Automated Curriculum Learning by Rewarding Temporally Rare Events

Niels Justesen
IT University of Copenhagen
Copenhagen, Denmark
noju@itu.dk

Sebastian Risi
IT University of Copenhagen
Copenhagen, Denmark
sebr@itu.dk

***Abstract*—Reward shaping allows reinforcement learning (RL) agents to accelerate learning by receiving additional reward signals. However, these signals can be difficult to design manually, especially for complex RL tasks. We propose a simple and general approach that determines the reward of pre-defined events by their rarity alone. Here events become less rewarding as they are experienced more often, which encourages the agent to continually explore new types of events as it learns. The adaptiveness of this reward function results in a form of automated curriculum learning that does not have to be specified by the experimenter. We demonstrate that this *Rarity of Events* (RoE) approach enables the agent to succeed in challenging VizDoom scenarios without access to the extrinsic reward from the environment. Furthermore, the results demonstrate that RoE learns a more versatile policy that adapts well to critical changes in the environment. Rewarding events based on their rarity could help in many unsolved RL environments that are characterized by sparse extrinsic rewards but a plethora of known event types.**

I. Introduction

Deep reinforcement learning and deep neuroevolution have achieved impressive results learning to play video games [14] and controlling both simulated and physical robots [2, 9, 12, 23]. These approaches, however, struggle to learn in environments where feedback signals (also called rewards) are sparse and/or delayed. A popular way to overcome this issue is to shape the reward function with prior knowledge such that the agent receives additional rewards to guide its learning process [19, 20, 26]. Another approach is to gradually increase the difficulty of the environment to ease learning through curriculum learning [4, 38]. Both approaches are time-consuming, require substantial domain knowledge and are especially difficult to implement for complex environments. In this paper, we propose a simple method that automatically shapes the reward function during training and performs a form of curriculum learning that adapts to the agent's current performance. The only required domain knowledge is the specification of a set of positive events that can happen in the environment (e.g. picking up items, moving, winning etc.), which is easy to implement if raw state changes are accessible.

The method introduced in this paper rewards a reinforcement learning agent by the rarity of experienced events such that rare events have a higher value than frequent events. The idea is to completely discard the extrinsic reward and instead motivate the agent intrinsically towards a behavior that explores the pre-defined events. As the agent first experiences certain types of events that are relatively easy to learn (e.g. moving around and picking up items) they will slowly become less rewarding, pushing the agent to explore rare and potentially more difficult events. Thus by only rewarding events for their rarity, the system performs a form of automated curriculum learning.

The goal of this approach is to learn through a process of *curiosity* rather than optimizing towards a difficult pre-defined goal. We apply our method, called *Rarity of Events* (RoE), to learn agent behaviors from raw pixels in the VizDoom framework [16]. While our approach could be applied to any reward-based learning method and possibly also fitness-based evolutionary methods, in this paper we train deep convolutional networks through the actor-critic algorithm A2C [25]. In the future, RoE could offer a new way to learn versatile behaviors in increasingly complex environments such as StarCraft, which is a yet unsolved reinforcement learning problem [36].

The paper is structured as follows. We first review relevant previous work, including related approaches in Section II. After explaining RoE (Section III), we demonstrate the usefulness of the method on five challenging VizDoom scenarios with sparse rewards and show how RoE learns a versatile behavior that can adapt to critical changes in the environment (Section V).

II. Previous Work

A. Deep Reinforcement Learning

Deep reinforcement learning can be used to learn agent behaviors in video games directly from screen pixels, including Atari games [24], first-person shooters [16, 19, 38], and car racing games [25]. These methods are typically variants of Deep Q Networks (DQN) [24] or actor-critic methods with parallel actor-learners such as Asynchronous Advantage Actor-Critic (A3C) [25]. Deep neuroevolution have also shown promising results on the Atari games and parallelizes better [31, 35].

A key requirement for these methods to work out of the box are frequent and easy obtainable reward signals from the environment that can guide learning towards an optimal behavior. A popular environment where this is not the case because of sparse rewards, is the Atari game Montezuma's

Revenge; for this game, both DQN and A3C variants fail [24, 25].

The lack of frequent reward signals can be overcome by reward shaping where a smoother reward function is designed using prior domain knowledge [19, 20] or by gradually increasing the difficulty of the environment (e.g. the level itself or the NPCs' behaviors) to ease learning through curriculum learning [4, 38]. Related to curriculum learning is a method called *Power Play* that searches for new unsolvable problems while the agent is trained to progressively match the difficulty of the environment [33]. Another related approach is Hierarchical reinforcement learning where a meta-controller controls one or more sub-policies that are trained to reach sub-goals (equivalent to events) [7, 17].

