A Decision Metric for the Use of a Deep Reinforcement Learning Policy

Christina Selby [*1] **Edward Staley** [*1]

Abstract

Uncertainty estimation techniques such as those found in Osband et al. (2018) and Burda et al. (2019) have been shown to be useful for efficient exploration during training. This paper demonstrates that such uncertainty estimation techniques can also be used as part of a time-series based methodology for out-of-distribution (OOD) detection for an off-line model-free deep reinforcement learning policy. In particular, this paper defines a "decision metric" that can be utilized for determining when another decision-making process should be used in place of the deep reinforcement learning policy.

1. Introduction

Deployment of deep reinforcement learning policies in safety-critical systems will require the ability to assure a policy's performance. One aspect of this problem is the ability to observe change between the training environment and the operational environment. This is often called out-of-distribution (OOD) detection. Given an OOD metric, a decision rule based on risk-tolerance can be defined so that a trusted controller (human or otherwise) can replace the deep RL policy when appropriate. The OOD metric needs to be interpretable, so that the decision rule does not have to be application specific and is well-understood.

This paper uses an out-of-distribution metric derived from a modification of Random Network Distillation (Burda et al., 2019). The application for Random Network Distillation (RND) in the previous work was for efficient exploration in the training of a deep RL policy. The presented application uses a modified version of RND in order to provide a time-series of noisy un-interpretable OOD metrics that are processed in order to produce a smoother and interpretable metric via Kalman filtering.

We utilize Meta Arcade (Staley et al., 2021) for training our models and creating OOD environments for testing our methodology. Meta Arcade is a recently developed tool to easily define and configure custom 2D arcade games that share common visuals, state spaces, action spaces, game components, and scoring mechanisms. In particular, we present experimental results where a deep RL policy for Meta Arcade Breakout is trained in a default environment, but the policy is deployed in versions of Breakout with new background colors.

Deployment of a deep RL policy in a safety-critical application will require assurance beyond confirming than that the deployment domain is in distribution. Further work will need to be completed in order to address other sources of uncertainty. The presented methodology can be one component in a suite of tools designed to determine if the use of a deep RL policy is appropriate for a given environment and input state.

1.1. Related Work

The uncertainty estimation techniques used for deep RL uncertainty estimation have been motivated by work in uncertainty estimation for deep neural networks, or derived from work in efficient exploration during the training of deep RL policies. In Lütjens et al. (2019), Kahn et al. (2017), and Hoel et al. (2020) techniques such as MC-Dropout (Gal & Ghahramani, 2016), Bootstrapping (Efron, 1986), (Efron & Tibshirani, 1994), and Randomized Prior Functions (Osband et al., 2018) are used in order to obtain uncertainty estimates that are utilized for applications in autonomous driving. In Kahn et al. (2017), the uncertainty estimates are used as input to a collision-avoidance prediction capability so that collisions are avoiding during online training of the RL policy. In Lütjens et al. (2019), collision avoidance is also the goal and the uncertainty estimates are used as input into a Model Predictive Controller (MPC) for selecting the safest action with minimal cost. Both of these collision avoidance applications utilize model-based RL and the policies are being updated online. The concept of using uncertainty estimation as input into a decision rule can be found in Hoel et al. (2020), where an uncertainty estimate for a given action is calculated and compared to a safety

[*]Equal contribution [1]Johns Hopkins Applied Physics Laboratory, Laurel, MD. Correspondence to: Christina Selby <christina.selby@jhuapl.edu>, Edward Staley <edward.staley@jhuapl.edu>.

ICML 2022 Workshop on Responsible Decision Making in Dynamic Environments, Baltimore, Maryland, USA, 2022. Copyright 2022 by the author(s).

threshold. This safety threshold is determined by an analysis of test episodes within the training distribution. The value of the safety threshold is directly tied to the training, and thus not interpretable as a stand-alone value. The decision is made at each timestep, using only the uncertainty value at that timestep. Our work will use a time-series of uncertainty estimates to inform an OOD metric that has a value that has a statistical interpretation independent of environment.

