Uncertainty Uses in Reinforcement Learning Both Training and Deployment

Rahul Krupani, Micah Bryant, Joseph Gleason, Tom Watson, Anastacia MacAllister

General Atomics ASI

Poway, CA

rahul.krupani@ga-asi.com, micah.bryant@ga-asi.com, joseph.gleason@ga-asi.com, thomas.watson@ga-asi.com, anastacia.macallister@ga-asi.com

ABSTRACT

Artificial Intelligence (AI) and Machine Learning have become focal points for Department of Defense (DoD), as evidenced by President Trump announcing a $500 billion investment in infrastructure that is tied to AI. A critical aspect of this heightened investment in AI is the development of fully autonomous, unmanned aerial systems (UAS). However, the inherent challenges posed by the complexity of operational environments in which these UAS operate, coupled with the evolving prioritization of subgoals, present significant hurdles for traditional control algorithms. Reinforcement Learning (RL) offers a solution that will enable warfighters to dynamically devise actionable control strategies to achieve mission success, but RL can struggle with long horizon tasks and the formulation of a reward structure thereby causing long periods of trial and error by developers. When deploying the system, there are concerns when the agent begins to proceed into areas outside of its training process thereby leading to indeterminate results.

In previous work we examined how the usage of distribution and ensemble methods allows reinforcement learning agents to gain an understanding of both epistemic and aleatoric uncertainties. In this work we shift our focus to examine how uncertainty can be used during training to improve exploration and simplify reward function creation. Our initial study demonstrates that epistemic uncertainty can be used to help increase exploration during training in areas that are no associated with high rewards. Additionally, we show that uncertainty can be utilized during deployment to prevent unknown behavior and trigger an appropriate contingency thereby allowing for autonomy that can be utilized in real world environments. More studies are needed but this paper introduces how uncertainty while training and deploying RL agents can benefit the warfighter.

ABOUT THE AUTHORS

Rahul Krupani is a Machine Learning Engineer for General Atomics Aeronautical Systems (GA-ASI). His work focuses on software autonomy infrastructure and reinforcement learning. He holds an M.S. in Computer Science from USC.

Micah Bryant is a Machine Learning Engineer for GA-ASI. His work focuses on architecting, implementing, and validating reinforcement learning agents. He holds a B.S. in Bioengineering from UCLA and his M.S. in Mechanical Engineering from UCSD.

Joseph D. Gleason, Ph.D., is a Machine Learning Engineer with GA-ASI. His work focuses on applied controls and reinforcement learning algorithms. He has a B.S. in Electrical Engineering from Arizona State University and a Ph.D. in Electrical Engineering from the University of New Mexico.

Tom Watson is a Machine Learning Engineer for General Atomics Aeronautical Systems. His work focuses on architecting and implementing software for autonomous flights. He graduated from the University of Southern California with his B.S and M.S in Computer Science.

Anastacia MacAllister, Ph.D., is Technical Director of Autonomy and Artificial Intelligence for GA-ASI. Her work focuses on prototyping and developing novel machine learning algorithms and exploratory data analytics. She has provided key contributions in prognostic health management, human performance augmentation, advanced sensing, and AI for future warfare strategies. Dr. MacAllister holds a B.S from Iowa State University (ISU) in Mechanical Engineering, and an M.S. and Ph.D. from ISU in Mechanical Engineering and Human-Computer Interaction.

Uncertainty Uses in Reinforcement Learning Both Training and Deployment

Rahul Krupani, Micah Bryant, Joseph Gleason, Tom Watson, Anastacia MacAllister

General Atomics ASI

Poway, CA

rahul.krupani@ga-asi.com, micah.bryant@ga-asi.com, joseph.gleason@ga-asi.com, thomas.watson@ga-asi.com, anastacia.macallister@ga-asi.com

INTRODUCTION

Autonomous systems are a rapidly growing field of development for both public and private industry. From self-driving cars, autonomous submersible, and unmanned aerial vehicles, the development of advanced algorithms and the increase of computational capabilities have brought forward potential for fully (or largely) autonomous vehicles. When designing a control strategy for these autonomous systems we often separate control into two categories: direct solutions and sampling-based methods. Direct solution methods are the foundations of control theory and rely on the ability to either solve an optimal control problem explicitly or solve some numerical approximation. This often imposes certain restrictions, the most common of which are the assumptions of linearity and/or convexity. Sampling-based methods such as genetic algorithms or particle swarm optimizations are more generalizable but at the cost of computation time which often renders them ineffective in many complex environments that require frequent state updates.

