Estimating the Variance of Return Sequences for Exploration

Zerong Xi
Department of Computer Science
University of Central Florida
Orlando, FL, USA
xizerong@gmail.com

Gita Sukthankar
Department of Computer Science
University of Central Florida
Orlando, FL, USA
gitars@eecs.ucf.edu

***Abstract*—This paper introduces a method for estimating an upper bound for an exploration policy using either the weighted variance of return sequences or the weighted temporal difference (TD) error. We demonstrate that the variance of the return sequence for a specific state-action pair is an important information source that can be leveraged to guide exploration in reinforcement learning. The intuition is that fluctuation in the return sequence indicates greater uncertainty in the near future returns. This divergence occurs because of the cyclic nature of value-based reinforcement learning; improved estimates of the value function result in policy changes which in turn modify the value function. Although both variance and TD errors capture different aspects of this uncertainty, our analysis shows that both can be valuable to guide exploration. We propose a two-stream network architecture to estimate weighted variance/TD errors within DQN agents for our exploration method and show that it outperforms the baseline on a wide range of Atari games.**

***Index Terms*—exploration, reinforcement learning, DQN**

I. Introduction

Having a good exploration policy is an essential component of achieving sample efficient reinforcement learning. Most RL applications use two heuristics, visitation counts and time, to guide exploration. *Count-based exploration* [1] assumes that it is worth allocating the exploration budget towards previously unexplored actions by awarding exploration bonuses based on action counts. *Time-based exploration* [2] is usually implemented using a Boltzmann distribution that reduces exploration during later stages of the learning process. This paper presents an analysis of the benefits and drawbacks of weighted sequence variance for guiding exploration; we contrast the performance of weighted variance with the more familiar weighted temporal difference (TD) error.

Our intuition about the merits of weighted variance as a heuristic to guide exploration is as follows. Imagine that the returns are being summed in a potentially infinite series. Weighted variance can be computed online in order to estimate the convergence speed of the series for a specific state-action pair. We estimate the upper bound using uncertainty, modeled as the weighted standard deviation, as an exploration bonus to guide action selection.

Fluctuation in the return sequence may foretell greater uncertainty in the near future returns that should be rectified through allocation of the exploration budget. Value-based RL algorithms are particularly susceptible to divergence, since improvements in the value function result in rapid policy changes which in turn affect the value estimation. Unlike event counts, weighted variance is more sensitive to the dynamics of the return sequence; if multiple visitations yield consistent reward, weighted variance will quickly prioritize a different state-action sequence even if the total event counts are smaller. We present an empirical analysis showing how weighted variance reacts to the dynamics of raw, smoothed, and residual return sequences.

Computing weighted variance within a deep reinforcement learning framework is a challenging problem, due to the instability of deep neural networks. Simply computing the variance directly from the output of DQN risks overestimating the error. The second contribution of the paper is introducing a two-stream network architecture to estimate either weighted variance or TD errors within DQN agents. Our new architecture (Variance Deep Q Networks) uses a separate σ stream to estimate a weighted standard deviation of the outputs from the original stream.

II. Related Work

Several groups have proposed strategies for balancing exploration/exploitation in deep reinforcement learning including 1) extensions on count-based methods [3]–[5]; 2) noise injection techniques [6], [7]; 3) improving uncertainty estimation [8], [9]; 4) driving exploration with intrinsic motivation [10], [11] and 5) entropy-guided architectures [12], [13]. Our proposed weighted-variance guided exploration technique is a compatible addition to some of these other techniques (see Section VI for further details).

Moving from tabular to deep reinforcement learning makes the problem of estimating quantities such as counts and variance more challenging. [4] showed how count-based techniques could be generalized to neural networks by learning a density model on state sequences; pseudocounts for states are then derived from the density model's recoding probability. In contrast our model learns the weighted variance over the return sequence rather than the state sequence.