2. Methodology

2.1. Modified Random Network Distillation

Random network distillation (RND) (Burda et al., 2019) is a technique to encourage exploration during the training of deep reinforcement learning policies. In RND, a frozen "prior", a randomly initialized fully connected deep neural network, is defined as a function over the state space. A "distiller" network is then trained to predict the vector-valued output of this prior for visited states, by using the mean-squared error between the two networks as a loss function. The output of the RND is the difference between the prior and the distiller. Therefore, novel states have a non-zero RND output.

In this work, we re-purpose RND as an uncertainty mechanism to be used during deployment rather than during training. If a trained RL agent is deployed in a setting that is out-of-distribution from its training regime, we hypothesized that the RND will output values much further from zero than what is typical on training data. This revised RND training approach is described in section 2.2. Our experimental setup and an analysis of the RND output for in-distribution versus out-of-distribution environments are provided in section 3.3. We define an out-of-distribution (OOD) metric at episode step t as the mean of the components of the vector-valued output of the RND, and denote this score o_t.

One concern with the use of the RND model for out-of-distribution indication is that the RND model may extrapolate well for out-of-distribution data that is close to in-distribution data. To increase sensitivity, we modified the RND approach by introducing a high-frequency term to the prior network. The output of the RND's prior was modified to be $\sin(\beta p)$ instead of p. The choice of β will be discussed in section 3.3. This modification makes training the distiller network more challenging for in-distribution data, but also makes extrapolation more difficult for out-of-distribution data.

2.2. RL Architecture

To use our modified RND during deployment, we simply train it alongside the PPO agent for later use. During training, the only interaction between the two algorithms is that the states used to train the RND are exactly those collected by the PPO agent rollouts. It is not until after training that the RND is utilized, at which point it provides a measure of uncertainty for the trained PPO agent with regards to any encountered states.

In addition to the sinusoidal prior described previously, we make several changes to the RND/PPO architecture from its initial publication in Burda et al. (2019). We use a common feature extraction CNN for policy network, value network, and the RND networks, as opposed to separate CNN extractors for the RND. Our rationale is that we are primarily interested in detecting changes during deployment that have a direct impact on the agent's behavior, which is best captured by the image features learned by the policy (as opposed to those from a randomly initialized CNN). Our RND prior and distiller network consist of MLPs with two hidden layers of size 256, with an output of 256. The RND takes as input the features extracted by the PPO network, but does not propagate gradients back into the this extractor during training. This introduces a moving input space to the RND, albeit one that moves very slowly.

A diagram of our architecture is shown in Figure 1.

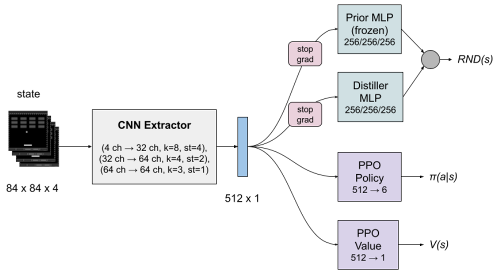
state
84 x 84 x 4
CNN Extractor
(4 ch → 32 ch, k=8, st=4),
(32 ch → 64 ch, k=4, st=2),
(64 ch → 64 ch, k=3, st=1)
512 x 1
stop grad
stop grad
Prior MLP (frozen) 256/256/256
Distiller MLP 256/256/256
RND(s)
PPO Policy 512 → 6
π(a|s)
PPO Value 512 → 1
V(s)

Figure 1: Architecture for parallel training of PPO (purple) and RND (green), which itself consists of a frozen prior network and learned distiller. A feature extractor learned under PPO is also used as a feature extractor for the RND networks, but does not receive weight updates from the RND.