Perhaps the most promising sample-based method is reinforcement learning (RL). RL has some unique advantages: 1) compared to other sample-based methods, its backward gradient propagation is computationally efficient at tuning models with a large number of parameters; 2) RL is capable of being trained in simulated scenarios and moved to, and potentially quickly retrained on, real-world systems; 3) as a parameterized model, RL can be used on real-time systems. It has been applied in the area of games (Silver, et al., 2017) (Schrittwieser, et al., 2020) (Berner, et al., 2019), automated driving (Kiran, et al., 2022) (Cao, et al., 2023) (Wang, et al., 2023) (Jebessa, Olana, Getachew, Isteefanos, & Mohd, 2022), robotics (Ibar, et al., 2021) (Toyota Research Institute Unveils Breakthrough in Teaching Robots New Behaviors, 2023) (Toner, Saez, Tilbury, & Barton, 2023), and is an active area of research interest in academia, government, and industry.

RL has advanced significantly within the last decade but still sees very limited real-world application especially in safety-critical systems. One reason for this limited application is because of its lack of understanding of both risk and uncertainty. This can lead to unsafe behaviors especially when the model encounters states and values it has not seen in training which is commonly called an out-of-distribution problem. In the best case this only causes harm to the system or erratic behavior of the model, but in unconstrained environments can lead to harm to operators and bystanders of the system. To address this problem, we use a combination of distributional (Bellemare, Dabney, & Munos, 2017) and ensemble (Song, et al., 2023) reinforcement learning.

With a distributional reinforcement learning system (Bellemare, Dabney, & Munos, 2017), instead of estimating and predicting the expected cumulative reward, the system learns to estimate an approximation of the distribution of the cumulative reward. This is important because it provides the autonomous systems with an intrinsic understanding of risk since its internal representation of its reward is a distribution instead of a scalar approximation of the expected return.

The ensemble methods (Song, et al., 2023) enable the system to determine estimates of both aleatoric and epistemic uncertainty. Aleatoric uncertainty is a representation of the intrinsic uncertainty present in the system. Epistemic uncertainty is related to a lack of knowledge, which for a data-based learning system can be uncertainty from a lack of exploration, i.e. uncertainty because the system may be in a state that it did not see in training. Systems that model their uncertainty can use the uncertainty to, for example, help drive exploration during training by adding training rewards for moving the system to uncertain states.

The primary focus of this work is to expand upon our previous research (Bryant, Gleason, & Macallister, 2024) (Krupani, Gleason, Bryant, Watson, & Macallister, 2025) and discuss the utility of uncertainty estimation for RL agents used in autonomous aerial systems. This paper seeks to demonstrate the various uses that uncertainty can play while training and deploying models that can calculate this uncertainty rapidly. Uncertainty lets us calculate an intrinsic measure of trust which can be utilized during deployment; however, it also lets us calculate an intrinsic level of the agent's understanding about the environment which has been shown to provide benefits during training which is referred to as curiosity in the reinforcement learning realm.

RELATED WORK

Previous work has examined combining ensemble and distributional RL methods (Hoel, Wolff, & Laine, 2023) (Williams, Gee, MacDonaland, & Liarokapis, 2024) (Eriksson, Basu, Alibeigi, & Dimitrakakis, 2022), however to the best of the authors' knowledge, there have been no works focusing on combined distributional and ensemble RL utilized to estimate usable values of uncertainty on UAS. In (Seres, Liu, & Kampen, 2023), the authors examine risk-sensitive distributional RL for flight control systems but do not examine the combination of ensemble and distributional RL. Both (Eriksson, Basu, Alibeigi, & Dimitrakakis, 2022) and (Williams, Gee, MacDonaland, & Liarokapis, 2024), the authors examine combined distributional and ensemble RL but from the lens of theoretical extension and focus less on application. The authors of (Hoel, Wolff, & Laine, 2023) examine applied ensemble and distributional RL for autonomous driving which, while similar, faces different challenges than the warfighter experiences in flight operations.

BACKGROUND

In this section we detail traditional reinforcement learning and the updates—ensemble and distributional RL—used in this work. Common to all these methods, RL is a sample-based unconstrained optimization procedure that seeks to find an optimal control policy that maximizes some function of the reward (or minimizes the cost). In each we consider the dynamical system to be a Markov Decision Process (MDP), which is described by the tuple (S, A, P, r, γ), where S is the state space, A is the action space, $P: S \times A \to S$ is the state transition function, i.e. given a state and action it provides the next state according to some probability distribution, $r: S \times A \to \mathbb{R}$ is the reward function (or negative cost function), and $\gamma \in [0, 1)$ is the discount factor. Typically, MDPs are considered to represent discrete-time systems. While continuous MDP variants exist (Puterman, 2005), we will also assume a discrete system. We denote the state and action at time t as s_t and a_t, respectively.