B. Curiosity & Intrinsic Motivation

Curiosity-driven learning is another approach that ignores the extrinsic rewards and guided by an intrinsic motivation seek to explore new situations for fun or challenge [15, 27, 30]. One theory of intrinsic motivation is *reduction of cognitive dissonance*; the motivation to learn a cognitive model that can explain and predict sensory input [11, 28]. This theory has also been formalized in the context of reinforcement learning in which agents are intrinsically rewarded when observing temporarily novel, interesting, or surprising patterns based on their own model [32]. This also follows the idea of *optimal incongruity*, that discrepancy between the perceived and what is usually perceived produce a high stimulus; thus novel situations that yet lie within our current understanding are highly rewarding [5, 13].

One way of implementing intrinsic motivation is to model the expected learning progress $\zeta(s, a)$ of a state-action pair [22]. The Intrinsic Curiosity Module (ICM) is another approach that encodes states s_t and s_{t+1} into features $\Phi(s_t)$ and $\Phi(s_{t+1})$ and determines the intrinsic reward based on the prediction error of these features and the forward model's features [29]. State-density models, that assign probabilities to screen images, can be learned together with a policy and then determine intrinsic motivation as the model's temporal change in prediction, such that surprising screen images produce higher rewards [3].

C. Novelty Search

The pursue of novel situations also shares some similarities with novelty search [21] in evolutionary computation. The idea in novelty search is to only search for novel behaviors instead of optimizing a specific objective function directly. Both novelty search and our approach RoE push the search towards unexplored areas; however, novelty search does so for a population of individuals where novelty is defined as the behavioral distance to other behaviors in the population. Our approach is trained through reinforcement learning and novelty (or rather rarity of events) is based on experiences of previous versions of the policy.

D. VizDoom

Our approach will be tested in VizDoom, an AI research platform based on the commercial video game Doom that allows reinforcement learning to learn from raw visual information [16]. Doom is interesting as we can create several diverse environments, including some that are very challenging to reinforcement learning methods due to sparse and delayed rewards. Several deep reinforcement learning approaches exist that can learn to play challenging scenarios in Doom. These approaches include auxiliary learning [18, 19], game-feature augmentation [6, 8, 10], manual reward shaping [8, 10, 19], and curriculum learning [38]. A very different approach applies neuroevolution on top of a pre-trained auto-encoder [1]. Our approach builds on top of a vanilla implementation of the reinforcement learning algorithm A2C without any of these extensions.

III. Approach

This section will describe our approach *Rarity of Events* (RoE) and how it was integrated with A2C in VizDoom.

A. Rewarding Temporally Rare Events

The reward function in RoE adapts throughout training to the policy's ability to explore the environment. By rewarding events based on how often they occur during training, the agent is intrinsically motivated towards exploring new parts of the environment rather than having a single goal that might be difficult to obtain directly. In effect, the approach performs a form of curriculum learning since events are rewarded based on the agent's current ability to obtain them. As the agent learns, it becomes less interested in events that are frequent and *curious* about newly discovered events.

Our method requires a set of pre-defined events, and the reward $R_t(\epsilon_i)$ for experiencing one of these events ϵ_i at time t is determined by its temporal rarity $\frac{1}{\mu_t(\epsilon_i)}$, where $\mu_t(\epsilon_i)$ is the temporal episodic mean occurrence of ϵ_i at time t, i.e. how often ϵ_i occurs per episode at the moment. The mean occurrences of events are clipped to be above a lower threshold τ (we used 0.01 such that the maximum reward for any event is 100). For a vector of event occurrences x, such that x_i is the number of times ϵ_i occurred in a game step, the reward is the sum of all event rewards:

$$R_t(x) = \sum_{i=1}^{|x|} x_i \frac{1}{\max(\mu_t(\epsilon_i), \tau)}$$

The rarity measure $\frac{1}{\mu_t(\epsilon_i)}$ is not arbitrary but is designed such that all events have equal importance. If any event ϵ_i is experienced n times during an episode, and $n = \mu_t(\epsilon_i)$ (which is the expected amount), then the accumulated reward for ϵ_i is 1 regardless of the rarity. I.e. this means that in theory all events have equal importance. In practice, the policy might learn that some events have a negative influence on the occurrence of others.