2.3. Calibration

Given a trained modified RND model and policy, we run M episodes in the training environment and consider the sets of out-of-distribution metrics, $O_j = \{o_{j,t} : t = 1, ..., L_j\}$, where $j = 1, ..., M$, and L_j is the length of the jth episode. We calculate the mean of O_j and denote it $\bar{o}_j$. We then consider the set of means $\{\bar{o}_1, ..., \bar{o}_M\}$. We estimate the mean and standard deviation of this normal distribution with the sample mean. The sample mean and standard deviation are denoted $\bar{\mu}$ and $\bar{\sigma}$.

2.4. Metric Calculation

We have defined an out-of-distribution metric that can be evaluated at each timestep of an episode. These measurements are quite noisy because it is impossible to train on every possible input. Therefore, the decision to stop using the deep RL policy should not be based on one measurement. We will call the "decision metric" a metric that is derived from the time-series of out-of-distribution metrics and used for determining when it is appropriate to deploy an alternate controller. The decision metric is interpretable, with the same interpretation for any RL environment.

We utilize a Kalman filter in order to estimate $\bar{o}_E$ for a given episode E as we are obtaining out-of-distribution metrics from the episode, $o_{E,t}$. The Kalman filter approach is used as a way to smooth measurements, but other smoothing techniques could also be used. We denote the estimate of $\bar{o}_E$ after t measurements as $\widehat{o_{E}}_t$. The Kalman filter is initialized with $o_{E,1}$ and an initial variance of $(6\bar{\sigma})^2$. We calculate the number of standard deviations the estimated mean is from the in-distribution sample mean $\hat{\mu}$. That is, we calculate

$$z_{E,t} = \frac{|\widehat{o_{E}}_t - \hat{\mu}|}{\bar{\sigma}}. \quad (1)$$

The decision metric at time t is defined

$$d_{E,t} = 2(1 - cdf(z_{E,t})), \quad (2)$$

where the cumulative distribution function is with respect to $N(0,1)$. Therefore, the decision metric can be interpreted as the percent of episodes we expect to have mean out-of-distribution metric further away from the mean of the means of in-distribution episodes. This is essentially an anomaly detector for the episode.

3. Experimental Setup

3.1. Meta Arcade Environment

Meta Arcade (Staley et al, 2021) is a configurable suite of arcade environments for deep reinforcement learning that exposes access to the underlying game parameters and appearance. This configurability is useful when constructing multiple related tasks, or when studying environment shift, as in this work. We use Meta Arcade to first train an agent on a game with a known color scheme, and then deploy the agent under shifted appearance. Although Meta Arcade also supports shifts in task dynamics or task definition, these may not result in large visual changes and thus could be very difficult to detect with a convolutional feature extractor. We leave those types of MDP alterations to future work (see Section 5).

3.2. Training Details

The architecture discussed in Section 2.2 was trained using PPO to play Meta Arcade Breakout. We used the PPO implementation from Stable-Baselines 3 (Raffin et al., 2021), modified to additionally train the RND and modified RNDs with each minibatch. We trained this model for 20 million steps using 8 parallel workers. The calibration step as described in section 2.4 was performed using 250 episodes of the default environment.

3.3. Preliminary Statistical Analysis

In order for the decision metric to be successful as an out-of-distribution indicator, it must be the case that the out-of-distribution time-series metrics come from distributions that differ from the distribution obtained from the default background. We calculated the mean over each of 100 episodes for each of the gray scale colors defined by $0, 15, 30, 45, 60, 75, 90$ in order to get a distribution of out-of-distribution scores for each of these backgrounds. The corresponding environment states are visualized in Figure 2. In Figure 3, we have a plot of these distributions for the RND network. We can observe in this plot that the distributions are not sufficiently separated. Thus, we hypothesize that the RND network is extrapolating too well when the gray-scale values are close to 0. In order to overcome this problem, we introduced a sine function as described in section 2.1. In Figure 4, we can observe the distributions for $\beta = 1, 2, 3, 4, 5, 6, 8, 10$. We can observe that there is significantly more separation in the distributions. We observe that $\beta = 3$ provides the best separation for the change in distribution.