In general, we would like to determine an optimal feedback control strategy $\pi: S \to A$ over some policy space Π. We denote the optimal policy as π^*. In practice, searching over the space of all feedback policies is intractable, so, as with current RL methods, we search over a parametrized policy space Π_θ, where $\theta \in \Theta$ is a parameter vector and Π_θ is derived by the structure of the neural network we use. In the remainder of this paper, we will often use π to refer to π_θ for notational simplicity unless it is important to distinguish the difference.

Traditional reinforcement learning seeks to optimize the expected return over some (typically infinite) time-horizon:

$$\text{maximize}_{\theta \in \Theta} \quad \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r(s_i, a) \,|a \sim \pi(s_i)\right] \tag{1}$$

This has several computational and theoretical advantages. Under some conditions, RL is guaranteed to converge to the optimal policy π^* (Sutton & Bartow, 2018). However, because it maximized the expected return, it can be susceptible to taking risky behavior if the reward for the behavior is sufficiently large.

Distributional Reinforcement Learning

For simulated systems, e.g. games, the maximization of the expected reward done by traditional RL may be acceptable. However, in real-world systems, e.g. UAS, this average risk tolerance may be unacceptable. From (1) we define the Q-function (Sutton & Bartow, 2018)

$$\begin{aligned} Q(s,a) &= \mathbb{E}\left[\sum_{i=0}^{\infty} \gamma^i r(s_i,a) \,|a{\sim}\pi(s_i)\right] \\ &= \mathbb{E}[r(s,a)|\, a{\sim}\pi(s)] + Q(s',a') \end{aligned}$$

where $s'{\sim}P(s,a)$ and $a'{\sim}\pi(s')$. This is called the Bellman equation under policy π. We can also see that traditional RL seeks to maximize Q over the set of parameters θ.

The challenge with optimizing Q, or optimizing only for the expectation, is that it can cause the RL system to strike an, often, unintentional balance between reward and uncertainty. Consider the following example: A user can choose to press either button A or B. If A is pressed, the user receives a reward of 1. If B is pressed, then with probability $p = 0.5$, the user receives a reward of 4 and receives a reward of -1 otherwise. On average, i.e. from the perspective of Q, the best choice is to press B, whose average reward is 1.5. However, if we are concerned with determining optimal conditions that, for example, maximize the minimum possible return, then the optimal choice becomes A. While reward optimization with simple numbers is of little consequence, the warfighter, and systems that assist the warfighter, must be more considerate of the possibility for negative outcomes.

For distributional RL, we instead examine the distributional Bellman equation:

$$Z(s,a) \overset{D}{=} r(s,a) + Z(s',a')$$

The above is an equality in distribution where s' and a' are defined as before. The advantage here is that because we learn the distribution of the reward, the system can represent the intrinsic uncertainty in the reward, as seen in the simple example above. We must, however, choose what kind of distribution we want to use to represent our reward. In this work we use Gaussian models as in (Duan, et al., 2022).

Ensemble Reinforcement Learning

There are many uses for ensemble methods (Song, et al., 2023) but the focus of this work is on using ensemble methods to create an approximate metric for the uncertainty of the RL network, specifically the epistemic uncertainty. In uncertainty quantification there are two primary categories of uncertainty, aleatoric and epistemic. Aleatoric uncertainty deals with intrinsic uncertainty in the dynamics of the system while epistemic uncertainty describes uncertainty in knowledge. In general, both are extremely hard, if possible, to exactly quantify (Hüllermeier & Waegeman, 2021). However, ensemble learning methods have demonstrated promise in creating estimates of uncertainty that can be practically used in complex system (Hoel, Wolff, & Laine, 2023) (Song, et al., 2023) (Abdar, et al., 2021), for example UAS.

Ensemble RL is an extension on traditional RL (or distributional RL) where instead of optimizing over a single Q-function, we optimize over several of them, Q_m for $m \in \{0, 1, 2, \dots, n\}$. We may also have an ensemble of policies π_k for $k \in \{0, 1, 2, \dots, l\}$. While exact quantification of epistemic uncertainty is nontrivial, with multiple critic networks, we can create an estimate of the epistemic uncertainty by examining different statistics in the variation of the predictions from the various critics. For example, we can assess the uncertainty given a state and action by looking at the variance in the critic predictions:

$$U(s,a) = \mathrm{var}\big(Q_m(s,a)\big)$$

Higher variance indicates more variation in the critic predictions which typically indicates an insufficient number of training samples to create alignment in the ensemble. Variance in the critic estimates does not fully decouple aleatoric and epistemic uncertainty, however this is not possible for general systems (Hüllermeier & Waegeman, 2021).

USES FOR UNDERTAINTY DURING TRAINING

While certainly not comprehensive, there are two immediately apparent applications for uncertainty during the training of an RL agent: 1) to increase exploration, and 2) as an additional disincentive for agents.