Osband et al. [8] argue for the necessity of deep exploration and design two environments, a MDP chain and Deep Sea, to illustrate it. The immediate rewards direct the agent away

from reaching the goal, and an uninformed search can expect to take $\mathcal{O}(|\mathcal{A}|^N)$ time to explore the goal, where $\mathcal{A}$ is the set of actions. However, the goals in those environments are designed to be achieved with consistent action preferences since action outcomes are deterministic. Therefore, the expected search time reduces to $\mathcal{O}(|\mathcal{A}|)$ among agents who have diverse but consistent preferences on actions. Bootstrapped DQN [8] and Randomized Value Function [14] achieve this effect by sampling a value function or an agent and applying it through a whole training episode. Though many other exploration methods, including ours, perform poorly in these artificial scenarios, they can be adjusted to achieve the goal by adding a set of randomly sampled but fixed noise $\epsilon \in R^{|\mathcal{A}|}$ on top of the value function over a training episode. This setting gives the agents an action preference which remains consistent in a single training episode and diverse across episodes.

The use of randomness or noise to drive exploration is a common theme across many approaches. NoisyNets [6] directly injects noise into the weights of the neural networks; the parameters of the noise are learned in combination with the network weights. Like NoisyNet, our uncertainty is learned directly by the network, reducing the need for extra hyper-parameters. Aleatoric uncertainty poses a common challenge for all exploration methods built on bonuses of uncertainty or curiosity. However these methods remain useful when the aleatoric one doesn't dominate the overall uncertainty.

III. Background

Our aim is to learn an action policy for a stochastic world modeled by a Markov Decision Process by balancing the exploration of new actions and the exploitation of actions known to have a high reward. This is done by learning a value function ($Q(s,a)$) using the discounted return information ($G(s,a)$) and learning rate (α):

$$Q(s,a) = Q(s,a) + \alpha \cdot (G(s,a) - Q(s,a)) \tag{1}$$

Actions are selected using the learned value function. This paper illustrates how our weighted variance exploration approach can be integrated into agents using deep Q-learning.

Deep Q Networks [15] utilize deep neural networks as approximators for action-value functions in Q learning algorithms. The updating procedure of the function is formulated as an optimization problem on a loss function:

$$\mathbb{E}_{(s,a,r,s')\sim D}\left[\left(r + \gamma \cdot \max_{b\in\mathcal{A}} Q(s', b; \zeta^-) - Q(s,a;\zeta)\right)^2\right] \tag{2}$$

where ζ are the parameters of the network, $\mathcal{A}$ is a set of valid actions and D is a distribution over a replay buffer of previously observed transitions. A target network with parameters ζ^- is regularly synchronized with ζ and used to estimate the action values of the successor states; the use of a target network promotes estimation stability. Since the original introduction of DQN, several improvements to the updating procedure and network architecture have been proposed.

Double DQN [16] updates the network according to a different rule in which the action for the successor state is selected based on the target network rather than the updating network. This change alleviates the overestimation problem by disentangling the estimation and selection of action during optimization steps. The loss function for Double DQN is:

$$\mathbb{E}_{(s,a,r,s')\sim D}\left[\left(r + \gamma \cdot Q(s', \text{argmax}_{b\in\mathcal{A}} Q(s', b; \zeta); \zeta^-) - Q(s,a;\zeta)\right)^2\right]. \tag{3}$$

IV. Measuring Variance for Exploration

During Monte Carlo policy evaluation, the value function $Q(s,a)$ for a particular state-action pair is updated using a sequence of returns $\mathcal{G}_n(s,a) = G_1(s,a), G_2(s,a), ..., G_n(s,a)$. This series can start diverging due to the cyclic nature of value-based approaches; the changing value function results in policy improvements which in turn modify the value function. We believe that the agent should leverage information from these variations to quantify uncertainty in order to explore non-optimal, but still promising, actions. Specifically, agents can follow a greedy exploration policy based on an upper bound:

$$\pi(s) = \text{argmax}_{a\in\mathcal{A}} Q(s,a) + \sigma(s,a) \cdot c, \tag{4}$$

where σ is a measurement of uncertainty and c is a fixed hyper-parameter which adjusts the extent of exploration.

