Return Dispersion as an Estimator of Learning Potential for Prioritized Level Replay

Anonymous Author(s)
Affiliation
Address
email

Abstract

Prioritized Level Replay (PLR) has been shown to induce adaptive curricula that improve the sample-efficiency and generalization of reinforcement learning policies in environments featuring multiple tasks or levels. PLR selectively samples training levels weighed by a function of recent temporal-difference errors experienced on each level. We explore the dispersion of returns as an alternative prioritization criterion to address certain issues with TD error scores.

1 Introduction

Recently, there has been a growing interest in testing the generalization and sample efficiency of RL algorithms [1–5]. This led to the development of new benchmarks such as Procgen [6], which consists of 16 procedurally-generated game environments, allowing for distinct train and test levels.

One method which has been shown to be effective is Prioritized Level Replay (PLR) [7]. PLR performs active sampling of training levels in environments featuring multiple levels by selectively sampling the next training level in proportion to its learning potential, as estimated by a function of recent TD-errors experienced on each level. For Proximal Policy Optimization (PPO) [8], the best results were achieved when prioritizing based on the average L1 value loss, or equivalently, the magnitude of the GAE [9]. PLR improves both sample-efficiency and generalization by actively sampling for the largest opportunities for reducing TD errors or the value loss, and thus the epistemic uncertainty—the uncertainty intrinsic to the agent's own predictive modules.

Despite the effectiveness of PLR, its usage of value losses makes it susceptible to unreliable scores under aleatoric uncertainty—sources of irreducible uncertainty due to inherent randomness in the environment [10, 11]—as the value loss does not distinguish between epistemic and aleatoric uncertainty. In this paper, we propose using a measure of episodic return dispersion for assessing a level's learning potential. We discuss how value losses can struggle in the presence of aleatoric uncertainty, and how the proposed dispersion-based method can help us to partially address this issue. However, we find PLR using return dispersion leads to inconsistent improvements in test performance across the environments in the Procgen Benchmark, improving test returns in some environments, while hurting it in others. Moreover, in its current instantiation, our idea is impractical due to its sample inefficiency in requiring multiple rollouts per level, although we will conclude the paper by discussing how this inefficiency can be addressed in future work.

2 Background: Prioritized Level Replay

Prioritized Level Replay is an adaptive curriculum learning algorithm which varies the probability that an RL agent trains on particular *levels* (e.g. configurations of the environment or task) as a function of the level's priority. The algorithm prioritizes levels based on a mixture of a scoring metric

Under review at ICBINB Workshop @ NeurIPS 2021. Do not distribute.

aimed at estimating the learning potential of the agent on the level, and of how long ago the level was last acted on by the policy (as a mechanism to keep learning potential estimates up-to-date). As it pertains to learning potential, the scoring criterion is the only component of PLR that we will focus on here.

Suppose we run a policy π_θ on a level l_i for T steps until termination, resulting in a trajectory τ. PLR computes the level's score as a magnitude of the GAE averaged across T steps:

$$S_i(\tau, \pi) = \frac{1}{T}\sum_{t=0}^{T}\left|\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}\right| = \frac{1}{T}\sum_{t=0}^{T}\left|\sum_{k=t}^{T}(\gamma\lambda)^{k-t}\delta_k\right|, \tag{1}$$

where $\delta_t = r_t + \gamma V(s_{t+1}) - V(s_t)$ is the one-step TD error. The λ hyperparameter trades off the bias and variance of the advantage estimator. The two extreme cases with the most bias and most variance correspond to $\lambda = 0$ and $\lambda = 1$: $A_t^{\mathrm{GAE}(\gamma,0)} = r_t + \gamma V(s_{t+1}) - V(s_t)$ and $\hat{A}_t^{\mathrm{GAE}(\gamma,1)} = \sum_{k=0}^{T}\gamma^k r_{t+k} - V(s_t)$. Unsurprisingly, the latter Monte-Carlo estimate proved unsuitable as a PLR score due to its high variance, while $\mathrm{GAE}(\gamma, 0)$ and $\mathrm{GAE}(\gamma, 0.95)$ both tend to work well [7].