B. Determining the Temporal Episodic Mean Occurrence

There are arguably many ways to determine the temporal episodic mean occurrence $\mu_t(\epsilon_i)$; here we employ a simple approach that nevertheless achieves the desired outcome. Whenever an episode during training reaches a terminal state a vector containing the occurrence of events ϵ in this episode is added to a buffer of size N. The size of the buffer determines the adaptability of the reward function. If N is small, the agent quickly becomes *bored* of new events as it easily forgets its rarity in the past. If N is large, the agent stays *curious* longer. The temporal episodic mean occurrence $\mu_t(\epsilon)$ is then determined as the mean of all records in the buffer, i.e. the episodic mean of the last N episodes.

C. Events in Doom

For our VizDoom experiments, we track 26 event types that can be triggered in the game by implementing a function that determines which events occur in every state transition (i.e. in each time step). The event types include movement (one unit), shooting (decrease in ammo), picking up an item (one for each type; health pack, armor, ammo, and weapons 0–9), killing (one for each weapon type 0–9 as well as one regardless of weapon type). Movement events happen when the agent has traveled one unit from the position of the last movement event (or the initial position if the agent has not moved yet).

D. Policy

The presented reward shaping approach can be applied to most (if not all) reinforcement learning methods that learn from a reward signal. It can also be used for evolutionary approaches such as Evolution Strategies by defining fitness as the sum of rewards in an episode but we do not yet have experimental results showing how well that would work. In our experiments, we demonstrate the usefulness of RoE by integrating it with A2C. A standard policy network is employed that has three convolutional layers followed by a fully connected layer of 512 units, and a policy and value output. We use filter sizes of [32, 64, 32] with strides [4, 2, 1], ReLU activations for hidden layers, and softmax for the policy output.

The input is a single frame of 160×120 pixels in grayscale, cropped by removing 10 pixels on top/bottom and 30 pixels on the sides and then resized to 80×80. In most of the scenarios, the agent can perform four actions: attack, move forward, turn left, and turn right. In this case, the policy output has $2^4 = 16$ values to allow any combination of the four actions. The event buffer is updated whenever a worker reaches a terminal state. The rewards from VizDoom, which is between -100 and 100, is normalized to into the interval $[0, 1]$. Rewards based on our approach is not normalized and is between $[0, 100]$ (due to $\tau = 0.01$), while for all events where $\mu_t(\epsilon_i) \geq 1$ the reward will be between $[0, 1]$ (following the formula in Section III-A).

E. Advantage Actor-Critic (A2C)

The deep networks in this paper are trained with the deep reinforcement learning algorithm A2C, a synchronous variant of Asynchronous Advantage Actor-Critic (A3C) [25], which is among state-of-the-art in a wide range of environments [34, 37, 39].

A2C is an actor-critic method that optimizes both a policy π (the actor) and an estimation of the state-value function $V(s)$ (the critic). Parallel worker threads share the same model parameters and synchronously collect trajectories $(s_t, s_{t+1}, a_t, r_{t+1})$ for t_{max} game steps where after the model's parameters are updated. Threads restart new episodes individually when they are done. The discounted return $R_t = \sum_{i=1}^{k-1} \gamma^i r_{t+i} + \gamma^k V(s_{t+k})$, where k is the number of trajectories collected after t, and the advantage $A(s_t, a_t) = R_t - V(s_t)$ is determined for each step, for every worker. A2C then uses the traditional A3C update rules in [25] based on the policy loss $\log \pi(a_i|s_i)A(s_i)$ and value loss; the mean squared error between the experienced R_t and the predicted $V(s_t)$: $\frac{1}{2}(R_t - V(s_t))^2$. With A2C the parameters are updated synchronously in batches in contrast to A3C.