To measure the uncertainty of returns for a specific state-action pair, we propose 1) a weighted variance estimation method for general RL, 2) a neural network architecture, and 3) novel optimizing strategy which explicitly estimates either weighted variance or weighted TD error in the DRL configuration.

A. Reinforcement Learning with Variance Estimation

Although the vanilla form of sequence variance doesn't reflect the higher importance of the recent returns, we define the uncertainty as an exponentially decaying weighted standard deviation

$$\sigma_n(s,a) = \sqrt{\frac{\sum_{i=1}^{n}(1-\alpha)^{n-i}(G_i(s,a) - Q_n(s,a))^2}{\sum_{i=1}^{n}(1-\alpha)^{n-i}}}, \tag{5}$$

where $Q_n(s,a)$ is the value function which is updated using $\mathcal{G}_n(s,a)$ and α (the update step size) in Eq. 1.

The update formula for σ is as follows

$$\sigma_{n+1}(s,a) = \sqrt{\begin{array}{l}(1-\alpha)\cdot\left[\sigma_n^2(s,a) + (Q_{n+1}(s,a) - Q_n(s,a))^2\right] \\ +\alpha\cdot(G_{n+1}(s,a) - Q_{n+1}(s,a))^2\end{array}}. \tag{6}$$

The first term inside the square root represents the adjusted estimation of variance on $\mathcal{G}_n(s,a)$ with the updated $Q_{n+1}(s,a)$, and the second term is the estimation from the incoming $G_{n+1}(s,a)$. Instead of estimating the variance from

TD-errors [17], we deduce it from the mathematical definition of weighted variance. That results in an additional adjustment on the estimation of the previous variance with the updated Q value.

When updates are performed using the above formula, $\sigma(s,a)$ is biased during the early stage, due to the undecidable prior $\sigma_0(s,a)$ as well as the bias incipient to the usage of a small n. We propose two strategies for initializing the σ function: 1) warming up with an ϵ-greedy method with ϵ decayed to 0 to ensure a gradual transition to our exploration policy, which effectively starts with a larger n; 2) initializing $\sigma_0(s,a)$ as a large positive value to encourage uniform exploration during early stages, which is theoretically sound since the variance of the value of a state-action pair is infinitely large if it has never been visited.

B. Variance Deep Q Networks (V-DQN)

Our new algorithm, V-DQN, incorporates weighted variance into the exploration process of training DQNs. Due to the known instability of deep neural networks during the training process, it is risky to calculate the weighted variance from composing multiple estimations (e.g., the state-action values before and after the optimization step).

Instead of computing the target variance as a byproduct, we propose a two-stream neural network architecture along with an novel loss function to allow end-to-end training while estimating the weighted standard deviation.

It simplifies the optimization for variance by ignoring the adjustment on the previous variance, which is closer to the form in [17]. Since the deep neural networks with gradient descent cannot strictly follow the above updating formula, we believe it's an acceptable compromise. Our empirical results demonstrate the effectiveness of the simplification.

Variance DQN uses neural networks with a separate σ-stream to estimate a weighted standard deviation of the outputs from the original stream on moving targets, which is common in the context of deep reinforcement learning where the value function improves as the policy evolves. In practice, the σ-stream shares lower layers, e.g. convolutional layers, with the original stream to reduce computational demands.