First, uncertainty can help increase exploration. This is particularly true for epistemic uncertainty. As mentioned before, epistemic uncertainty has to do with uncertainty relating to knowledge—or the lack thereof. For an RL system this allows us to utilize estimates of epistemic uncertainty to help create an incentive for curiosity, or an incentive for the agent to increase its overall exploration. To motivate this, we will first discuss how we approximate epistemic uncertainty in the distributional ensemble learning methodology.

With ensemble RL methods, as we have done previously (Krupani, Gleason, Bryant, Watson, & Macallister, 2025) (Bryant, Gleason, & Macallister, 2024), we utilize an average of our critic predictions to determine an estimate of the epistemic uncertainty. Specifically, for our distributional critics $Z_i, i \in \{1, \dots, N\}$, where N is the total number of critics we use during our ensemble training, we define our approximation of the epistemic uncertainty as

$$u_e = Var_N(E[Z_i])$$

In words, the epistemic uncertainty is the variance of the mean reward values given by the critics.

The reason this provides an estimate of the epistemic uncertainty is that each of the neural networks of the distributions are initialized, they are given random starting weights. Thus, each of the critic networks, upon initialization, will have differing estimations of the mean reward $E[Z_i]$. As these networks experience more training, however, they will experience more samples of similar rewards which will drive each critic to have a consistent estimation of the mean reward. So, during training as the system more often explores a state, the critics are more likely to predict the same mean reward, thus reducing the epistemic uncertainty.

Ultimately, we want to use this epistemic uncertainty to influence the training system to increase exploration. We can do this by augmenting the reward function for the policy to provide increased reward for driving the system to epistemically uncertain states.

$$r_e(s_t, a_t) = r(s_t, a_t) + \lambda_e u_e$$

Here, λ_e is a hyperparameter that controls the total weighting of the epistemic uncertainty influence during training. As the actual magnitude of the reward, and thus the magnitude of the epistemic uncertainty, is not consistent for every environment, this hyperparameter must be chosen for each training scenario. Additionally, while we want to increase exploration during training, we do not want our deployed system to have a similar drive to explore. As such we suggest that this hyperparameter conforms to some form of scheduled annealing during training or that the training pipeline utilizes some type of curriculum staging in which exploration is initially rewarded but eventually $\lambda_e \to 0$ as training increases. This ensures that the deployed system's goal is to maximize the given reward.

The second use-case we will discuss for uncertainty in training is to add an additional penalty for RL systems to avoid areas of high uncertainty. For this we are typically concerned with ensuring that the system avoid areas of higher aleatoric uncertainty, as this uncertainty is intrinsic to the MDP that the RL agent is trying to optimize and cannot typically be reduced through increased exploration. For this, we define our approximation of aleatoric uncertainty as the mean of the variance in the reward given by each critic.

$$u_a = \frac{1}{N}\sum_{i=1}^{N} \mathrm{var}_i\big(Z_i(s,a)\big)$$

Similar to the epistemic uncertainty enhancing training, we can alter the reward function to provide a disincentive for the agent to reach areas of high uncertainty:

$$r_a(s_t, a_t) = r(s_t, a_t) - \lambda_a u_a$$

It is important to note that often distributional RL will use some kind of risk aware metric which can already provide some kind of penalty for states with high uncertainty. In fact, the approximate CVaR risk metric used in (Krupani, Gleason, Bryant, Watson, & Macallister, 2025) has a term that is proportional to $-u_a$ intrinsically. Thus, this type of penalty is not always needed. However, there are instances where CVaR may not be available or additional conservativeness is desired, and this aleatoric uncertainty penalty provides the designer with different training options.

USES FOR UNCERTAINTY IN DEPLOYMENT

The primary utility for uncertainty in deployment that we will highlight is for bounding regions of "safe" operation of the deployed agents. Again, there are myriad other potential uses that we do not discuss here.

There are two ways to use uncertainty to constrain operational zones. The first is to, through post-hoc analysis, sample and define regions in which epistemic and aleatoric uncertainties are very low. You can see an example of these graphs in (Bryant, Gleason, & Macallister, 2024) Figure 1. We do not typically suggest this because often the state space of interest for RL applications is very large or sufficiently complex that determining safe operation regions manually is intractable. So instead, we propose an online solution using normalized uncertainty estimations.

In our training setup, we use a distributional form of the Soft Actor-Critic algorithm. This is an off-policy learning method and, as such, stores its training data in a training buffer. Once training is complete, we save the data in this buffer. Before deployment we use the data in the buffer to establish a mean and standard deviation of both the epistemic and aleatoric uncertainty values. These parameters are used to create a normalization of our uncertainty estimates that we will use in deployment. This normalization is not strictly necessary but does allow for more general application since the uncertainty estimates ranges are not necessarily similar across different training environments.

