AppliedMath

MDPI

Article

Improved Exploration in Reinforcement Learning Environments with Low-Discrepancy Action Selection

Stephen W. Carden *, Jedidiah O. Lindborg and Zheni Utic

Department of Mathematical Sciences, Georgia Southern University, Statesboro, GA 30458, USA; jl26083@georgiasouthern.edu (J.O.L.); zutic@georgiasouthern.edu (Z.U.)
* Correspondence: scarden@georgiasouthern.edu

Abstract: Reinforcement learning (RL) is a subdomain of machine learning concerned with achieving optimal behavior by interacting with an unknown and potentially stochastic environment. The exploration strategy for choosing actions is an important component for enabling the decision agent to discover how to obtain high rewards. If constructed well, it may reduce the learning time of the decision agent. Exploration in discrete problems has been well studied, but there are fewer strategies applicable to continuous dynamics. In this paper, we propose a Low-Discrepancy Action Selection (LDAS) process, a novel exploration strategy for environments with continuous states and actions. This algorithm focuses on prioritizing unknown regions of the state-action space with the intention of finding ideal actions faster than pseudo-random action selection. Results of experimentation with three benchmark environments elucidate the situations in which LDAS is superior and introduce a metric for quantifying the quality of exploration.

Keywords: reinforcement learning; low-discrepancy sequence; Markov decision process

check for updates

Citation: Carden, S.W.; Lindborg, J.O.; Utic, Z. Improved Exploration in Reinforcement Learning Environments with Low-Discrepancy Action Selection. *AppliedMath* **2022**, *2*, 234–246. https://doi.org/10.3390/appliedmath2020014

Academic Editor: Boaz Lerner

Received: 6 April 2022
Accepted: 5 May 2022
Published: 16 May 2022

Publisher's Note: MDPI stays neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Copyright: © 2022 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In most statistics and machine learning applications, a greater value is placed on new, unseen information. There is a strong intuition that obtaining new data will be more valuable than data that has already been observed. In discrete settings, without-replacement sampling prioritizes the selection of unobserved individuals, and is superior in many aspects to with-replacement sampling. In machine learning, applications of without-replacement sampling include choosing ensemble classifiers [1], constructing low-rank approximations of large matrices [2], kernel embeddings [3], computational learning theory [4], and as a general purpose tool in evaluating data analysis applications [5]. For example, a strategy called random reshuffling has been shown to improve convergence of stochastic gradient descent [6].

Reinforcement learning is a subdomain of machine learning concerned with learning optimal behavior in a potentially stochastic environment from data obtained by interacting with that environment. A key element of the reinforcement learning problem is exploration: selecting actions that are not currently considered optimal for the given state in order to learn more about the environment in the hope of finding superior actions. Exploration strategies are a large portion of the reinforcement learning literature and new approaches continue to be introduced. A common component of many is uniform selection of actions using pseudo-random numbers that are independent of previous actions. It is possible that this selection method may choose an action which is similar to previously selected actions, when intuition suggests something new and as dissimilar as possible is desirable. Some strategies, such as optimistic value initialization [7], will try to avoid this and force selection of previously unseen actions, but many strategies do not.

When the action space is discrete, variants of without-replacement sampling can be applied in a straightforward way [8]. Surprisingly, recent research has shown that

when using a simple metric such as expected time to reach a goal state, it is possible to construct environments for which without-replacement is inferior [9]. However, these environments are somewhat pathological and unlikely to be encountered by chance, so without-replacement sampling continues to be a promising avenue of research for improving reinforcement learning.

When state and action spaces are continuous, without-replacement sampling does not have the same meaning, but it is still heuristically desirable to try actions that are as different from previously seen actions as possible. With this in mind, a reinforcement learning agent that prioritizes obtaining information that is dissimilar to previously obtained information during the exploration stage may produce superior results during the learning stage.

In this paper, we introduce a new method for environments in which both states and actions are continuous: Low-Discrepancy Action Selection (LDAS). LDAS takes account of previously visited state–action pairs and tries to select an action that is as dissimilar from previous state-actions as possible. We also introduce a metric for comparing the discrepancy of exploration strategies relative to each other, and perform experiments in three reinforcement learning environments that compare three action selection methods.

Section two introduces the mathematical framework that underlies reinforcement learning. Section three defines the concept of discrepancy and introduces metrics for comparison. Section four combines the two previous concepts and defines an algorithm for selecting actions in a low-discrepancy manner. Section five presents experimental results, and section six concludes the paper.

Literature Review

A review of four action-selection strategies for continuous dynamics [10] includes one method based on discretization of the action space and three grouped as optimization techniques based on gradient descent [11,12], and Newton's Method [13]. By becoming progressively finer-grained in selective regions of the action space, discretization strategies can approximate continuous actions efficiently. For example, Sequential Monte Carlo (SMC) methods [14] samples actions from a prior distribution and resamples according to an estimated utility, thus promoting a thorough exploration of the most favorable action-space areas and permitting SMC-learning to locate real continuous actions. The method is model-free and may learn to follow stochastic policies.

The previous methods are based on action-value functions and value iteration. The other paradigm in reinforcement learning, policy iteration, is also represented in literature for continuous processes. Continuous-action Approximate Policy Iteration (CAPI) [15] can search the set of policies (functions from the state to action space) directly. Two adaptive models are suggested to approximate the value function, and simulations demonstrate that the model converges to a near-optimal policy in a few iterations.

Recent papers approach continuous dynamics in innovative ways, such as sampling based on the likelihood ratio between the action probabilities of a target policy and the data-producing behavior policy [16]. The behavior policy action probabilities are replaced with their maximum likelihood estimators. These estimators reduce Monte Carlo sampling variance, result in faster learners for policy gradient algorithms, and are shown to be consistent.