The loss function for Variance DQN is a sum of mean square error losses on the original stream, which is identical to formula 2 (for DQN) or formula 3 (for Double DQN), and the square of the σ-stream:

$$L_{\text{V-DQN}} = \mathbb{E}_{(s,a,r,s')\sim D}\left[(G - Q(s,a;\zeta))^2 \left((G - Q(s,a;\zeta))^2 - \sigma^2(s,a;\zeta)\right)^2\right] \tag{7}$$

$$s.t.\ G = \begin{cases} r + \gamma \cdot \max_{b\in\mathcal{A}} Q(s',b;\zeta^-) & \text{for DQN} \\ r + \gamma \cdot Q(s', \text{argmax}_{b\in\mathcal{A}} Q(s',b;\zeta);\zeta^-) & \text{for DDQN} \end{cases} \tag{8}$$

It is worth noting that the Q function used in the second part of the loss function on the σ-stream doesn't contribute to gradients directly. Therefore, the optimization steps are in effect unchanged for the original stream on the Q value, except for the shared lower layers. While the sign of the σ-stream's output is eliminated in the loss function, we need to do the same during the exploration process. The modified exploration policy is

$$\pi(s) = \text{argmax}_{a\in\mathcal{A}} Q(s,a) + |\sigma(s,a)| \cdot c \tag{9}$$

The full procedure is shown in Algorithm 1. We also propose a variant of our method (TD-DQN) which updates σ-stream with absolute temporal difference error. The loss function for TD-DQN is

$$L_{\text{TD-DQN}} = \mathbb{E}_{(s,a,r,s')\sim D}\left[(G - Q(s,a;\zeta))^2 + (|G - Q(s,a;\zeta)| - \sigma(s,a;\zeta))^2\right] \tag{10}$$

where G is the same as Equation 8.

Both networks measure the uncertainty of the Q value based on the return history in order to construct an upper bound for exploration policy. The difference between the approaches can be interpreted based on their implicit usage of different distance metrics: Euclidean (Variance DQN) vs.

Algorithm 1 Variance DQN (V-DQN)

Input: exploration parameter c; minibatch k; target network update step τ;
Input: initial network parameters ζ; initial target network parameter ζ^-;
Input: Boolean DOUBLE

Initialize replay memory $\mathcal{H} = \emptyset$
Observe s_0
for $t \in \{1, ..., T\}$ **do**
  Select an action $a \leftarrow \text{argmax}_{b\in\mathcal{A}} Q(s,b;\zeta) + |\sigma(s,b;\zeta)| \cdot c$
  Sample next state $s \sim P(\cdot|s,a)$ and receive reward $r \leftarrow R(s,a)$
  Store transition (s_{t-1}, a, r, s_t) in $\mathcal{H}$
  for $j \in \{1, ..., k\}$ **do**
    Sample a transition $(s_j, a_j, r_j, s'_j) \sim D(\mathcal{H})$ ▷ D can be uniform or prioritised replay
    if s'_j is a terminal state **then**
      $G \leftarrow r_j$
    else if DOUBLE **then**
      $b^*(s'_j) = \text{argmax}_{b\in\mathcal{A}} Q(s'_j, b; \zeta)$
      $G \leftarrow r_j + \gamma Q(s'_j, b^*(s'_j); \zeta^-)$
    else
      $G \leftarrow r_j + \gamma \text{max}_{b\in\mathcal{A}} Q(s'_j, b; \zeta^-)$
    end if
    $\hat{\sigma} \leftarrow G - Q(s_j, a; \zeta)$
    Do a gradient step with loss $(G - Q(s_j,a;\zeta))^2 + (\hat{\sigma}^2 - \sigma^2(s_j,a;\zeta))^2$
  end for
  if $t \equiv 0 \pmod{\tau}$ **then**
    Update the target network $\zeta^- \leftarrow \zeta$
  end if
end for

Manhattan (TD-DQN). Generally, V-DQN is more sensitive to fluctuations in the return sequence, investing a greater amount of the exploration budget to damp variations.

There may be applications in which it is is valuable to estimate still higher order statistics, such as the skew or kurtosis of the return sequence. However, this sensitivity can also sabotage exploration by directing resources away from promising areas of the state space that are slowly trending towards convergence; overemphasizing the elimination of small variations could ultimately result in longer training times. While the c hyper-parameter in our exploration policy adjusts the trade-off between exploration and exploitation, the choice of the distance metrics used to measure sequence variation determines the distribution of exploration time.