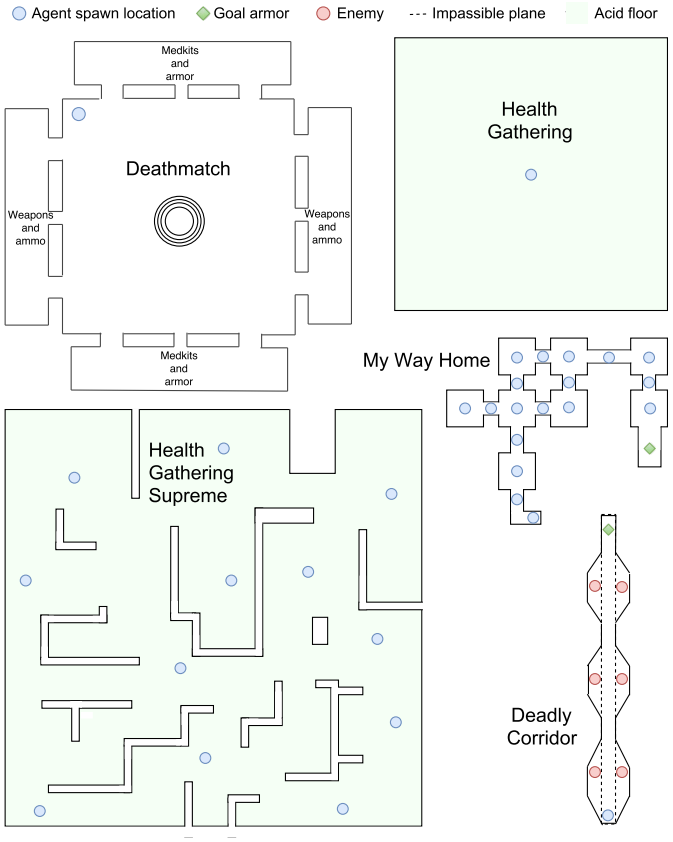

Fig. 1: The five ViZdoom scenarios used in our experiments.

.

IV. VizDoom Testing Scenarios

This section describes the five VizDoom scenarios used in our experiments. They all have sparse and/or delayed rewards and are therefore suitable for our approach. The scenarios

are from the original VizDoom [16] repository[1]. We have, however, rescaled some of the rewards to be coherent across scenarios. The agent can move forward, turn left, turn right, and shoot if nothing else is stated. Screenshots from these scenarios are shown in Figure 2 and a top-down view of the scenarios in Figure 1.

1) Health Gathering: The goal is to survive as long as possible in a square room with an acid floor that deals damage periodically. Medkits spawn randomly in the room and can help the agent to survive as they heal when picked up. The agent is rewarded 1 for every time step it is alive, and -100 for dying. The maximum episode length is 2,100 time steps. The agent cannot shoot.

2) Health Gathering Supreme: Same as *Health Gathering* but within a maze.

3) My Way Home: The goal is to pick up an armor, which gives a reward of 100 and ends the scenario immediately. The agent is rewarded -0.1 for every time step it is alive, cannot shoot and is randomly spawned at one of the spawn locations with a random rotation.

4) Deadly Corridor: Similarly to *My Way Home*, the goal is to pick up an armor, which gives a reward of 100 and ends the scenario immediately. The armor is located at the end of a corridor which is guarded by enemies on each side. The agent must kill most, if not all, of the enemies to reach it, but receives a -100 reward if it dies. The original reward shaping function (the distance to the armor) has been removed to make it harder. The maximum episode length is 2,100 time steps.

Fig. 2: From top-left to bottom-right: Screenshot from *Deathmatch*, *My Way Home*, *Health Gathering Supreme*, and *Deadly Corridor*. Notice that in some scenarios the agent cannot shoot. The scenario *Health Gathering* is similar to *Health Gathering Supreme* but without walls within the room.

5) Deathmatch: The agent spawns in a large battle arena with an open area in the middle and four rooms, one in each direction, each containing either medkits and armor, or weapons (chainsaw, super shotgun, chaingun, rocket launcher, and plasma gun) and ammunition for each weapon. The maximum episode length is 4,200 time steps. The agent is rewarded (the amount is shown in brackets) when killing an enemy Zombieman (100), shotgunGuy (300), MarineChainsawVzd (300), Demon (300), ChaingunGuy (400), HellKnight (1,000), that spawn randomly on the map when the scenario has started.

To test how well the approach can adapt to new scenarios, five variations of *Deathmatch* were also created that only include a certain weapon type. These scenarios are called *Deathmatch Chainsaw*, *Deathmatch Chaingun*, *Deathmatch Shotgun*, *Deathmatch Plasma*, and *Deathmatch Rocket* to denote which weapon remains on the map. The ammunition for the other weapons was also removed.