2. Markov Decision Process Preliminaries

Reinforcement learning is formalized using the mathematical structure of a Markov Decision Process (MDP) [7]. A decision-making entity called an agent observes a system or environment in discrete time. We call each point in time a "step". At time t, the agent observes the environment to be in state $S_t \in \mathcal{S}$, where $\mathcal{S}$ represents the set of all possible states. The agent makes a decision by choosing an action $A_t \in \mathcal{A}(S_t)$, where $\mathcal{A}(S_t)$ is the set of all allowable actions from state S_t, and is a subset of the set of all possible actions $\mathcal{A}$. Then, the system transitions to a new state S_{t+1} and the agent receives a numeric reward $R_t \in \mathcal{R}$, where $\mathcal{R} \subset \mathbb{R}$ is the set of all possible rewards. Consider first the case that $\mathcal{S}$,

$\mathcal{A}$, and $\mathcal{R}$ are discrete. Transition probabilities and reward distributions are governed by the dynamics function $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \to [0, 1]$ where $p(s', r|s, a)$ is the probability of transitioning to state s' and receiving reward r when the environment is in state s and the agent chooses action a. The objective is to find a policy: a function $\pi : \mathcal{S} \to \mathcal{A}$ that maximizes some measure of long-term reward.

A fundamental issue is how the agent should select actions, and it is commonly known as the exploration–exploitation dilemma. Should the agent exploit its current knowledge of the system dynamics by selecting the action thought to be optimal in the current state, or should the agent explore new actions in hope of finding one better than what is already known? This paper will focus on exploration and how to select actions so that the state-action space is investigated evenly and completely.

If states are not discrete but continuous, the state space needs to be equipped with a sigma-algebra $\sigma(\mathcal{S})$ and the resulting dynamics function needs to be measurable. The primary objective of this paper is efficient exploration of the environment, and we limit the scope of the paper to that extent.

3. Low-Discrepancy Sequences

Intuitively, the discrepancy of a sequence is a measure of deviation from an ideal uniformly distributed sequence. High-discrepancy sequences exhibit clustering and large empty regions, while low-discrepancy sequences have evenly spaced points and are not too close. Having a measure of discrepancy is essential for comparing action selection strategies. It will help one determine if selection is sufficiently uniform (as would be compatible with the "without-replacement" heuristic) or if some state-actions are repeated (as could happen under "with-replacement" sampling.)

A simple and well-known low-discrepancy sequence is the van der Corput sequence [17], constructed by reversing the sequence of natural numbers of a certain base across the decimal point. The multidimensional extension, the Halton sequence [18], is obtained by letting each dimension be a van der Corput sequence with co-prime bases. Figure 1 displays 300 points from a Halton sequence and pseudo-random uniform sequence side-by-side. Under pseudo-random generation, the observed density is uneven, with some regions containing many points and some containing few. In contrast, the Halton sequence is closer to an ideal uniform density.

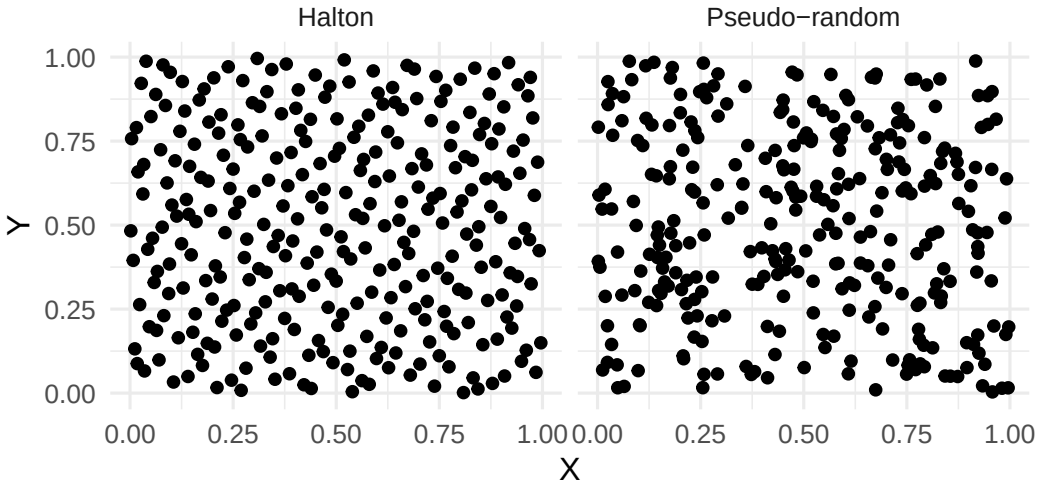

Figure 1. A visual comparison of 300 points from a Halton sequence and a pseudo-randomly generated sequence.

Unfortunately, these sequences are not readily applicable to action selection for reinforcement learning. They require that each dimension of a point can be chosen freely. However, in reinforcement learning, only the action dimensions are freely choosable, while the state dimensions are a consequence (either stochastically or deterministically) of the

system dynamics. Before discussing development of a method for choosing actions in a low-discrepancy manner, it will be beneficial to consider how to measure discrepancy.

3.1. Classic Discrepancy Measure

A classic measure of discrepancy is based on the difference between the proportion of a sequence in an interval and its length. It is particularly useful for theoretical analysis of sequences [19]. We first give the definition for one dimension. Let $\{x_n\}_{n=1}^N$ be a finite sequence taking values in the interval $[0,1]$ and $E \subseteq [0,1]$. Let $A(E;N)$ count the number of the points of $\{x_n\}_{n=1}^N$ contained in E. Then, the discrepancy of the sequence is

$$D_N(x_1,...,x_N) = \sup_{0\le\alpha<\beta\le 1} \left| \frac{A([\alpha,\beta);N)}{N} - (\beta-\alpha) \right|.$$

For multiple dimensions, the definition is generalized as follows. Let $I^k = [0,1]^k$ represent the k-dimensional unit hypercube, and let J run through all sub-intervals of I^k of the form $J = \{(x_1,...,x_k) \in I^k : \alpha_i \le x_i < \beta_i \quad \text{for} \quad 1 \le i \le k\}$ and let λ represent k-dimensional Lebesgue measure, then:

$$D_N(x_1,...,x_N) = \sup_J \left| \frac{A(J;N)}{N} - \lambda(J) \right|.$$

The intuition behind this concept is that discrepancy is high when there are large hyper-rectangles that contain a small proportion of the total points, or small hyper-rectangles with a high proportion of points.