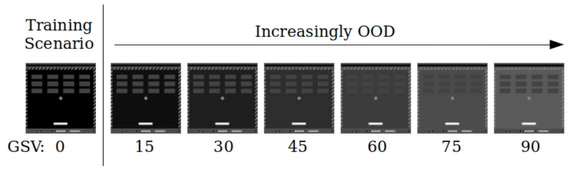

Figure 2: Gray-scaled frames of Meta Arcade with varying background color. Our agents were all trained with a gray-scale value of 0 (left), and evaluated as this value was raised, resulting in increasingly out-of-distribution environments. Our RL agents received histories of four such frames as their input observations.

3.4. Experiments and Results

We utilized the modified RND with $\beta = 3$ and ran background color change experiments in Meta Arcade in order to calculate the decision metric defined in section 2.4. Each episode we consider in the following is just one sample from the relevant distribution discussed in section 3.3. First, we observe an example of the performance of the mean out-of-

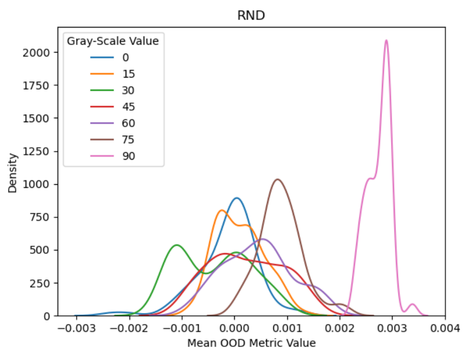

Figure 3: Distributions of episode mean of out-of-distribution metrics for varying gray-scale values without using a sinusoidal prior.

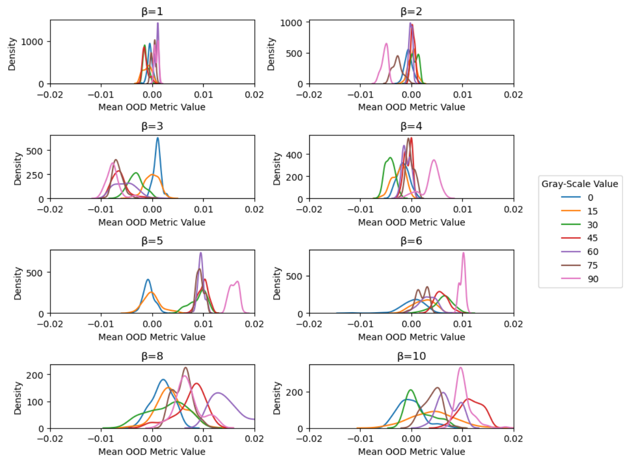

Figure 4: Distributions of episode mean out-of-distribution metrics for varying gray-scale values for a collection of modified RND models.

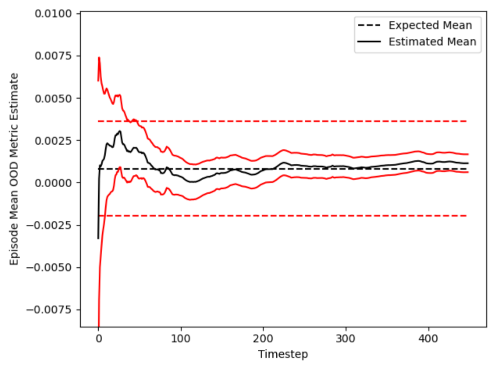

Figure 5: Example of estimate of the mean OOD metric for the default environment. (The corresponding decision metric would be close to 1.) We observe that the estimated mean for this episode is statistically consistent with the default environment since the mean estimate stays within three standard deviation of the mean.