Jiang et al. [7] interprets the score in Eq. 1 as the L1 value loss, $\left|V_t^{\mathrm{target}} - V_t\right|$. This inspired the formulation of the Value Correction Hypothesis, which states that the levels most conducive to learning are those with the highest value losses. PLR computes these value losses based on the single, most recent trajectory for each level. However, in settings with aleatoric uncertainty, such scores can become unreliable. Consider the case of a partially-observable level, where the agent finds itself in the middle of a long corridor, and the unobserved goal can be at either end. Even if the agent has a perfect strategy for checking both sides of the corridor, it will still incur non-zero value losses. Relatedly, value losses can be arbitrarily high if the rewards are stochastic, as we increase the reward variance. In both of these cases, value losses cannot disentangle epistemic and aleatoric sources of uncertainty, thereby potentially wrongly prioritizing between levels with high aleatoric uncertainty, but in which the policy is already close to optimal, and levels with low aleatoric uncertainty, in which the policy can still do better. Therefore, value losses are liable to improperly capturing the desired notion of learning potential in stochastic settings.

Despite this shortcoming, PLR with L1 value loss scores leads to significant generalization gains on the Procgen Benchmark. However, as pointed out in Raileanu and Fergus [12], even though Procgen levels have deterministic transitions, the value function will still capture irreducible uncertainty due to observational feature aliasing across levels—the overlap between state representations as learned by the value network. This effect can be a consequence of using unsuitable network architectures, or directly stem from partial observability as in the corridor example above. In both cases, it leads to less reliable value loss scores.

3 PLR with Dispersion Scores

Consider a level that is easy for the agent to solve. On such a level, the agent will consistently attain high returns. Likewise, for a level that is too hard, the agent will consistently receive low returns. In both cases, there is little signal from which the agent can learn. In contrast, levels on the frontier of the agent's current abilities will be solved sporadically, leading to returns that vary across episodes, leading to a rich learning signal. To identify those levels, one can measure how dispersed the returns are, for instance, using the standard deviation (SD) or the coefficient of variation (CV): the ratio of the standard deviation to the mean. CV is scale-invariant and its asymmetric behaviour as shown in Figure 1 reflects one way of how we might want to prioritize levels: difficult levels that are rarely solved would be given a higher priority.

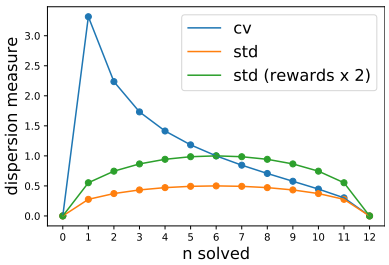

Figure 1: Coefficient of variation σ/μ and standard deviation σ of returns across 12 episodes as a function of the number of solved episodes, where max return is +1. SD, unlike CV, is not scale-invariant, so multiplying rewards by a factor of two leads to larger σ. We set CV to 0 when both σ and μ are 0.

Similarly to the value loss, the proposed return dispersion score conflates the epistemic and aleatoric sources of randomness in the case of stochastic returns. However, in the

case of aleatoric uncertainty due to partial observability, return dispersion scores can help alleviate potential inaccuracies in level scores resulting from value function bias due to feature aliasing. Return dispersion scores have the added benefit of being easier to combine with various base RL algorithms. RL algorithms vary significantly in how their value losses are computed, making it necessary to customize implementations of value-loss scores for each algorithm. In contrast, computing return dispersion is agnostic to the underlying RL algorithm. As we point out in the Appendix, there are, however, similarities between value losses and the return dispersion.

4 Experiments

For our experiments, we extend the original PLR implementation[1], and set the default hyperparameters unless mentioned explicitly. In Figure 2, we use the Procgen Benchmark to compare variants of PLR with the following scoring functions: average GAE magnitude as in Eq.1 (L1 value loss PLR), CV of returns, and SD of returns. The latter two scores are computed based on the undiscounted and unnormalized returns from 12 episodes for each level sampled during a training iteration. These additional episodes are only used to compute the CV or SD of returns, and are not used for PPO updates. In the Appendix, we include an experiment with a higher number of evaluation episodes, and provide some extra results for more variants of value-based PLR.