V. Results

To illustrate how weighted variance improves exploration, this paper first presents results on its usage in tabular Q-learning for the Cartpole inverted pendulum problem. Then we report the performance of our proposed techniques (V-DQN and TD-DQN) on the Atari game benchmark. The results show that our two stream architecture for guiding exploration with weighted variance or weighted temporal difference outperforms the standard DDQN benchmark.

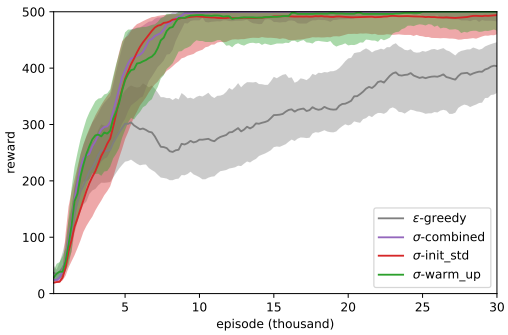
A. Cartpole

To demonstrate the effectiveness of our Variance Estimation (VE) method, we compare it with ϵ-greedy on Cartpole balancing problem. For this experiment, we use the classic Q-learning algorithm [18] with a tabular look-up Q-table.

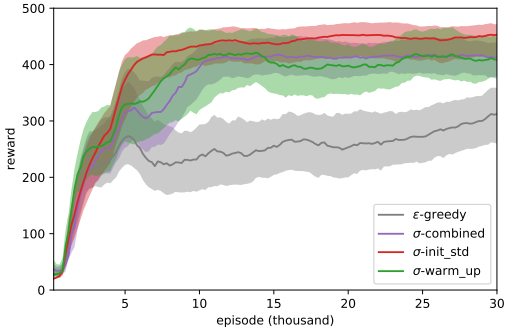
The Cartpole environment has a 4-dimensional continuous state space $\mathcal{S} = \mathbb{R}^4$ and a discrete action space $\mathcal{A} = \{$*Push Left, Push Right, Do Nothing*$\}$. It provides a reward of 1 in every time step. The episode terminates if any of the following conditions is met: 1) the episode length is greater than 500, 2) the pole falls below a threshold angle, 3) the cart drifts from the center beyond a threshold distance. With this setting, the maximum accumulated reward any policy can achieve is 500. To apply the tabular Q-learning configuration, the continuous state space is discretized into 18,432 discrete states by dividing $\{$*Cart Position, Cart Velocity, Pole Angle, Pole Velocity*$\}$ into $\{12, 8, 16, 12\}$ intervals.

With grid search, ϵ-greedy achieves the best performance when the discounting factor $\gamma = 1.0$ and the exploration rate ϵ decays from 1.0 to 0.01 in 5000 episodes.

For Variance Estimation, we experimented on both initialization methods as well as a combination of them. A similar configuration is applied on warming up with the ϵ-greedy method in which ϵ decays from 1.0 to 0.0 during 5000 episodes. The initial standard deviation is set to 5000 for initializing the σ_0 method to ensure sufficient early visits on states. The combination method warms up with ϵ-greedy while retaining the large initial standard deviation; it uses the same hyper-parameters. The value of c is set to 1.5 for initializing the σ_0 method and 0.5 for the other two.



(a) Episode reward ($\epsilon_{\text{eval}} = 0$)



(b) Episode reward ($\epsilon_{\text{eval}} = 0.05$)

Fig. 1: Average episode rewards in the Cartpole balancing problem. The curves and the shadowed areas represent the means and the quartiles over 9 independent runs. The models are evaluated for 10 evaluation episodes every 200 training episodes.