3.2. Coefficient of Variation Measure

An alternative method for calculating discrepancy is based on distance from each point to its nearest neighbor. On average, this distance should be as large as possible, and it should vary as little as possible. This ensures the density in a region is somewhat constant and points are spread evenly. Then discrepancy can be measured by the ratio of standard deviation to the mean, which is the well-known coefficient of variation (CV) applied to minimum distances.

One property of the coefficient of variation is scale invariance. For example, suppose a new sequence is created by multiplying every point in an existing sequence by one-half. In that case, the two sequences will have the same coefficient of variation, even though the second one does not fill the space as fully. This property is undesirable for measuring discrepancy. For this reason, we make a slight modification and square the mean. Under this change, if two sequences have identical distributions except for scale, the one with greater distance between the points will have lower discrepancy and be preferred. To keep terminology simple, we still refer to this as the CV measure of discrepancy.

$$CV_N(x_1,\ldots,x_N) = \frac{s_d}{\bar{d}^2} \text{ where } d_i \text{ is distance from } x_i \text{ to closest } x_j, i \ne j. \tag{1}$$

For the experiments forthcoming in this paper, we use the CV measure. We found it easier to calculate and more closely related to our intuition of discrepancy for sequences when visualized. Figure 2 shows the difference between low- and high-discrepancy sequences under the CV measure grows rather than shrinks as the number of points increases. To create this figure, ten Halton and ten pseudo-random two-dimensional sequences of length 500 were generated. The discrepancies D_N and CV_N of the first N points were calculated for each sequence from $N = 5$ to $N = 500$. The average discrepancy of the ten sequences is calculated and plotted.

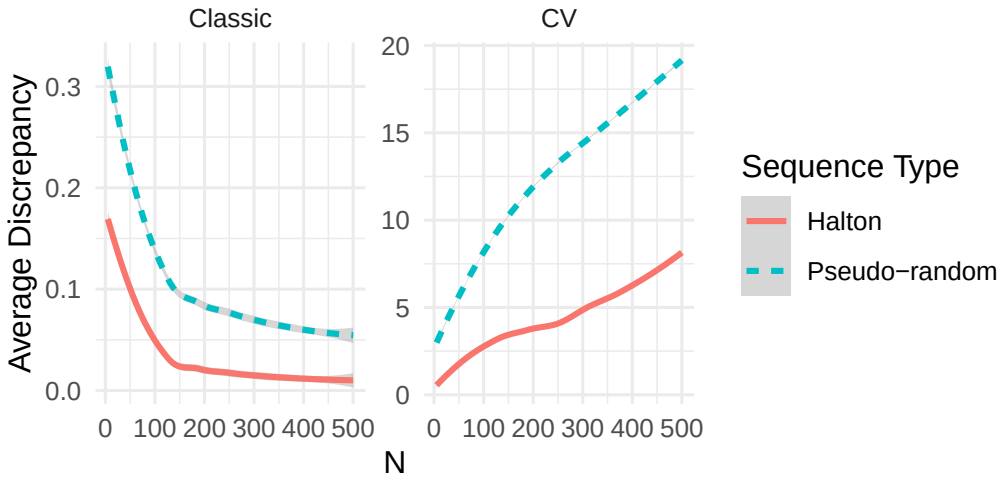

Figure 2. A comparison of discrepancy measures for a Halton sequence and a pseudo-randomly generated sequence.

4. Low-Discrepancy Action Selection

This section combines the concepts of exploration in an MDP and low-discrepancy sequences, with the goal of exploring so that discrepancy in the state-action space is minimized. This is not as simple as directly using a Halton or similar sequence to choose actions because it does not account for the state dimensions, which are not free for the agent to choose. Instead, a new method is needed which can observe the history of state-actions visited and the present state, and use this information to select an action such that:

- The new state–action pair is as dissimilar from previous ones as possible;
- Boundaries of the space have small but non-zero selection probabilities so that extreme actions can be tested without being overrepresented;
- Action selection is reasonably computationally efficient.

The basic idea behind our approach is to place a candidate action randomly, but move it so that some measure of distance from previously visited state-actions is maximized via numerical methods. Many variations and specific measures were tested, such as each previous point emitting a field modeled on electrical charges to repel the new point, or gradient descent to minimize a function defined as the cumulative sum of multivariate Gaussian densities centered at all previous points. These approaches produced unsatisfactory discrepancies both visually and numerically.

The method we eventually arrived at is simple: the agent finds the previous state-action that is closest to the present, and moves the action component directly away according to a slowly decaying learning rate. This continues until the action stabilizes or the boundary of the action space is reached. This procedure, termed LDAS (Low-Discrepancy Action Selection), is described in Algorithm 1. Notice that when determining the nearest point, both states and actions are used, but only the action component is updated.

This pseudo code represents the core of the algorithm. In practice, a few minor alterations were made to improve performance. For example, early testing revealed that too many points were placed at the boundaries of the action dimensions. We reasoned that new actions were being pushed outwards from points in the interior of the space, with no counterbalancing pressure from the exterior of the space. The issue was corrected by temporarily placing so-called "phantom points" just beyond the boundaries. The distance of phantom points from the edge is a decreasing function of the sample size. Another simplification was normalizing the action space to the unit hypercube before applying the algorithm, then transforming back to the original scale at the end. Finally, to limit memory and computational burden, we placed a cap on the number of state-actions stored in *sa_hist*. The default value is 500, and was used in all experiments in

this paper. Exact details are available in the complete code in the GitHub repository at https://github.com/swcarden/LDAS (accessed on 10 May 2022).