With this normalization, during deployment we can use our distributional and ensemble setup to obtain normalized estimates of the aleatoric and epistemic uncertainties[1]. Usually, we are concerned with uncertainties that are uncharacteristically high, thus we can threshold our uncertainties to be, for example, no more than a single standard deviation above the norm. This parameter can be chosen at the operator's intent or even dynamically updated during deployment. If the uncertainty of the RL agent is higher than the desired threshold, then it is possible to trigger contingency behavior or some operator notification so that the system can be monitored or controlled to ensure safe operation during this time. Control can then be returned once uncertainty is within permitted thresholds.

There is a larger philosophical question about which uncertainties to throttle. Epistemic uncertainty is analogous to a human's ability to rate their own skill level at a given task. Thus, having high epistemic uncertainty if the RL agent's way of indicating that it has not experienced sufficient training to perform well within a given region. However, areas of high aleatoric uncertainty, the RL agent is recognizing the inherent risk in operating within this region. This is not to say that the agent cannot behave optimally according to an appropriate risk metric, thus constraining operation in times of high aleatoric uncertainty may not provide more optimal results. However, as the full explainability of behaviors for RL systems is still limited, it is also reasonable to constrain operation during high aleatoric uncertainty because of insufficient trust. Additionally, since aleatoric and epistemic uncertainty approximations are not guaranteed to be fully separable it can be advantageous to simply restrict a deployed agent's operational capacity for all types of uncertainty. Ultimately that decision is left to the deployment configuration.

[1] Note here that we are discussion normalization in the sense of a Gaussian (Normal) distribution. However, uncertainty is, by definition, a strictly positive value so normalization to something like a Gamma distribution may be more suited if tractable.

TESTING RESULTS AND DISCUSSION

Experimental Setup

For our simulations, trained agents in a simulated environment with a discretized dynamics that are meant to represent an aircraft in motion. The agent can control the rate at which the aircraft turns (alters its heading) by choosing actions between $[-1,1]$. These values are mapped to +/-20 degree turns, which is the maximum turning capacity the agent has between steps. We hold altitude and speed constant.

The agent is initialized at 0 degrees latitude and 3 degrees longitude with a random heading between $[-180, 180]$ degrees from true north. The goal for the agent is to fly west past the 0-degree longitude line without being detected by a detector located at 0 degrees latitude, 1.5 degrees longitude. The detector has an exponentially decaying probability of detection given by

$$p_d = e^{-\left(\frac{d}{10000}\right)^2}$$

where d is the geodetic distance between the agent and detector measure in meters. If the agent gets detection, then the game is over. At each step the agent gets a reward,

$$r_t = e^{-\left(\frac{h}{60}\right)^2}$$

where h is the relative heading of the agent away from due west.

This is a simple scenario that can still cause stability problems for traditional RL algorithms. For a more thorough analysis of the challenges see our previous work (Bryant, Gleason, & Macallister, 2024). In this paper we use the same scenario but focus on developing initial studies of the applications of uncertainty for training and deployment.

Uncertainty and Training

To examine the differences during training caused by epistemic uncertainty, as previously discussed, we ran a small experiment training four agents. Each agent was equivalent in hyperparameters except for the initial value of the epistemic weights: 0, 0.01, 0.001, and 0.0001. For simplicity we will refer to these as Agent+0, Agent-2, Agent-3, and Agent-4, respectively. Each weight was decayed according to a fixed schedule. The epistemic weights can be seen in Figure 1. Each agent was trained for one million steps. At the end of training, each agent had learned to reach the target and behaved similarly, with tolerance for differences based off the intrinsic randomness in the training initialization.

From our limited study, we will note two interesting and generally consistent results we saw. First, the agents with an initial epistemic uncertainty weight greater than zero generally saw an earlier peak and faster decay in the epistemic uncertainty during training. This is shown in the top graph of Figure 1. This graph shows the epistemic uncertainty value from the batches of data samples during training. For Agent-2 and Agent-3, the early peak and stronger decay than Agent+0 is obvious. For Agent-4, the maximum value is high enough that the peak occurs after approximately the same number of steps as Agent+0, but the decay is still stronger, and the training epistemic uncertainty coincides around 250k steps. Because of the scheduled reduction in the epistemic weight, all agents start to have similar epistemic uncertainties during training after around 300-400k steps. We can see an additional example of this strong decay in epistemic uncertainty in Figure 3, which shows the heatmaps for Agent+0 (top) and Agent-3(bottom) as steps increase. At 200k steps, the epistemic uncertainty for Agent-3 drops to about a 19% of its value at 100k. For Agent+0, this drop is to around 37% of its 100k value. In these heatmaps, we also see that around 300-400k the epistemic uncertainty values for each are equalizing.