To reduce the possibility of over-fitting, we evaluate the models with an additional environment in which the agent has a probability $\epsilon_{\text{eval}} = 0.05$ of acting randomly. All of our methods outperform ϵ-greedy consistently for both evaluation settings. When the training time is prolonged, the baseline method generally achieves similar scores to our methods, but requires approximately 10 times the training episodes.

B. Atari Games

We evaluate our DQN-based algorithms on 55 Atari games from the Arcade Learning Environment (ALE) [19], simulated via the OpenAI Gym platform [20]. Defender and Surround are excluded because they are unavailable in this platform. The baseline method (denoted as DDQN) is DQN [15] with all the standard improvements including Double DQN [16], Dueling DQN [21] and Prioritized Replay [22].

Our network architecture has a similar structure to Dueling DQN [21], but with an additional σ-stream among fully connected layers. The 3 convolutional layers have 32 8×8 filters with stride 4, 64 4×4 filters with stride 2, 64 3×3 filters with stride 1 respectively. Then the network splits into three streams of fully connected layers, which are value, advantage and σ streams. Each of the streams has a hidden fully connected layer with 512 units. For the output layers, the

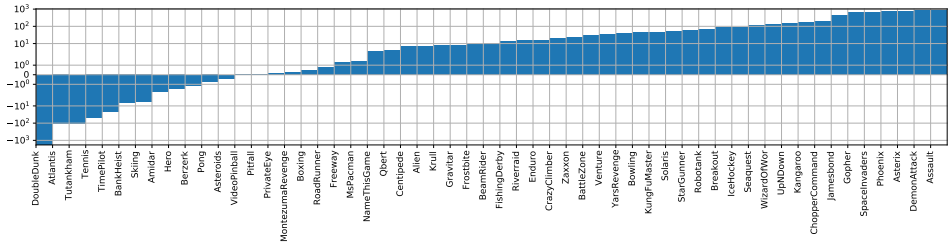


Fig. 2: Improvement in normalized scores of V-DQN over DDQN in 200M frames

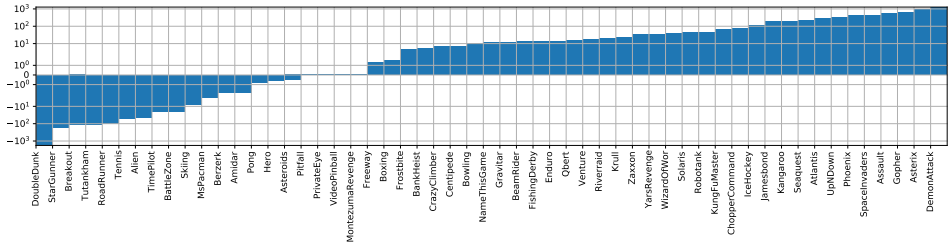
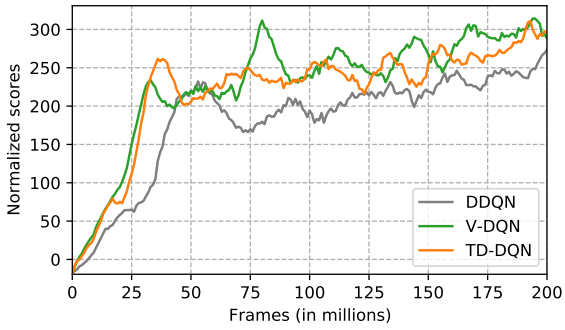
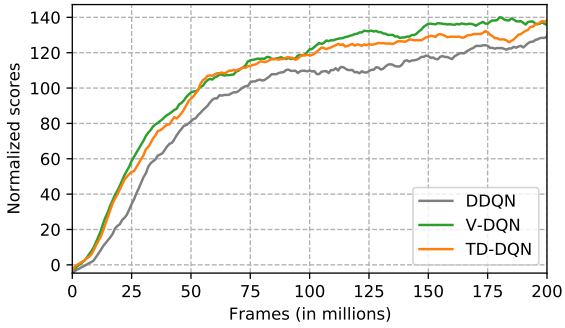


Fig. 3: Improvement in normalized scores of TD-DQN over DDQN in 200M frames



(a) Mean normalized scores of all Atari games over 200M frames.