A2C	
Learning rate	7e-4
γ (discount factor)	0.99
Entropy coefficient	0.01
Value loss coefficient	0.5
Learning rate	0.0007
Max. gradient-norm	0.5
Worker threads	4 (16 in DM)
t_{max} (Steps per. update)	20
Batch size	64
Frame skip	4
RMSprop Optimizer	
ϵ	1e-5
α	0.99
RoE	
N (event buffer size)	100
τ (mean threshold)	0.01

TABLE I: Experimental configurations for A2C and A2C+RoE. 16 worker threads were used in *Deathmatch* and its variations.

V. Results

We tested A2C with our approach *Rarity of Events* (A2C+RoE) on the five challenging VizDoom scenarios described in Section IV. The *Deathmatch* variations were not used for training. As a comparison baseline, A2C was also trained using the extrinsic reward from the environment as described in Section IV.

When training with A2C+RoE, the agent did not have access to the extrinsic reward throughout training but only the intrinsic reward based on the temporal rarity of the predefined events. The algorithms ran for 10^7 time steps for each scenario and 7.5×10^7 for the Deathmatch scenario. For both A2C and A2C+RoE we save the model parameters whenever the mean extrinsic reward improves across all workers. The complete configurations for A2C and A2C+RoE are shown in Table I and the code for the experiments and trained models will be published online shortly. Videos of the learned policies are available on YouTube[2].

A. Learned Policies

The A2C baseline did not progress in learning in *Health Gathering Supreme* and *Deadly Corridor*, and only slightly in *Health Gathering* (See Figure 3). These results show that the selected scenarios are indeed difficult using only extrinsic

[1]https://github.com/mwydmuch/ViZDoom/tree/master/scenarios

[2]https://youtu.be/YG-lf732a0U

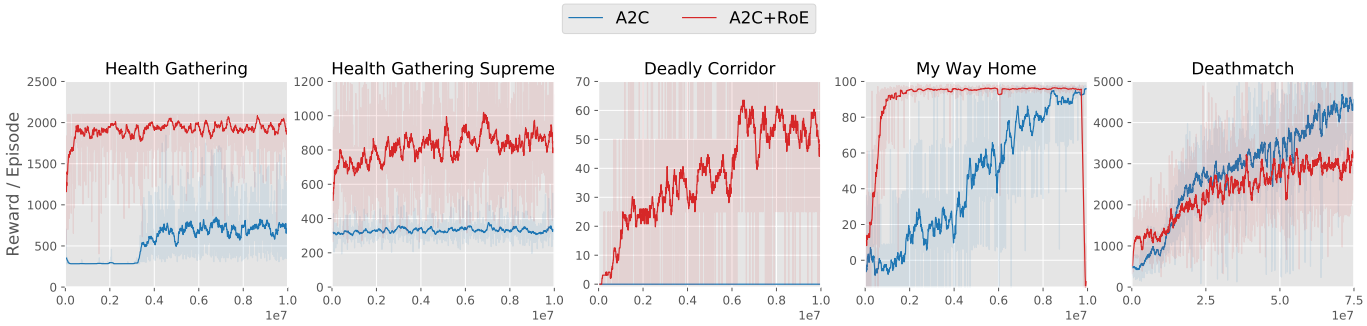

Fig. 3: The reward per episode of A2C and A2C+RoE during training in five VizDoom scenarios (smoothed). A2C is trained from the environment's extrinsic reward while A2C+RoE uses our proposed method without access to the reward. The drop in performance seen in the My Way Home scenario is discussed in-depth in Section V-A.