Algorithm 1: The LDAS algorithm.

Input: A threshold $\epsilon > 0$;
integer number of iterations $n \geq 1$;
learning rate $r > 0$;
learning rate decay $0 < rd < 1$;
initial state s_0;
integer window $w \geq 1$ of recent actions against which to compare.
Output: A state-action history matrix *sa_hist*.
Initialize pseudo-random a_0;
Initialize *sa_hist* with (s_0, a_0);
for $i \leq n$ **do**
  keep_going $\leftarrow$ TRUE;
  Get state s_i from interacting with environment from (s_{i-1}, a_{i-1});
  Generate pseudo-random a_i;
  r_temp $\leftarrow r$;
  while *keep_going* **do**
    Sweep *sa_hist*, find distances between (s_i, a_i) and each (s_j, a_j) for $j < i$;
    Set a^* to be action from closest state–action pair;
    $a_i \leftarrow a_i + r\_temp * \frac{a_i - a^*}{||a_i - a^*||}$;
    if a_i is on or beyond the boundary of the action space **then**
      $a_i \leftarrow$ boundary value;
      keep_going $\leftarrow$ FALSE;
    end
    if *r_temp* $< \epsilon$ **then**
      keep_going $\leftarrow$ FALSE;
    end
    if $||a_i - a_{i-w}|| < \epsilon$ **then**
      keep_going $\leftarrow$ FALSE;
    end
    r_temp $\leftarrow$ *r_temp* $*$ *rd*;
  end
  Append (s_i, a_i) to *sa_hist*;
end

Figure 3 compares LDAS to pseudo-random uniform action selection. In this simple illustration, states (horizontal component) are chosen pseudo-randomly for both methods. On the left, LDAS chooses the action (vertical component) according to Algorithm 1. On the right, actions are chosen pseudo-randomly. The more even distribution of actions chosen by LDAS is clear.

Visual evaluation of LDAS in higher dimensions is difficult, so we turn to Equation (1). Data were generated according to the same conditions as in Figure 3, but allowing for an increased number of state and action dimensions, each running from one to three, with 100 points generated. After each timestep beginning with the fifth (as discrepancy is less meaningful for such short sequences), the discrepancy of the sequence was calculated and recorded. This entire process was repeated 100 times. Finally, the average discrepancy over the 100 runs was calculated at each timestep. The results are presented in Figure 4.

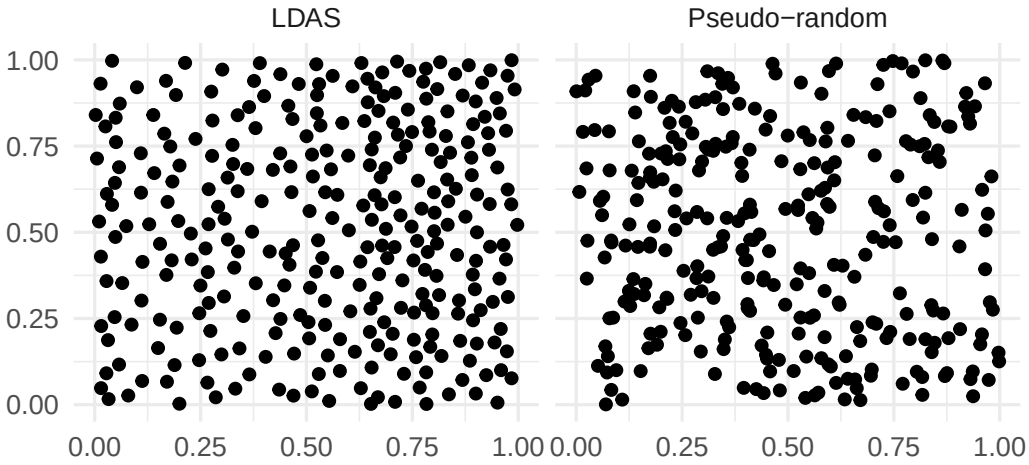

Figure 3. A visual comparison of 300 points from LDAS and pseudo-random sequences.