The heatmaps from Figure 3, also helps demonstrate our second result: for agents training with initial epistemic weights greater than 0, we generally see the epistemic uncertainty reduce strongly in areas with previously high epistemic uncertainty. For brevity we just show this for Agent-3. First consider the graphs of Agent+0 during the

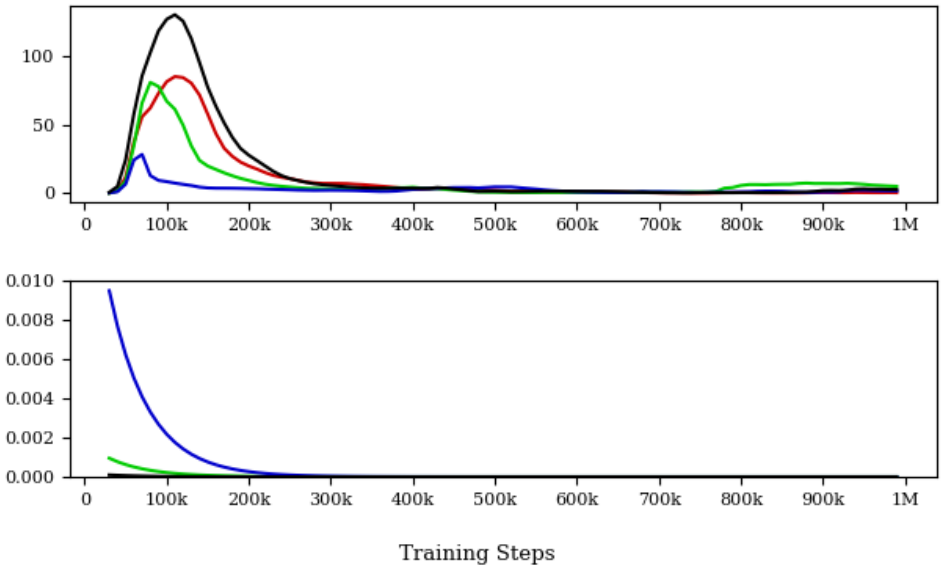

Figure 1: (Top) Random batch epistemic uncertainty and (bottom) epistemic weight during training for Agent+0 (red), Agent-2 (black), Agent-3 (green), and Agent-4 (blue).

transition from 100k to 200k steps. The area with the strongest reduction, the bottom left, is also an area that would be associated with a stronger reward. In contrast, for Agent-3 the strongest reduction was in the top right, which was an area with high epistemic uncertainty at 100k steps, and an area that reaching would not provide a strong reward signal otherwise. From 200k to 300k steps for Agent-3 we continue to see reductions in areas with previously higher uncertainty but also start to see a general reduction in epistemic uncertainty around the 0-degree longitude line. This is reasonable behavior for the agent as the epistemic weighting after 300k steps is becoming minimal, thus even Agent-3 will start reward seeking behavior.

The results from this limited study are promising but need additional work to better understand the effects of using epistemic uncertainty during training. For example, while we omit diagrams, we noticed during training that if we set the initial epistemic uncertainty weight to a value of 1, we often saw divergent behavior of agents during training. We expect this was because adding epistemic uncertainty has a dual but potentially conflicting goal: to both reduce and maximize epistemic uncertainty. Specifically, the agent wants to drive the system to high epistemic uncertainty. When stable, this generally causes epistemic uncertainty to see an overall reduction because when the agent visits a given state it now appears in the training set, allowing the critic ensemble to reach a consensus conclusion on its reward. However, we found empirically that if this drive is too strong, the agent can drive the epistemic uncertainty to such a large value that the agent would not recover, at least in the one million steps we observed.