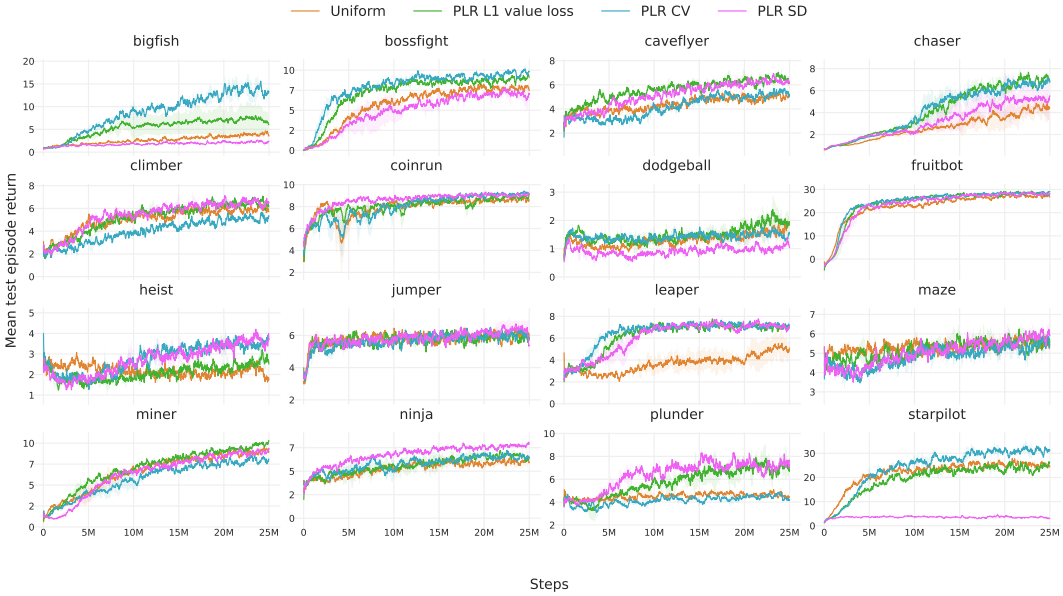

Figure 2: Mean test returns on the Procgen Benchmark during training. The returns after each training update are computed on 10 random test levels. Each PLR variant ran with 3 random seeds. The baseline method samples training levels uniformly. Environment steps on the x-axis only count the transitions used for training, thus excluding the additional steps for CV/SD evaluation episodes.

Our motivation for measuring the dispersion of returns mainly suits environments where a single reward is given for a successful completion of a level. In Procgen, these environments are Coinrun, Jumper, Leaper, Heist, Maze and Ninja. On each of these games, PLR with either CV or SD performs better or no worse than the L1 value loss. In other environments, like Bigfish, Chaser, Dodgeball, Miner, Climber and Bossfight, the agent aims to obtain a sequences of rewards accumulating over the course of a successful episode. Thus, our strategy of computing the dispersion of returns also applies.

Caveflyer is an example of an environment where dispersion-based scores may fail. In this environment, the agent must navigate through a cave to reach a goal, at which point the episode terminates and the agent receives a completion reward. However, the agent can obtain additional rewards by destroying objects along the way. These rewards can inflate the return variance, rendering CV or

[1] github.com/facebookresearch/level-replay

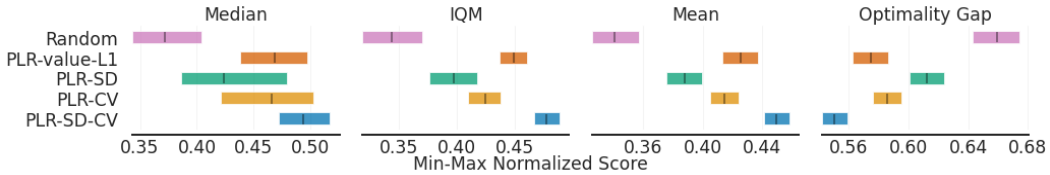

Figure 3: Aggregate metrics: median, interquartile mean, mean and the optimality gap (1 - mean) with 95% CIs. These metrics are computed over all 16 (environments) $\times$ 3 (seeds) normalized test scores. For each game, scores are normalized using the minimum and the maximum achievable scores. For PLR-SD-CV, we use PLR-SD test scores from Caveflyer, Climber, Ninja, Plunder and PLR-CV scores from the rest of the environments.

SD scores uninformative. Meanwhile, Plunder is an example environment where CV-based PLR performs particularly poorly. One explanation for this could be that it prioritizes levels that are too difficult from the start, making it impossible for the agent to learn a good policy.