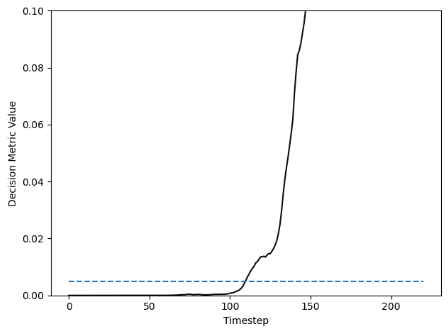

Figure 6: Example of the decision metric during an episode for gray scale value 15. At approximately timestep 50, the decision to switch the RL policy to another controller would be made at the confidence level .005.

distribution metric estimator in the default environment; see Figure 5. We observe that this episode is consistent with the statistics of the default environment. For the gray scale color 15, we should not expect excellent performance for the decision metric due to the results of the analysis we did in section 3.3. In Figure 6, we provided a case where the decision metric does indicate out-of-distribution. For the other gray scale values, the decision metric clearly indicated out-of-distribution, as we expect from the analysis in section 3.3.

4. Conclusion

The presented analysis demonstrates a case for which modification of random network distillation yields better separation between out-of-distribution uncertainty scores and in-distribution uncertainty scores than standard random network distillation. The resulting modified random network distillation model can be used as input into a decision metric calculation useful for indicating when to “turn off” the RL policy.

5. Future Work

A technique for predicting the effect of the RND modification is not available, but utilizing an ensemble of modified random network distillation models could provide a reasonable technique for out-of-distribution detection, even close to the case where the out-of-distribution scenario is very similar to the in-distribution scenario. We acknowledge that we are only considering changes to the environment for which we can analyze and chose our parameter β using this analysis. In practice, there are many different ways that the environment could change and impact the performance of an RL policy. The Meta Arcade environment provides a way to control the environment and experiment, but implementation of RL in the "real world" will introduce changes in the environment that we cannot be predicted or modelled ahead of time.

References

Burda, Y., Edwards, H., Storkey, A., and Klimov, O. Exploration by random network distillation. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=H1lJJnR5Ym.

Efron, B. Discussion: Jackknife, bootstrap and other resampling methods in regression analysis. *The Annals of Statistics*, 14(4):1301–1304, 1986. ISSN 00905364. URL http://www.jstor.org/stable/2241457.

Efron, B. and Tibshirani, R. J. *An Introduction to the Bootstrap*. CRC press, 1994.

Gal, Y. and Ghahramani, Z. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In Balcan, M. F. and Weinberger, K. Q. (eds.), *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pp. 1050–1059, New York, New York, USA, 20–22 Jun 2016. PMLR. URL https://proceedings.mlr.press/v48/gal16.html.

Hoel, C.-J., Wolff, K., and Laine, L. Tactical decision-making in autonomous driving by reinforcement learning with uncertainty estimation. In *2020 IEEE Intelligent Vehicles Symposium (IV)*, pp. 1563–1569, 2020. doi: 10.1109/IV47402.2020.9304614.

Kahn, G., Villaflor, A., Pong, V., Abbeel, P., and Levine, S. Uncertainty-aware reinforcement learning for collision avoidance. *CoRR*, abs/1702.01182, 2017. URL http://arxiv.org/abs/1702.01182.

Lütjens, B., Everett, M., and How, J. Safe reinforcement learning with model uncertainty estimates. pp. 8662–8668, 05 2019. doi: 10.1109/ICRA.2019.8793611.

Osband, I., Aslanides, J., and Cassirer, A. Randomized prior functions for deep reinforcement learning. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper/2018/file/5a7b238ba0f6502e5d6be14424b20ded-Paper.pdf.

Raffin, A., Hill, A., Gleave, A., Kanervisto, A., Ernestus, M., and Dormann, N. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021. URL http://jmlr.org/papers/v22/20-1364.html.

Staley, E. W., Ashcraft, C., Stoler, B., Markowitz, J., Vallabha, G., Ratto, C., and Katyal, K. D. Meta arcade: A configurable environment suite for meta-learning, 2021.