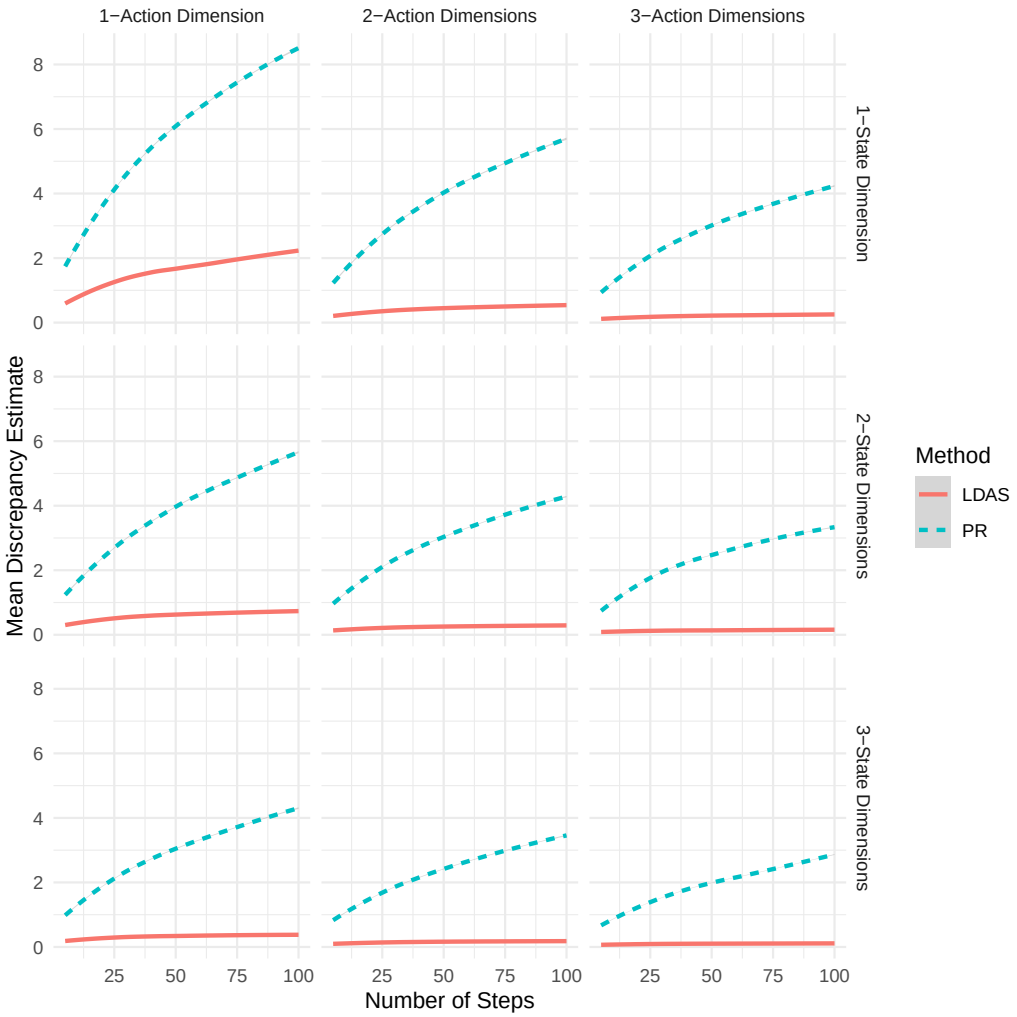

Figure 4. A comparison of discrepancy for pseudo-random and LDAS sequences for varying number of state and action dimensions.

Several things can be learned from this figure. First, we see that LDAS produces lower-discrepancy results than PR for any number of dimensions. Second, as the number of dimensions increases, discrepancy tends to decrease regardless of selection method. This is sensible as points are more spread out in larger spaces and are less likely to cluster. Third,

the difference between LDAS and PR decreases as the dimensionality increases, for the same reason as in the previous observation. Finally, while discrepancy for PR depends on the total number of dimensions, it does not matter if they are state or action dimensions. However, LDAS can attain lower discrepancy when more dimensions are assigned to actions. Observe that discrepancy under LDAS is lower for three-action and one-state dimensions compared to three-state and one-action dimensions.

5. Experimental Results

The experiments in the previous section can be seen as a trivial MDP in which states are randomly selected independent of actions. In this section, experiments are run on three common benchmark RL environments that allow for continuous actions.

In addition to comparing LDAS against pseudo-random uniform action selection, we also tested the Ornstein–Uhlenbeck (OU) method, which is inspired by Brownian motion and in discrete time is equivalent to an autoregressive time series with drift [20]. It purposely correlates actions and so, in a sense, is the opposite of LDAS, and is beneficial in some motion-based environments by introducing a sort of inertia. Letting μ be a drift parameter, $\sigma > 0$ be a noise parameter, $\theta > 0$ an inertia parameter, dt a time increment, and Z a standard normal random variable, the change in action a is governed by

$$da = \theta(\mu - a)dt + \sigma\sqrt{dt}Z.$$

The default values in the Python library are $\mu = 0$, $\sigma = 0.3$, and $\theta = 0.15$, and these are what we used.

Experiments were run in Python version 3.8.4 [21]. For reinforcement learning environments, we used the Gym library [22], version .17.2. For exploration strategies, we used the Garage library [23], version 2021.3.0. Data analysis and visuals were produced in R 4.1.1 [24] with the aid of the tidyverse package [25]. Simulations and analysis were completed on our university's high-performance computing cluster running CentOS Linux release 7.9.2009.

The Garage library contains classes EpsilonGreedyPolicy and AddOrnsteinUhlenbeckNoise. For uniform pseudo-random action selection, we used EpsilonGreedyPolicy with epsilon equal to one. To implement LDAS, we modified the EpsilonGreedyPolicy class to perform LDAS on exploration and set epsilon equal to one. The modification can be found on our GitHub page.

5.1. MountainCarContinuous-v0

The first Gym environment tested is MountainCarContinuous-v0 or simply Mountain Car, and is one of the classic RL environments. A car at the bottom of a valley is trying to reach the top of a mountain, but does not have enough power to drive directly up the mountain and must learn to build momentum by alternating directions. Observe Figure 5.

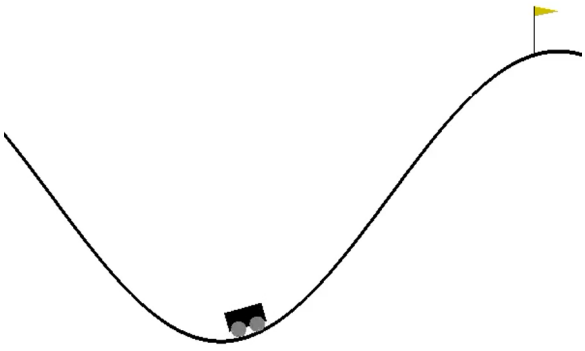

Figure 5. A visual representation of the MountainCarContinuous-v0 environment. Using position and velocity of the car as inputs, the agent must decide how to apply force with the goal of reaching the flag at the top of the hill.

The state space is two-dimensional, and consists of position and velocity. The simulation technically allows both components of the state space to be any real numbers, though realistically only values near the origin are feasible. The action space is the one-dimensional interval $[-1, 1]$, representing force from the engine.

5.2. LunarLanderContinuous-v2

The LunarLanderContinuous-v2 environment simulates a spacecraft that needs to land on the surface of the moon between two goal posts. Observe Figure 6.