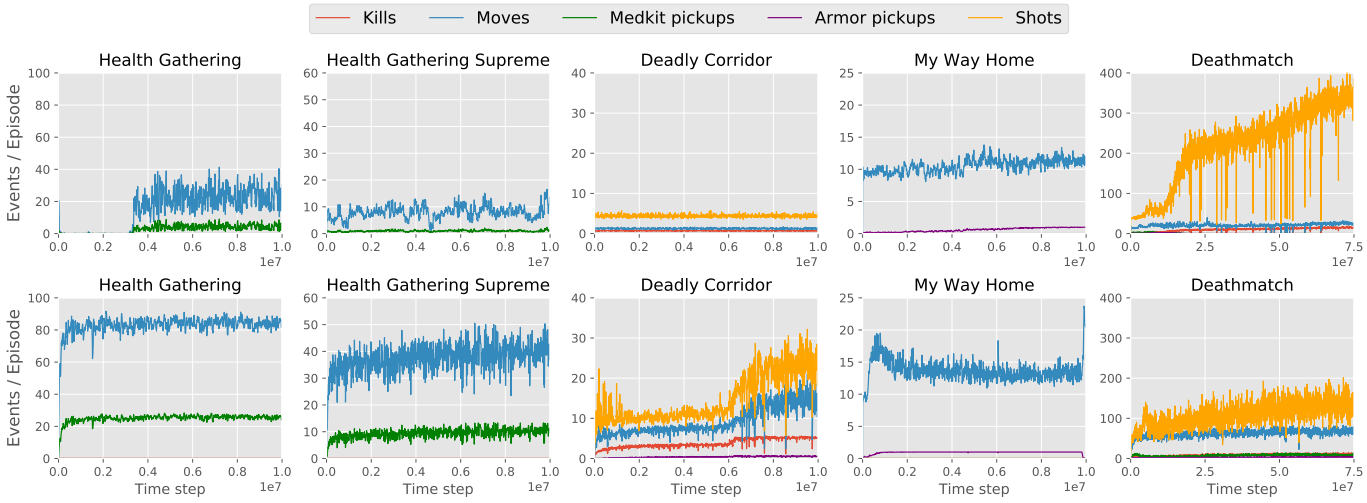

Fig. 4: Episodic mean occurrence of events during training for a subset of the event types in the five VizDoom scenarios. Notice the last spike in the My Way Home scenario with A2C+RoE where the policy ignores the final goal (armor pickup) to prioritize continuous movement around the maze.

Scenario	A2C	A2C+RoE	t-test
Health Gathering	399 (107)	**1261** (533)	$p < 0.0001$
Health Gathering Supr.	305 (60)	**1427** (645)	$p < 0.0001$
Deadly Corridor	0.00 (0.0)	**40** (49)	$p < 0.0001$
My Way Home	96.69 (0.12)	**97.89** (0.01)	$p < 0.0001$
Deathmatch	**4611** (2595)	4062 (2442)	$p = 0.1250$
Deathmatch Chainsaw	1025 (809)	**3750** (3130)	$p < 0.0001$
Deathmatch Chaingun	1487 (1189)	**2852** (2038)	$p < 0.0001$
Deathmatch Shotgun	1375 (941)	**1832** (1752)	$p = 0.0226$
Deathmatch Plasma	**4538** (1537)	3248 (2701)	$p < 0.0001$
Deathmatch Rocket	616 (583)	**1463** (1449)	$p < 0.0001$

TABLE II: Evaluation results averaged over 100 runs. The best results are shown in bold. The five last rows show how the policies that were trained on the original *Deathmatch* scenario generalize to five variations where only one weapon type is available. Standard deviations for each experiment are in parentheses and two-tailed p-values from unpaired t-tests.

rewards as A2C learned a weak policy in three of the five scenarios. In *My Way Home*, A2C does learn a strong behavior that consistently locates and picks up the armor but only after 8-9 million training steps. In *Deathmatch*, A2C learned a very high-performing behavior that directly walks to the plasma gun (the most powerful weapon in the scenario) and shoots from cover towards the center of the map. The behavior is simple but effective until it runs out of ammunition, where after it attempts to find more ammunition and sometimes fails.

Our approach, A2C+RoE learns strong behaviors in all five scenarios using the same setup. The learned behavior in *Deathmatch* does not learn to exclusive use the powerful plasma gun and the performance is not as good as A2C but with no significant difference ($p = 0.125$ using two-tailed t-test). The policy is still effective with over 10 kills per episode. These kills are spread across all weapons that are available, and the behavior is thus much more interesting. We will show in Section V-B that A2C+RoE learned a versatile behavior that can adapt to critical changes in *Deathmatch* in contrast to A2C.

The episodic mean occurrence of events (Figure 4) allows us to analyze how the policies change over time. In *Health Gathering* and *Health Gathering Supreme*, A2C+RoE quickly learns to move ∼80 and ∼30 units per episode, respectively. This behavior could potentially be the reason why it also quickly learns to pickup medpacks. In contrast, it takes a while

to figure out the relationship between movement, medpacks, and survival for A2C, at least in the *Health Gathering* scenario. In *Deadly Corridor*, an interesting behavior by the A2C+RoE policy can be observed. As soon as the agent learns to shoot all the enemies (the red line) and picking up the armor (purple line), it still manages to increase the occurrence of movement at shooting events, by walking back to its initial position while shooting to then come back and pick up the armor. This makes sense as it is intrinsically motivated to experience as many events as possible before finishing the episode.

