PPO-BNN are evaluated against the baseline expert that was not seen in training over 100 games. PPO-DNN and PPO-BNN scored similarly, with a score of 0.07 and 0.04 respectively.

Training Stability

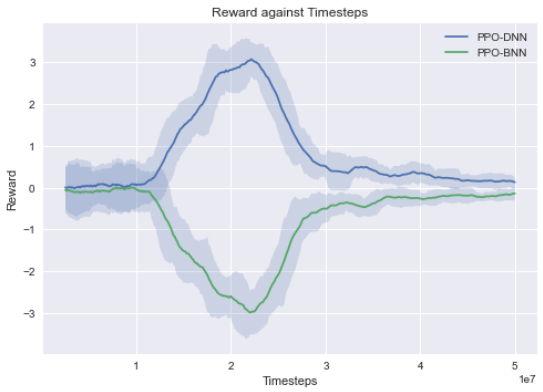

Fig. 13: Reward against timesteps

In multi-agent training between PPO-DNN and PPO-BNN, PPO-DNN first performs better than PPO-BNN as expected as PPO-DNN is exploiting PPO-BNN due to better sample efficiency, then converges towards 0 nearing the end of training.

Uncertainty

A batch of 100 games is evaluated between PPO-DNN and PPO-BNN. At uncertain regions of the state space, when ball velocity exceeds the 99th percentile, entropy of PPO-DNN=0.605 > PPO-BNN=0.667, and when the ball is near the net, entropy of PPO-DNN=0.508 > PPO-BNN=0.727, indicating higher uncertainty in PPO-BNN agents. One possible reason is because PPO-DNN is more sample efficient than PPO-BNN, resulting in PPO-DNN's exploitation of PPO-BNN during multi-agent training.

Generalisability

Evaluation (100 games)	**PPO-DNN** (Multi)	**PPO-BNN** (Multi)
Baseline	0.07 ± 0.941	0.04 ± 0.926
PPO-DNN (Expert)	1.19 ± 1.317	**2.45** ± 1.564
PPO-BNN (Expert)	2.03 ± 1.500	**3.96** ± 1.303
PPO-DNN (Self)	-0.34 ± 0.738	**-0.23** ± 0.811
PPO-BNN (Self)	**-0.28** ± 0.849	-0.50 ± 0.964

Both PPO-DNN and PPO-BNN trained by multi-agent training scored worse than their counterparts trained by self-training. Nonetheless, we note the impressive performance achieved by PPO-BNN (Multi) when compared to PPO-DNN (Multi), indicating much higher generalisability in the PPO-BNN agent.

Sample Efficiency

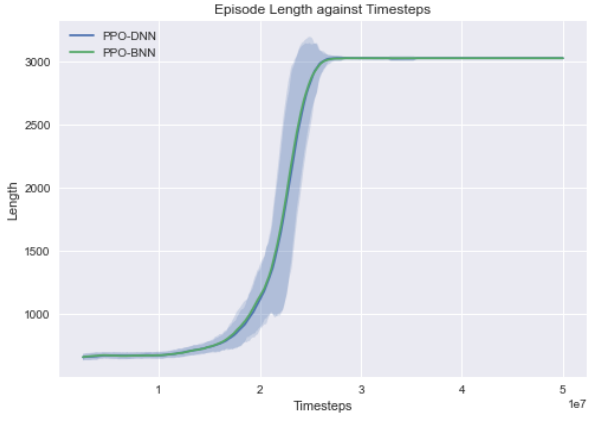

Fig. 14: Episode length against timesteps

Multi-agent training between PPO-DNN and PPO-BNN required more timesteps than self-training of PPO-DNN, as 25M timesteps were required to achieve the maximum episode length.

Agent Behaviour

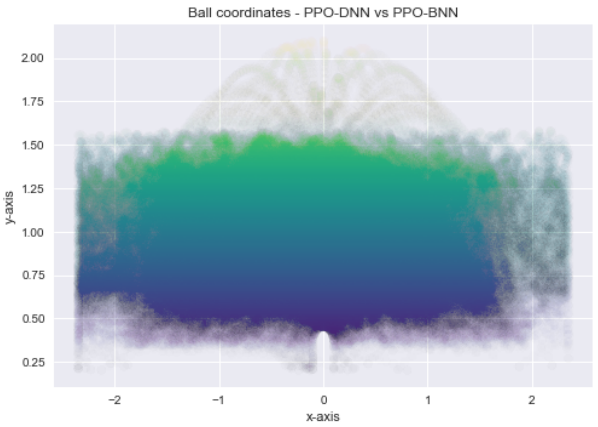

Fig 15: Ball coordinates (left court: PPO-BNN, right court: PPO-DNN)

When PPO-BNN (left court) is played against PPO-DNN (right court), PPO-DNN seldom visits the right-most side, as PPO-DNN performs better than PPO-BNN in multi-agent training.