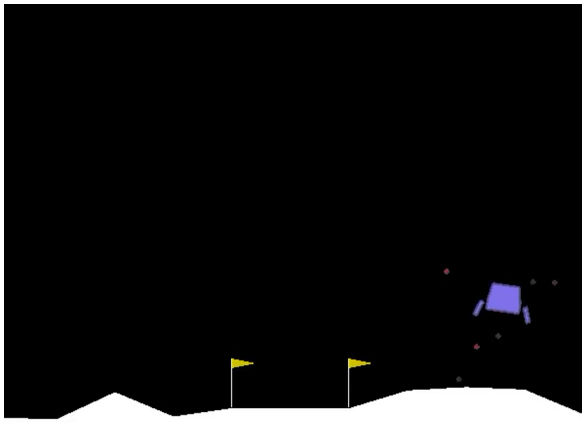

Figure 6. A visual representation of the LunarLanderContinuous-v2 environment. The agent uses information on the position and velocity of the lander (seen on the right of the image) and fires thrusters with the goal of landing safely between the flags.

The state space for this problem is eight-dimensional. It consists of horizontal and vertical position, horizontal and vertical velocity, angular position and velocity, and two binary indicators for whether each of the right and left legs are in contact with the ground. The allowed interval for all the components of the state space except the legs is $(-\infty, \infty)$.

The action space is two-dimensional. The first component is a main engine thrusting from the bottom of the spacecraft in the interval $[-1, 1]$. The subinterval $[-1, 0)$ represents the engine being off and the subinterval $[0, 1]$ represents the percent of the engine power being used. The second component represents side thrusters applying a rotational force that has values between $[-1, 1]$. The subinterval $[-1, 0.5]$ represents firing the left thruster, the subinterval $(-0.5, 0.5)$ represents no thrust, and the subinterval $[0.5, 1]$ represents firing the right thruster.

5.3. CarRacing-v0

The CarRacing-v0 or Race Car environment randomly generates a track for a car to race on from a top-down perspective. Observe Figure 7.

The state space is a pixel representation, and so it is much larger than the previous environments. An RGB value gives three components for each pixel in a 96×96 resolution, resulting in $96 \times 96 \times 3 = 27{,}648$ dimensions.

The action space is three-dimensional and consists of the steering in $[-1, 1]$, gas in $[0, 1]$, and brake in $[0, 1]$. For steering, a negative value turns the car to the left, a positive value turns it to the right, and a value of 0 keeps it moving straight ahead. For gas and brake, the value is the proportion of maximum acceleration or braking applied.

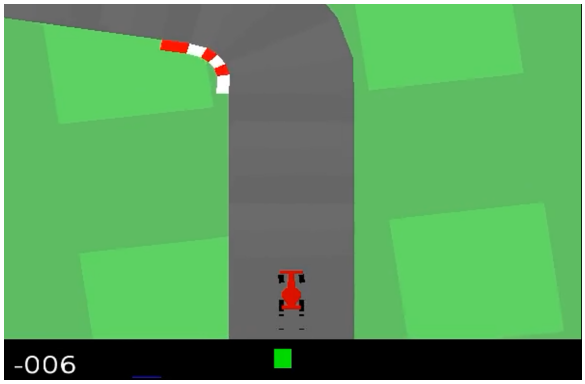

Figure 7. A visual representation of the CarRacing-v0 environment. The agent controls the throttle, brake, and steering of the car (seen in the bottom-center of the image) with the goal of encountering as much of the track as possible quickly.

5.4. Results

For each environment, each of the three action selection methods (PR, LDAS, OU) was tested for 100 episodes for 100 steps each. After each timestep beginning with the fifth (as discrepancy is less meaningful for such short sequences), the discrepancy of the sequence was calculated and recorded. This entire process was repeated 100 times. Finally, the average discrepancy over the 100 runs was calculated at each timestep. The results are presented in Figure 8.

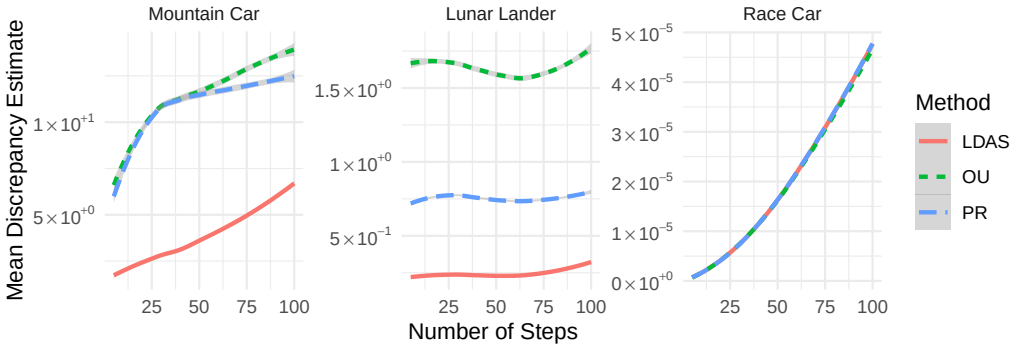

Figure 8. Average discrepancy versus number of steps for all environments. Lower is better.

The results confirm that in the two lower-dimensional problems, LDAS exploration results in a more even coverage of the state-action space. However, in the Race Car environment, the discrepancy estimates are almost identical for all three methods, with OU at a slight advantage. As the 27,648-dimensional state space is so large, state–action pairs have a low probability of being nearby each other until many steps have occurred. The purposefully correlated actions of OU can be advantageous for this problem when they result in the car consistently accelerating forward, whereas PR and LDAS may sporadically alternate between low and high acceleration and braking. If the experiment were run for a larger number of steps, it is possible that a difference in the discrepancy estimates may appear. This and Figure 4 suggest, agreeing with intuition, that advantages of LDAS will weaken in high-dimensional spaces where points will naturally be distant with high probability.