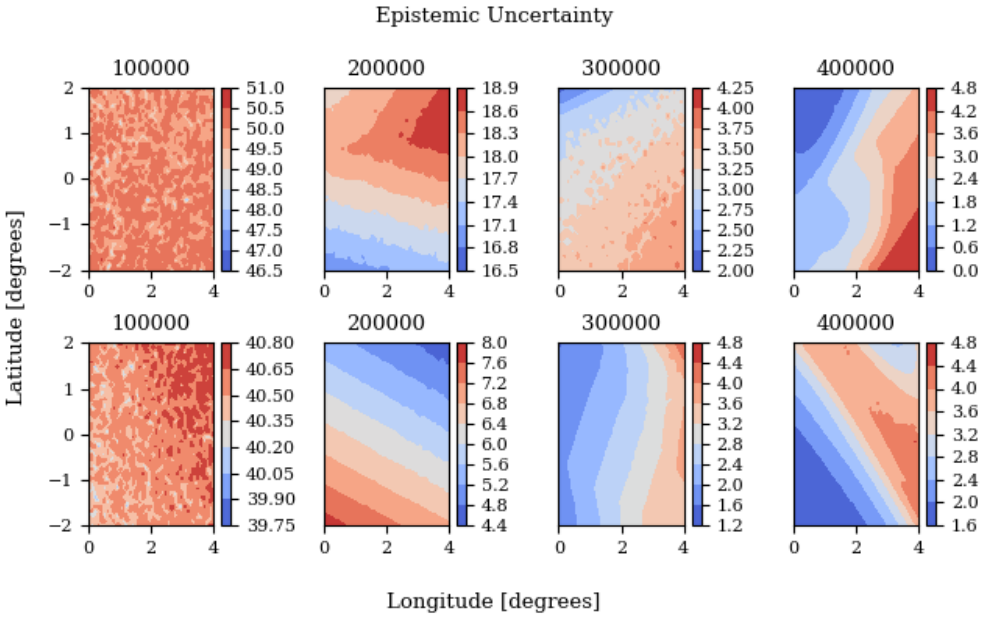

Figure 3: Epistemic uncertainty heat maps for Agent+0 (top) and Agent-3 (bottom) after 100k, 200k, 300k, and 400k training steps.

Deployment Scenario Results

During deployment it may be useful to utilize the uncertainty to deactivate and activate the model only when it is confident in its predictions. This can be done for both epistemic and aleatoric uncertainty. For this example, we utilize aleatoric uncertainty which can be visually seen by the heatmap showcased in the background in Figure 3. This

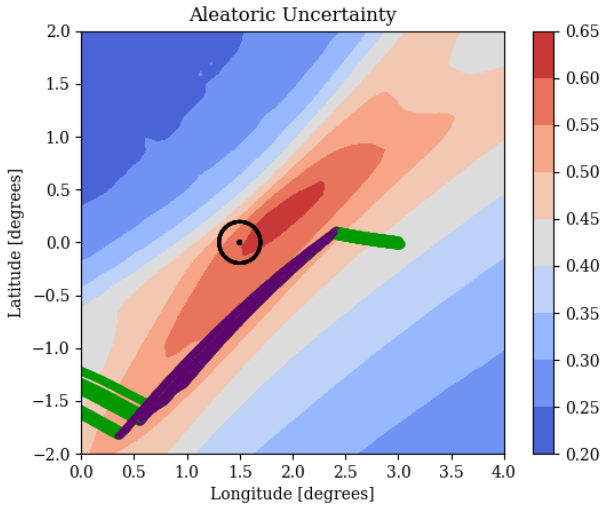

Figure 2: Aleatoric uncertainty heatmap with the path the agent follows. The agent takes actions according to the policy (green) and then switches to heuristic (purple) in regions of high uncertainty. The black circle represents the center of the detector.

heatmap was generated by taking the uncertainty on the Q values and averaging over all possible orientations to create a representation of the uncertainty based solely on latitude and longitude for the sake of visualization.

We set a threshold of uncertainty of 0.6 normalized over all uncertainty values in the state space. We then evaluated the uncertainty at each step of the model and activated a "safe" equivalent model that can navigate in these regions of high uncertainty. For the sake of this demonstration, the "safe" model followed the gradient of uncertainty until a region of low uncertainty was reached. When the model was above the threshold this "safe" model was activated and in regions below this threshold, the RL trained model was active. As shown, we are able to utilize this uncertainty to give the pilots more informed control over the aircraft during deployment. This provides additional opportunity to implement safety guards and checks on reinforcement learning to prevent the model from taking unpredictable or undesirable behavior due to a lack of confidence in its predictions. In the figure the heuristic is visualized as the purple section of the trajectory and the reinforcement learning model is visualized as the green section of the trajectory.

When deploying a model, a frequent concern is the trustworthiness of it and how can you quantify those values. These estimates of uncertainty can be utilized in two ways as shown above. The first was to provide a heatmap of the uncertainty to the pilots even before the aircraft is launched. The second is to provide on demand uncertainty estimations based on the current actions being output by the model. In order to calculate this uncertainty, it is a simple querying of the actor for the action and plugging that action back into the critic to get the estimated uncertainty on that action given the current state. This can also be fed back into the pilot to inform at runtime the confidence in the model.

CONCLUSION AND FUTURE WORK

In this work we discuss and provide small studies on the utility of uncertainty estimations provided by distributional and ensemble RL methods. We focus on how the warfighter can benefit from these estimates both in training agents and during deployment. While more work is required, this work helps motivate and demonstrate that the quantification of uncertainty provides meaningful benefits. These benefits are especially useful in aviation and navigation, where sensitivity to risk and the cost of failure are very large.

REFERENCES

Abdar, M., Pourpanah, F., Hussain, S., Rezazadegan, D., Liu, L., Ghavamzadeh, M., . . . Nahavandi, S. (2021). A review of uncertainty quantification in deep learning: Techniques, applications and challenges. *Information Fusion*, 243-297.