We see that SD struggles with environments like Starpilot and Bigfish. We attribute this to SD being unsuitable for comparing levels featuring large shifts in mean returns. On the other hand, CV, which measures the variability in relation to the mean, performs slightly better than L1 value loss on these two environments.

To see how these methods perform on average across all 16 environments, we ran the aggregate analysis from Agarwal et al. [13] using the final scores on 1000 test levels. Figure 3 shows no improvements in comparison to PLR with L1 value loss. However, we see that choosing the best performing method out of CV or SD for each environment, as we do post hoc for PLR-SD-CV, results in a marked improvement in test performance, suggesting that an automated mechanism for choosing between CV or SD may make dispersion-based scores a stronger alternative to L1 value loss in this domain.

5 Discussion

We discussed two ways in which aleatoric uncertainty can lead to unreliable PLR scores, namely through stochastic rewards or feature aliasing. In order to address the feature-aliasing case, we proposed to score levels based on the dispersion of returns. We empirically showed that, with PLR, such dispersion scores can improve upon value loss scores. Nevertheless, the inconsistent improvements across the Procgen Benchmark using SD and CV scores show that the choice of the dispersion measure has a significant impact on performance. Thus, an interesting continuation of this work would be to determine whether there are better dispersion measures for PLR. Alternatively, we might be able to find a suitable reward transformation that makes SD work well across all environments, or, perhaps, a mechanism for choosing the dispersion measure upfront depending on the properties of the environment.

In our current implementation, evaluating CV and SD scores requires collecting additional episodes, making it extremely sample-inefficient compared to standard PLR. Future work is required to make our idea practical in terms of sample efficiency, and existing estimators for the variance of returns [14] can help in this regard. Further, while dispersion scores improve the reliability of PLR scores when dealing with aleatoric uncertainty due to feature-aliasing, they do not address the case of stochastic rewards. We plan to extend the ideas developed here—namely using PLR scores based on the return distribution—to handle the case of stochastic rewards in future work.

Lastly, while PLR has been primarily motivated via the Value Correction Hypothesis, this explanation only provides a heuristic argument for the success of this method. We hypothesize its effectiveness may be tied to the update mechanics of the underlying RL algorithm. Jiang et al. [7] used PPO, for which GAE is used in the policy gradient estimator, $\hat{g} = \hat{\mathbb{E}}_t[\nabla_\theta \log \pi_\theta(a_t \mid s_t)\hat{A}_t^{\mathrm{GAE}(\gamma,\lambda)}]$. Larger GAE magnitudes (or L1 value losses) correlate with larger parameters updates, which may lead to both faster training and improved generalization [15]. This may provide an alternative explanation for the benefits of PLR that may be worth further investigation.

References

[1] Charles Packer, Katelyn Gao, Jernej Kos, Philipp Krahenbuhl, Vladlen Koltun, and Dawn Song. Assessing generalization in deep reinforcement learning. *Preprint arXiv:1810.12282*, 2019.

[2] Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. Generalization in reinforcement learning with selective noise injection and information bottleneck. In *Advances in Neural Information Processing Systems*. 2019.

[3] Rishabh Agarwal, Marlos C. Machado, Pablo Samuel Castro, and Marc G Bellemare. Contrastive behavioral similarity embeddings for generalization in reinforcement learning. In *International Conference on Learning Representations*, 2021.

[4] Chiyuan Zhang, Oriol Vinyals, Rémi Munos, and Samy Bengio. A study on overfitting in deep reinforcement learning. *Preprint arXiv:1804.06893*, 2018.

[5] Dibya Ghosh, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine. Why generalization in rl is difficult: Epistemic POMDPs and implicit partial observability. *Preprint arXiv:2107.06277*, 2021.

[6] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[7] Minqi Jiang, Edward Grefenstette, and Tim Rocktäschel. Prioritized Level Replay. In *The International Conference on Machine Learning*. 2021.

[8] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *Preprint arXiv:1707.06347*, 2017.

[9] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2016.

[10] Armen Der Kiureghian and Ove Ditlevsen. Aleatory or epistemic? does it matter? *Structural safety*, 31(2): 105–112, 2009.