Lower discrepancy in exploration comes at a computational cost. Figure 9 displays the average algorithm run time per step for the three methods. As expected, LDAS incurs a higher computational cost for all environments, and increasingly so for the more complex environments.

Remember that these experiments only consider exploration and data collection, and do not yet take into account the effect on learning a policy from the data via a neural network or similar model. Time used in data generation is typically dwarfed by time spent

learning a policy. Therefore, while LDAS will take longer to generate the same amount of data as PR or OU, the data should be of higher quality in the sense that they represent more of the state-action space, and the learning phase may require less data or converge to an optimal policy more quickly. We conjecture that for certain environments, LDAS exploration will be a net benefit, and increased computation time in the exploration phase will be offset by a greater decrease in computation time in the learning phase. Investigating this conjecture will be the target of future research projects.

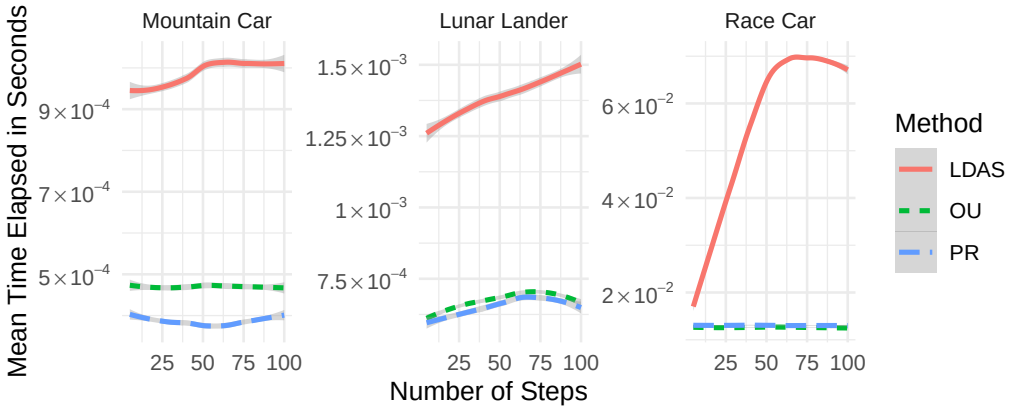

Figure 9. Average time per step (in seconds) versus number of steps for all environments.

6. Conclusions

This paper has presented LDAS, an algorithm for exploring MDPs with continuous states and actions. It combines the concepts of low-discrepancy sequence generation with action selection in RL. We performed experiments on three benchmark RL environments to demonstrate improved uniformity of state-action sampling. For two of the three environments, Mountain Car and Lunar Lander, LDAS had a significantly lower discrepancy than PR and OU. For the Race Car environment, all of the methods have roughly the same discrepancy, with OU at a slight advantage. The reason lies with the number of dimensions for the environments. In the lower-dimensional problems, there is a high probability that pseudo-random actions will result in clusters, so there is much room for improvement by LDAS. On the other hand, for the high-dimensional problem, state-actions will naturally be more sparse and spread out, so the benefit of LDAS is reduced. This suggests that LDAS is a more attractive exploration strategy for low-dimensional problems than high-dimensional ones.

One limitation of LDAS is increased computation time. The average time per step increased roughly five-fold for low-dimensional problems, and roughly twenty-fold for high-dimensional problems. However, for the low-dimensional problems, the improved uniformity of exploration may be worth it. Future experiments will train an agent on data collected by LDAS, PR, and OU exploration methods to investigate the effect on quality of the resulting policy and computation time in the learning phase.

Several avenues of research remain open for the future. The foremost is discovering whether agents trained on low-discrepancy data learn to complete their task better, faster, or with fewer data requirements. This will be the ultimate test as to whether LDAS should become a standard exploration method. If so, the computational efficiency merits additional investigation, and whether LDAS can be improved, so that the increased computational time is less of a drawback.

At its core, LDAS is a gradient method for optimizing distance from the nearest point, but it is basic in the sense of deterministically and exhaustively calculating all distances, and using constant, prespecified parameters for learning rate and learning rate decay. The last decade has seen advances in adaptive gradient methods for stochastic objective functions, such as AdaGrad (adaptive gradients) [26] and Adam (adaptive moments) [27].

A more advanced version of LDAS could calculate distances from a sample of previous points and use an adaptive gradient method to quickly approximate the action optimal for exploration. With this change, LDAS could potentially be significantly faster at the cost of a small performance decrease.

This paper took the perspective of completely bifurcated exploration and learning phases, but many algorithms integrate the tasks, attempting to balance exploration and exploitation simultaneously. A worthy question is how to modify LDAS to assign higher selection probabilities to actions with a higher estimated value. The field of low-discrepancy exploration is still ripe for research.

Author Contributions: Conceptualization, S.W.C.; methodology, S.W.C. and J.O.L.; investigation, S.W.C. and J.O.L.; writing—original draft preparation, S.W.C., J.O.L., and Z.U.; writing—review and editing, S.W.C., J.O.L., and Z.U.; visualization, S.W.C. and J.O.L. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The Python and R code used to generate data and produce visualizations is publicly available at https://github.com/swcarden/LDAS (accessed on 10 May 2022).

Acknowledgments: The authors wish to acknowledge Georgia Southern University's Computational Research Technical Support office and the helpful suggestions of anonymous reviewers.