Bellemare, M. G., Dabney, W., & Munos, R. (2017). A Distributional Perspective on Reinforcement Learning. *34th International Conference on Machine Learning*, (pp. 449-458).

Berner, C., Brockman, G., Chan, B., Cheung, V., Dębiak, P., Dennison, C., . . . Zhang, S. (2019). Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv*.

Bryant, M., Gleason, J., & Macallister, A. (2024). Uncertainty Aware Distributional Ensemble Reinforcement Learning for Flight Control. *Interservice/Industry Training, Simulation and Education Conference.* Orlando.

Cao, Z., Jiang, K., Zhou, W., Xu, S., Peng, H., & Yang, D. (2023). Continuous improvement of self-driving cars using dynamic confidence-aware reinforcement learning. *Nature*, 145-158.

Duan, J., Guan, Y., Li, S. E., Ren, Y., Sun, Q., & Cheng, B. (2022). Distributional Soft Actor-Critic: Off-Policy Reinforcement Learning for Addressing Value Estimation Errors. *IEEE Transactions on Neural Networks and Learning Systems, 33*(11), 6584-6598.

Eriksson, H., Basu, D., Alibeigi, M., & Dimitrakakis, C. (2022). SENTINEL: taming uncertainty with ensemble based distributional reinforcement learning. *38th Conference on Uncertainty in Artificial Intelligence*, (pp. 631-640).

Hoel, C.-J., Wolff, K., & Laine, L. (2023). Ensemble Quantile Networks: Uncertainty-Aware Reinforcement Learning With Applications in Autonomous Driving. *IEEE Transactions on Intelligent Transportation Systems*.

Hüllermeier, E., & Waegeman, W. (2021). Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. *Machine Learning*, 457-506.

Ibar, J., Tan, J., Finn, C., Kalakrishnan, M., Pastor, P., & Levine, S. (2021). How to Train Your Robot with Deep Reinforcement Learning; Lessons We've Learned. *arXiv*.

Jebessa, E., Olana, K., Getachew, K., Isteefanos, S., & Mohd, T. K. (2022). Analysis of Reinforcement Learning in Autonomous Vehicles. *2022 IEEE 12th Annual Computing and Communication Workshop and Conference.* IEEE.

Kiran, B. R., Sobh, I., Talpaert, V., Mannion, P., Sallab, A. A., Yogamani, S., & Pérez, P. (2022). Deep Reinforcement Learning for Autonomous Driving: A Survey. *IEEE Transactions on Intelligent Transport Systems*, 4909-4926.

Krupani, R., Gleason, J., Bryant, M., Watson, T., & Macallister, A. (2025). Autonomous Aerospace Systems with an Understanding of Risk and Uncertainty using Distributional Ensemble Gaussian Mixture Model Reinforcement Learning. *AIAA.* Las Vegas.

Puterman, M. L. (2005). *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* Wiley-Interscience.

Schrittwieser, J., Antonoglou, I., Hubert, T., Sifre, L., Simonyan, K., Schmitt, S., . . . Silver, D. (2020). Mastering Atari, Go, chess and shogi by planning with a learned model. *Nature*, 604-609.

Seres, P., Liu, C., & Kampen, E.-J. v. (2023). Risk-sensitive Distributional Reinforcement Learning for Flight Control. *IFAC-PapersOnLine*, 2013-2018.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., . . . Hassabis, D. (2017). Mastering the game of Go without human knowledge. *Nature*, 354-359.

Song, Y., Suganthan, P. N., Pedrycz, W., Ou, J., He, Y., Chen, Y., & Wu, Y. (2023). Ensemble reinforcement learning: A survey. *Applied Soft Computing*.

Sutton, R. S., & Bartow, A. G. (2018). *Reinforcement Learning: An Introduction.* MIT Press.

Toner, T., Saez, M., Tilbury, D. M., & Barton, K. (2023). Opportunities and challenges in applying reinforcement learning to robotic manipulation: An industrial case study. *Manufacturing Letters*, 1019-1030.

Toyota Research Institute Unveils Breakthrough in Teaching Robots New Behaviors. (2023, September 19). Retrieved from Toyota Newsroom: https://pressroom.toyota.com/toyota-research-institute-unveils-breakthrough-in-teaching-robots-new-behaviors/

Wang, L., Liu, J., Shao, H., Wang, W., Chen, R., Liu, Y., & Waslander, S. L. (2023). Efficient Reinforcement Learning for Autonomous Driving with Parameterized Skills and Priors. *arXiv*.

Williams, D. V., Gee, T., MacDonaland, B. A., & Liarokapis, M. (2024). CTD4 - A Deep Continuous Distributional Actor-Critic Agent. *arXiv*.