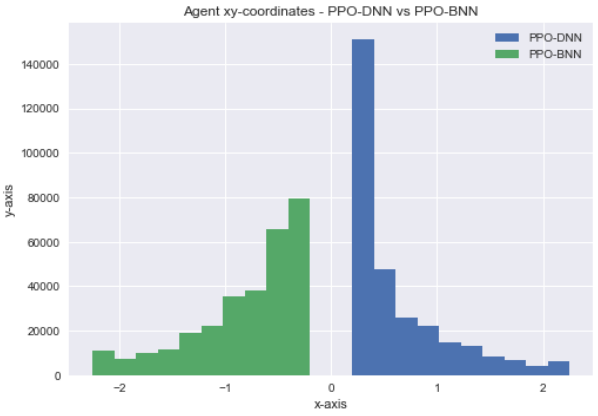

Fig 16: Histogram of agent x-coordinates (left court: PPO-DNN, right court: PPO-BNN)

Accordingly, the agent x-coordinates illustrate PPO-DNN positioning closer to the net, a common observation of better-performing agents.

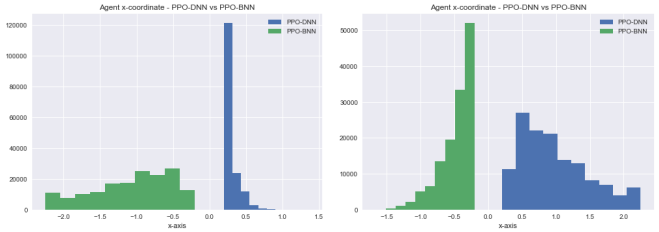

Fig. 17: Agent coordinates when the ball is on the left (left) and right (right)

Figure 17 is a histogram of agent x-coordinates when the ball is on the left and right of the court respectively. Notably, when the ball is on the right court (right graph), PPO-BNN (left court) sticks to the court much less than PPO-DNN when the ball is on the left court (left graph).

Fig. 18: Agent coordinates when PPO-BNN scores (left) and PPO-DNN scores (right)

Figure 18 is a histogram of agent x-coordinates when either PPO-BNN or PPO-DNN scores a point. Consistent with the Expert and

Self experiments, when either agent scores a point, the opponent is found most frequently close to the net, followed by being at the far end of the court, with few to none occurrences at the middle of the court.

6 Discussion

Comparison between DNN-PPO and BNN-PPO

PPO-BNN was superior to PPO-DNN in performance in the Self experiments. PPO-BNN was inferior to PPO-DNN in sample efficiency in the Expert and Self experiments. PPO-BNN was superior to PPO-DNN in training stability across all experiments. PPO-BNN was superior to PPO-DNN in uncertainty in the Expert and Self experiments. PPO-BNN was superior to PPO-DNN in generalisability across all experiments. Although PPO-BNN was slightly inferior in sample efficiency in achieving the maximum episode length, the superior generalisability of the PPO-BNN agent is a far greater improvement over the PPO-DNN agent.

Generalisability from Bayesian methods

The introduction of the Flipout layer as a feature extractor in the implementation of the PPO-BNN agent that significantly improved uncertainty at uncertain regions measured by entropy and generalisability over a range of unseen opponents across the three experiments, can be explained by the reduction of variance in estimation of gradients in the PPO algorithm.

Agent Behaviour

Emergent behaviour is observed as training converges to the optimal playstyle of agents in general, both PPO-DNN and PPO-BNN. At the start of training, agents first learn to prevent the ball from landing on their own court to prevent the opponent from scoring the point. Over a long period of training time, the agent learns to return the ball to the opponent in relatively fewer bounces. We initially thought that this was the best strategy that agents could learn. However, it surprised us that, at convergence, which is observed only by agents in the Self experiment, agents become more capable of returning low balls, (low change in the y-axis) and move quickly (high change in the x-axis). Agents at this level of gameplay can perform returns of high technical ability to maximise the chance of scoring a point.

Experimental Limitations

Hyperparameter optimisation, while attempted, is decidedly infeasible due to the highly stochastic nature of RL training, exacerbated by time and compute limitations. As each trial requires one hyperparameter optimisation, each iteration requires a training iteration that takes up to 4 hours, 10 trials of 10 hyperparameter iterations will easily take up to 400 hours for a single experiment, which is infeasible within the constraints of the project.