Conflicts of Interest: The authors declare no conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

MDP	Markov Decision Process
RL	Reinforcement Learning
LDAS	Low-Discrepancy Action Selection
PR	Pseudo Random
OU	Ornstein–Uhlenbeck
RGB	Red Green Blue

References

1. Pathical, S.; Serpen, G. Comparison of subsampling techniques for random subspace ensembles. In Proceedings of the 2010 International Conference on Machine Learning and Cybernetics, Qingdao, China, 11–14 July 2010; Volume 1, pp. 380–385. https://doi.org/10.1109/ICMLC.2010.5581032.
2. Kumar, S.; Mohri, M.; Talwalkar, A. Sampling Methods for the Nyström Method. *J. Mach. Learn. Res.* **2012**, *13*, 981–1006.
3. Schneider, M. Probability Inequalities for Kernel Embeddings in Sampling without Replacement. In Proceedings of the 19th International Conference on Artificial Intelligence and Statistics, Cadiz, Spain, 9–11 May 2016; Gretton, A., Robert, C.C., Eds.; PMLR: Cadiz, Spain, 2016; Volume 51, pp. 66–74.
4. Cannon, A.; Ettinger, J.M.; Hush, D.; Scovel, C. Machine Learning with Data Dependent Hypothesis Classes. *J. Mach. Learn. Res.* **2002**, *2*, 335–358.
5. Feng, X.; Kumar, A.; Recht, B.; Ré, C. Towards a Unified Architecture for In-RDBMS Analytics. In Proceedings of the SIGMOD '12, 2012 ACM SIGMOD International Conference on Management of Data, Scottsdale, AZ, USA, 20–24 May 2012; Association for Computing Machinery: New York, NY, USA, 2012; pp. 325–336. https://doi.org/10.1145/2213836.2213874.
6. Gürbüzbalaban, M.; Ozdaglar, A.; Parrilo, P. Why Random Reshuffling Beats Stochastic Gradient Descent. *arXiv* **2015**, arXiv:1510.08560.
7. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
8. Kearns, M.; Singh, S. Near-Optimal Reinforcement Learning in Polynomial Time. *Mach. Learn.* **2002**, *49*, 209–232.
9. Carden, S.W.; Walker, S.D. Exploration Using Without-Replacement Sampling of Actions Is Sometimes Inferior. *Mach. Learn. Knowl. Extr.* **2019**, *1*, 698–714. https://doi.org/10.3390/make1020041.

10. Nichols, B.D. A Comparison of Action Selection Methods for Implicit Policy Method Reinforcement Learning in Continuous Action-Space. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016. https://doi.org/10.1109/ijcnn.2016.7727688.
11. Nocedal, J.; Wright., S.J. Springer series in operations research and financial engineering. In *Numerical Optimization*; Springer: New York, NY, USA, 1999. https://doi.org/10.1007/b98874.
12. Nelder, J.A.; Mead, R. A Simplex Method for Function Minimization. *Comput. J.* **1965**, *7*, 308–313. https://doi.org/10.1093/comjnl/7.4.308.
13. Nichols, B.D.; Dracopoulos, D.C. Application of Newton's Method to action selection in continuous state-and action-space reinforcement learning. In Proceedings of the ESANN 2014 Proceedings—22nd European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Bruges, Belgium, 23–25 April 2014; pp. 141–146.
14. Lazaric, A.; Restelli, M.; Bonarini, A. Reinforcement Learning in Continuous Action Spaces through Sequential Monte Carlo Methods. In *Advances in Neural Information Processing Systems*; Platt, J., Koller, D., Singer, Y., Roweis, S., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2007; Volume 20.
15. Xu, X.; Liu, C.; Hu, D. Continuous-action reinforcement learning with fast policy search and adaptive basis function selection. *Soft Comput.* **2010**, *15*, 1055–1070. https://doi.org/10.1007/s00500-010-0581-3.
16. Hanna, J.P.; Niekum, S.; Stone, P. Importance sampling in reinforcement learning with an estimated behavior policy. *Mach. Learn.* **2021**, *110*, 1267–1317. https://doi.org/10.1007/s10994-020-05938-9.
17. van der Corput, J.G. Verteilungsfunktionen. I. *Proc. Akad. Wet. Amst.* **1935**, *38*, 813–821.
18. Halton, J.H. On the efficiency of certain quasi-random sequences of points in evaluating multi-dimensional integrals. *Numer. Math.* **1960**, *2*, 84–90. https://doi.org/10.1007/BF01386213.
19. Kuipers, L.; Niederreiter, H. *Uniform Distribution of Sequences*; Dover Books on Mathematics, Dover Publications: Mineola, NY, USA, 2012.
20. Uhlenbeck, G.E.; Ornstein, L.S. On the Theory of the Brownian Motion. *Phys. Rev.* **1930**, *36*, 823–841. https://doi.org/10.1103/PhysRev.36.823.
21. Van Rossum, G.; Drake, F.L., Jr. *Python Reference Manual*; Centrum voor Wiskunde en Informatica: Amsterdam, The Netherlands, 1995.
22. Brockman, G.; Cheung, V.; Pettersson, L.; Schneider, J.; Schulman, J.; Tang, J.; Zaremba, W. OpenAI Gym. *arXiv* **2016**, arXiv:1606.01540.
23. The Garage Contributors. Garage: A Toolkit for Reproducible Reinforcement Learning Research. 2019. Available online: https://github.com/rlworkgroup/garage (accessed on 10 May 2022).
24. R Core Team. *R: A Language and Environment for Statistical Computing*; R Foundation for Statistical Computing: Vienna, Austria, 2022.
25. Wickham, H.; Averick, M.; Bryan, J.; Chang, W.; McGowan, L.D.; François, R.; Grolemund, G.; Hayes, A.; Henry, L.; Hester, J.; et al. Welcome to the tidyverse. *J. Open Source Softw.* **2019**, *4*, 1686. https://doi.org/10.21105/joss.01686.
26. Duchi, J.; Hazan, E.; Singer, Y. Adaptive Subgradient Methods for Online Learning and Stochastic Optimization. *J. Mach. Learn. Res.* **2011**, *12*, 2121–2159.
27. Kingma, D.P.; Ba, J. Adam: A Method for Stochastic Optimization. *arXiv* **2014**, arXiv:1412.6980.