(b) Median normalized scores of all Atari games over 200M frames.

Fig. 4: The mean and median of the normalized training curve over all 55 Atari games

value stream has a single output while both advantage and σ streams have the same number of outputs as the valid actions.

The random start no-op scheme in [15] is used here in both training and evaluation episodes. The agent repeats no-op actions for a randomly selected number of times between 1 to 30 in the beginning to provide diverse starting conditions to alleviate over-fitting. Evaluation takes place after freezing the network every 250K training steps (1M frames). The scores are the averages of episode rewards over 125K steps (500K frames) where episodes are truncated at 27K steps (108K frames or 30 minutes of simulated play).

We use the Adam optimizer [23] with a learning rate of 6.25×10^{-5} and a value of 1.5×10^{-4} for Adam's ϵ hyper-parameter over all experiments. The network is optimized on a mini-batch of 32 samples over prioritized replay buffer every 4 training steps. The target network is updated every 30K steps.

The exploration rate of DDQN decays from 1.0 to 0.01 in 250K steps (1M frames) and retains that value until the training ends. Our methods do not rely on ϵ-greedy so that is simply set to 0 for all the steps. Instead, the value of c impacts the actual exploration rates of our methods, which are defined here to be the proportion of actions different from the optimal ones based on the current Q value function. Empirically, the performance on Atari games does not vary significantly over a wide range of c values, which is an unusual finding. A possible explanation is that most of the actions in those games are not critical. To keep the exploration policy from drifting too far from the exploitation policy, we set c to be 0.1 for both V-DQN and TD-DQN over all experiments to keep the average exploration rates of the majority of the Atari games to reside roughly between 0.01 and 0.1.

A summary of the results over all 55 Atari games is reported in Table I. To compare the performance of agents over different games, the scores are normalized with human scores

$$\text{Score}_{\text{Normalized}} = 100 \times \frac{\text{Score}_{\text{Agent}} - \text{Score}_{\text{Random}}}{\text{Score}_{\text{Human}} - \text{Score}_{\text{Random}}} \qquad (11)$$

where both the random and the human scores were taken from

[22].

The results clearly show that our proposed methods for guiding exploration, V-DQN and TD-DQN, both improve on the standard DDQN benchmark. Although there are small differences in the ranking, both versions perform well in the same games, and underperform the benchmark in a small set of games. The mean and median statistics do not reveal significant differences between V-DQN and TD-DQN. Our intuition remains that V-DQN is likely to more sensitive to fluctuations and will allocate more exploration budget to damp them out.

	DDQN	V-DQN	TD-DQN
Median	151%	164%	164%
Mean	468%	547%	533%

TABLE I: Summary of normalized scores

VI. Conclusion and Future Work

This paper presents an analysis of the benefits and limitations of weighted variance for guiding exploration. Both weighted convergence and its close cousin, weighted temporal difference, can be used to quantify the rate of convergence of the return series for specific state-action pairs. The return dynamics of value-based reinforcement learning is particularly susceptible to diverging as value improvements beget policy ones. This paper introduces a new two-stream network architecture to estimate both weighted variance/TD errors; both our techniques (V-DQN and TD-DQN) outperform DDQN on the Atari game benchmark.

While our methods capture the divergence of return sequences, they suffer from the "cold start" problem. It is unlikely that they will perform well for either empty or short sequences. To address this, we propose two simple initialization methods for tabular configurations in this paper. This issue is somewhat alleviated by the generalization capacity inherent to function approximators like deep neural networks. However, larger state spaces where most of the states will never be visited still pose a problem.