[11] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? In *Advances in Neural Information Processing Systems 30*, 2017.

[12] Roberta Raileanu and Rob Fergus. Decoupling value and policy for generalization in reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, 2021.

[13] Rishabh Agarwal, Max Schwarzer, Pablo Samuel Castro, Aaron Courville, and Marc G. Bellemare. Deep reinforcement learning at the edge of the statistical precipice. *Preprint arXiv:2108.13264*, 2021.

[14] Aviv Tamar, Dotan Di Castro, and Shie Mannor. Learning the variance of the reward-to-go. *Journal of Machine Learning Research*, 17(13):1–36, 2016.

[15] Yuanzhi Li, Colin Wei, and Tengyu Ma. Towards explaining the regularization effect of initial large learning rate in training neural networks. In *Advances in Neural Information Processing Systems 32*, 2019.

[16] Anonymous. Replay guided adversarial environment design. *Under Review*, 2021.

Appendix

A.1 Additional experimental results

PLR with other value-based scores

It is useful to recall the following definition of the advantage function $A(a_t, s_t)$ whose estimates $\hat{A}_t$ are used in the construction of the value loss:

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) = Q(s_t, a_t) - \mathbb{E}_{a \sim \pi_\theta(a|s_t)} Q(s_t, a). \tag{2}$$

From this follows the interpretation of $A(s_t, a_t)$ as the advantage, measured by the difference in expected returns, of taking an action a_t in state s_t in comparison to all other actions under our policy. Positive advantages point us to better actions, whose probability under π_θ should be increased. Usually, there are very few good actions in comparison to the number of bad actions one could take in each state, and there are many more ways in which an action can be bad. As Schulman et al. [9] points out, this asymmetry can be exploited for prioritization. In our case, instead of the GAE magnitude, we can use $\max(\hat{A}_t, 0)$ or $\exp(\hat{A}_t)$. This scoring function, also called the *positive value loss*, has previously been studied in the context of PLR, and shown to be effective in a continuous control domain [16]. As we show in Figure 4, PLR with positive value loss, i.e. $\max(\hat{A}_t, 0)$ averaged across time steps, tends to perform worse than PLR with L1 value loss in the Procgen domain.

In Figure 4, we also show the results of PLR with L1 value loss whose $V_t^{\text{target}} = \sum_{k=0}^{T} \gamma^k r_{t+k}$—the Monte-Carlo (MC) estimate of the return. For the majority of Procgen games, it performs worse than the uniform-sampling baseline. This is likely because of the high variance of the MC estimate.

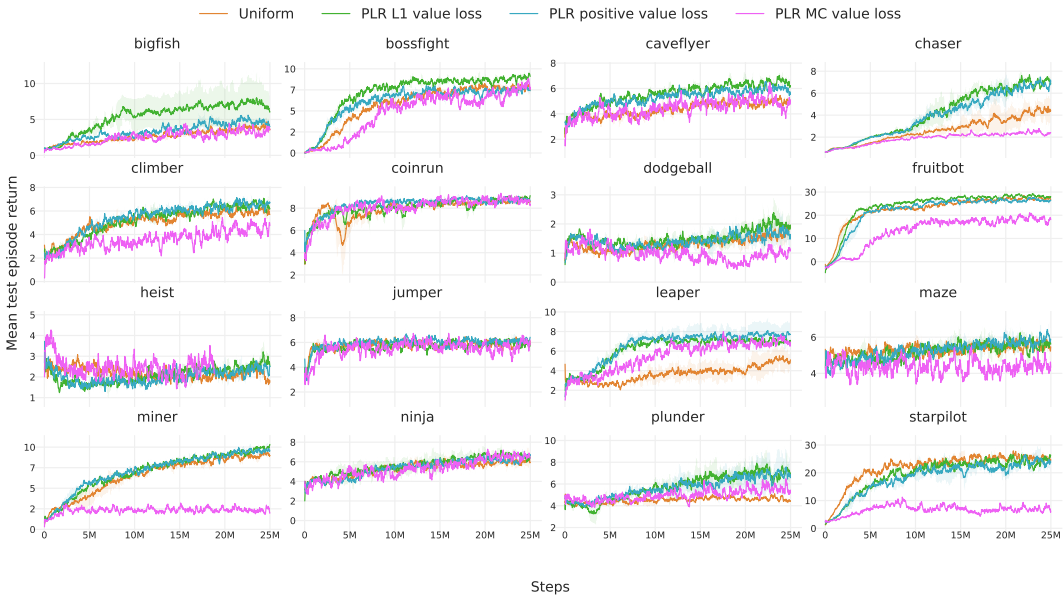

Figure 4: Mean test returns on 16 Procgen games measured during the course of training. The returns after each training update are computed on 10 test levels. Each method ran with 3 random seeds, except for PLR with MC value loss, which we ran only once. The baseline method samples training levels uniformly.