While the Expert experiments are performed across 10 trials, despite our best intentions, Self and Multi experiments are performed only once, to the undesirable but likely possibility of variance across trials. Because Expert experiments only require 5M timesteps to show signs of convergence to maximum episode length, Self and Multi experiments were run with 50M timesteps, and is difficult to repeat over multiple trials with the time available allocated to the project. Nonetheless, it is important to emphasise that we did not, and could not anyway due to time constraints, cherry pick the results.

Multi-agent training is implemented in generations of opponents instead of live, due to the highly restricting single-agent framework of Stable Baselines. It is a known challenge for many open source libraries, not just Stable Baselines, to support multi-agent training due to the many nuances required to adapt and generalise across all the possible, highly diverse types of multi-agent training experiments.

Future Work

We hope to extend our work in the future by implementing various agent exploration strategies such as neuroevolution and partial observability in the environment by withdrawing frames, investigating the influence of Bayesian methods on multi-agent reinforcement learning training and agent behaviour in various contexts. A potential direction to consider working towards is using the visual scene pixels as input to the policy model, instead of using features extracted from the simulation, to reveal new challenges in the domain of multi-agent reinforcement learning.

7 Conclusion

In conclusion, from studying the difference from integrating a Bayesian method into multi-agent training of reinforcement learning agents, in terms of performance, training stability, uncertainty, generalisability and sample efficiency, we show that Flipout improves multi-agent PPO in Slime Volleyball across performance, training stability, uncertainty and generalisability, with a relatively small trade-off in sample efficiency. Our results that Bayesian methods improve uncertainty and generalisability of reinforcement learning agents is significant not just to games but to many applications in the real world, such as in robotic process automation, autonomous vehicles and service robots.

Acknowledgements

We would like to thank our tutor Liu Yingnan for her guidance, Professor Kan Min-Yen for the opportunity to do this project in the scope of the CS3244 Machine Learning class, David Ha for the Slime Volleyball environment that enabled our experiments, AWS for sponsoring compute instances and, leaving the best for the last, Jet's Dell XPS 15 9570 laptop for its noble sacrifice of its laptop battery in the line of duty due to impending deadlines.

References

[1] Baker, Bowen, et al. "Emergent tool use from multi-agent autocurricula." arXiv preprint arXiv:1909.07528 (2019).

[2] Zheng, Stephan, et al. "The AI Economist: Improving Equality and Productivity with AI-Driven Tax Policies." arXiv preprint arXiv:2004.13332 (2020).

[3] Kaelbling, Leslie Pack, Michael L. Littman, and Andrew W. Moore. "Reinforcement learning: A survey." Journal of artificial intelligence research 4 (1996): 237-285.

[4] Ghavamzadeh, Mohammad, et al. "Bayesian reinforcement learning: A survey." arXiv preprint arXiv:1609.04436 (2016).

[5] Hessel, Matteo, et al. "Multi-task deep reinforcement learning with popart." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 33. 2019.

[6] Fischer, Thomas G. Reinforcement learning in financial markets-a survey. No. 12/2018. FAU Discussion Papers in Economics, 2018.

[7] Kober, Jens, J. Andrew Bagnell, and Jan Peters. "Reinforcement learning in robotics: A survey." The International Journal of Robotics Research 32.11 (2013): 1238-1274.

[8] Zhang, Kaiqing, Zhuoran Yang, and Tamer Başar. "Multi-agent reinforcement learning: A selective overview of theories and algorithms." arXiv preprint arXiv:1911.10635 (2019).

[9] Deisenroth, Marc Peter, Gerhard Neumann, and Jan Peters. A survey on policy search for robotics. now publishers, 2013.

[10] Schulman, John, et al. "Proximal policy optimization algorithms." arXiv preprint arXiv:1707.06347 (2017).

[11] Hoffman, Matthew D., et al. "Stochastic variational inference." The Journal of Machine Learning Research 14.1 (2013): 1303-1347.

[12] Wen, Yeming, et al. "Flipout: Efficient pseudo-independent weight perturbations on mini-batches." arXiv preprint arXiv:1803.04386 (2018).

[13] David Ha (2020) Slime Volleyball Gym Environment. Available at: https://github.com/hardmaru/slimevolleygym

[14] Ashley Hill (2018) Stable Baselines - [Online]. Available at: https://github.com/hill-a/stable-baselines