Our method can further benefit from unification with other exploration methods. Count-based upper confidence bound (UCB) methods [4] balance visits among states during the early phases of exploration; this effect decays gradually as visits increase. This characteristic makes it a natural complement for our sequence-based methods. Noisy DQN [6] is another option that assigns greater randomness to less visited states. Our method focuses on promising actions whereas Noisy DQN chooses actions more randomly in those areas of the state space. We ran some experiments on a rudimentary design in which the linear layers of Q-value stream were replaced with noisy ones; our preliminary results (not reported) show an improvement by hybridizing the two architectures.

VII. Acknowledgments

This research was supported with funding from Lockheed Martin Corporation.

References

[1] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," *Machine Learning: ECML*, vol. 2006, pp. 282–293, 09 2006.

[2] L. P. Kaelbling, M. L. Littman, and A. W. Moore, "Reinforcement learning: A survey," *J. Artif. Intell. Res.*, vol. 4, pp. 237–285, 1996.

[3] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," in *Advances in Neural Information Processing Systems*, 2016.

[4] G. Ostrovski, M. G. Bellemare, A. van den Oord, and R. Munos, "Count-based exploration with neural density models," in *Proceedings of the International Conference on Machine Learning*, 2017.

[5] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. Xi Chen, Y. Duan, J. Schulman, F. DeTurck, and P. Abbeel, "#exploration: A study of count-based exploration for deep reinforcement learning," in *Advances in Neural Information Processing Systems*, 2017.

[6] M. Fortunato, M. G. Azar, B. Piot, J. Menick, M. Hessel, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin, C. Blundell, and S. Legg, "Noisy networks for exploration," in *International Conference on Learning Representations*, 2018.

[7] M. Plappert, R. Houthooft, P. Dhariwal, S. Sidor, R. Y. Chen, X. Chen, T. Asfour, P. Abbeel, and M. Andrychowicz, "Parameter space noise for exploration," in *International Conference on Learning Representations*, 2018.

[8] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via bootstrapped DQN," in *Advances in Neural Information Processing Systems*, 2016.

[9] N. Nikolov, J. Kirschner, F. Berkenkamp, and A. Krause, "Information-directed exploration for deep reinforcement learning," in *International Conference on Learning Representations*, 2019.

[10] N. Chentanez, A. G. Barto, and S. P. Singh, "Intrinsically motivated reinforcement learning," in *Advances in Neural Information Processing Systems*, 2005.

[11] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell, "Curiosity-driven exploration by self-supervised prediction," in *Proceedings of the International Conference on Machine Learning*, 2017.

[12] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine, "Reinforcement learning with deep energy-based policies," in *Proceedings of the International Conference on Machine Learning*, 2017.

[13] E. Hazan, S. Kakade, K. Singh, and A. V. Soest, "Provably efficient maximum entropy exploration," in *International Conference on Learning Representations*, 2019.

[14] I. Osband, B. V. Roy, D. J. Russo, and Z. Wen, "Deep exploration via randomized value functions," *Journal of Machine Learning Research*, 2019.

[15] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, Feb 2015.

[16] H. v. Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016.

[17] C. Sherstan, D. R. Ashley, B. Bennett, K. Young, A. White, M. White, and R. S. Sutton, "Comparing direct and indirect temporal-difference methods for estimating the variance of the return," in *Proceeding of Uncertainty in Artificial Intelligence*, 2018.

[18] C. J. C. H. Watkins and P. Dayan, "Q-learning," in *Machine Learning*, 1992.

[19] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling, "The arcade learning environment: An evaluation platform for general agents," *Journal of Artificial Intelligence Research*, May 2013.

[20] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[21] Z. Wang, T. Schaul, M. Hessel, H. Van Hasselt, M. Lanctot, and N. De Frcitas, "Dueling Network Architectures for Deep Reinforcement Learning," *the International Conference on Machine Learning*, 2016.

[22] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized Experience Replay," *Proceedings of the International Conference on Learning Representations*, 2015.

[23] D. P. Kingma and J. L. Ba, "Adam: A method for stochastic optimization," in *International Conference on Learning Representations*, 2015.