Experiments with more evaluation episodes

Estimates of the CV based on 12 episodes can be noisy, and so we checked if it helps to increase this number. Figure 5 shows the results for the Leaper environment with 12 and 48 episodes used for estimating the CV. While the latter converges on a slightly higher value, the improvements are marginal in comparison to the amount of extra compute. However, it indicates possible gains from having less noisy estimates of the dispersion.

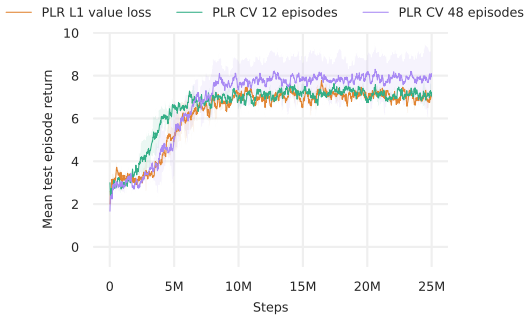

Figure 5: Mean test return over the course of training in the Leaper game with 12 and 48 episodes for evaluating CV. Each method was run with 3 random seeds.

A.2 Connection between values losses and the variance of returns

To see the connection between advantage estimates and the variance of returns, let us start from the definition of $GAE(\gamma, 1)$:

$$\left(\hat{A}^{GAE(\gamma,1)}(s_t, a_t)\right)^2 = \left(\sum_{k=0}^{\infty} \gamma^k r_{t+k} - V(s_t)\right)^2$$
$$= \left(\sum_{k=0}^{\infty} \gamma^k r_{t+k} - \mathop{\mathbb{E}}_{\substack{a_{t:}, r_{t:}, \\ s_{t+1:}}} \left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \mid s_t\right]\right)^2$$

Compare this to the variance of a random variable $R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k}$ for some fixed s_t:

$$\text{Var}(R_t|s_t) = \mathop{\mathbb{E}}_{\substack{a_{t:}, r_{t:}, \\ s_{t+1:}}} \left[(R_t - \mathop{\mathbb{E}}_{\substack{a_{t:}, r_{t:}, \\ s_{t+1:}}} (R_t|s_t))^2\right]$$

We see that $\left(\hat{A}^{GAE(\gamma,1)}(s_t, a_t)\right)^2$ estimates $\text{Var}(R_t|s_t)$ using a single sample, i.e. a trajectory that starts from s_t and a_t, and the mean of R_t given by the value function.

In our approach, when computing the SD of returns, we effectively estimate $\text{Var}(R_0|s_0)$ based on 12 samples of the random variable $R_0 = \sum_{t=0}^{T} r_t$.

For $GAE(\gamma, 0)$, we can follow similar steps:

$$\left(\hat{A}^{GAE(\gamma,0)}(s_t, a_t)\right)^2 = (r_t + \gamma V(s_{t+1}) - V(s_t))^2$$

In this case, the random variable is $R_t = r_t + V(s_{t+1})$ and its variance is:

$$\text{Var}(R_t|s_t) = \mathop{\mathbb{E}}_{\substack{a_t, r_t, \\ s_{t+1}}} \left[\left(r_t + V(s_{t+1}) - \mathop{\mathbb{E}}_{\substack{r_t, a_t, \\ s_{t+1}}} [r_t + V(s_{t+1})]\right)^2\right]$$
$$= \mathop{\mathbb{E}}_{\substack{a_t, r_t, \\ s_{t+1}}} \left[(r_t + V(s_{t+1}) - V(s_t))^2\right]$$

Thus, $GAE(\gamma, 0)$ is an estimator of the variance of R_t for a fixed s_t and a sample of a_t, r_t and s_{t+1}.