In *My Way Home*, after the A2C+RoE policy has learned to routinely pick up the armor, it shifted into a different behavior just by the end of the training. The agent learned to avoid the armor to instead continuously move around the maze. We suspect that the policy would shift back to the previous behavior if training was continued, as the movement reward is decreasing and the armor reward is increasing. Since our rarity measure is temporal it might also loop between the two behaviors. As the extrinsically best policy are saved during training, these sudden changes do not affect the final policy. In fact, one might argue that this is exactly what we would like to see, as the policy, which has converged to some optimum, can escape it to find other behaviors.

B. Ability to Adapt

A2C+RoE motivates the agent intrinsically to learn a balanced policy that attempts to experience a good mix of events. Reinforcement learning algorithms that do not do pre-training or proper reward shaping, including our A2C baseline, can easily converge into local optima with a very *narrow* behavior. In this context *narrow* refers to behaviors that act in a very particular way, only utilizing a small subset of the features in the environment. This handicap prevents the learned policies from adapting to critical changes in the environment as they only know one way of behaving.

To test for such adaptivity, the learned policies are evaluated on five *Deathmatch* variations in which critical weapons and ammunition packs have been removed. Note that the policies were not directly trained on these variations. The results in Table II show that A2C+RoE learned a policy that significantly outperforms A2C ($p < 0.05$ using two-tailed t-test) in four out of five *Deathmatch* variations. A2C+RoE learned a policy that is more versatile, capable of using all the weapons in the map, which is why it can easily adapt. Figure 5 shows heat maps (i.e. the proportional time spent at each location during the evaluation) from the evaluation runs of the two policies on *Deathmatch* and its variations. The A2C+RoE policy expresses different strategies depending on the weapon available on the map, while the A2C policy mostly circles around the plasma gun location, regardless of it actually being there. However, if the plasma gun is present, A2C alone does execute a fairly effective strategy, shooting towards enemies in the middle of the map.

The heat maps show that the A2C-policy has learned to stay at only one location on the map from which it can pick up the powerful plasma gun and thereafter shoot efficiently towards enemies in the middle of the map. This can also be seen in the video demonstrations. In the *Deathmatch* variations, where the map only contains two weapons of the same type, the A2C-policy fails to adapt to use the other weapons and instead walks around the area where the plasma gun would have been located.

The A2C+RoE-policy has a more balanced distribution of locations on the map in the *Deathmatch* scenarios. In the variations, a clear change in behavior can be observed when the bot is forced to use a certain type of weapon. For example, in the DM Rocket scenario, it lures enemies into the top and bottom room while efficiently using the rocket's splash damage.

VI. Discussion

While the presented approach worked well in VizDoom it will be important to tests its generality in other domains in the future. The approach is designed to work well in challenging environments that have a plethora of known events and sparse and/or delayed rewards. Video games are thus a very suitable domain. For domains in which reward shaping is not necessary, i.e. the extrinsic reward smoothly leads to an optimal behavior, our approach is less well suited. We imagine the approach should also work well in deceptive environments, such as mazes with dead ends, just as novelty search outperform traditional evolutionary algorithms in such cases [21]. Novelty search and Rarity of Events have the ability to learn interesting behaviors without the need for a goal.

The specification of adequate events is intimately tied to the success of our approach; events that lead to direct negative performance should be avoided. If e.g. the extrinsic reward is negative when the agent wastes ammunition it should not be intrinsically rewarded for the shooting event. A benefit of the presented methods is that events that contribute to the occurrence of other events, such as movement leads to medkit pickups, can lead to a system that performs automated curriculum learning. Other events can be contradicting, such as killing with the chainsaw and killing with the plasma gun, as the agent cannot do both at the same time. Our approach is designed to learn a policy that can balance their occurrences which results in a more versatile behavior. Important future work will test if our approach still works well with hundreds or perhaps thousands of events. A promising testbed for this is StarCraft, where events can easily be defined as the production of each unit and building type, as well as killing different enemy units types. We believe that reinforcement learning methods that are guided by intrinsic motivation are key to solve these challenging environments.

While our A2C baseline reached the best performance in the original *Deathmatch* it can be argued whether it learned to actually play Doom, or whether it just remembered a fixed sequence of actions that lead to the same behavior every time. While it in many cases is useful to find a niche behavior with high performance, we argue that learning a rich and versatile behavior can be more useful. In the domain of video games,

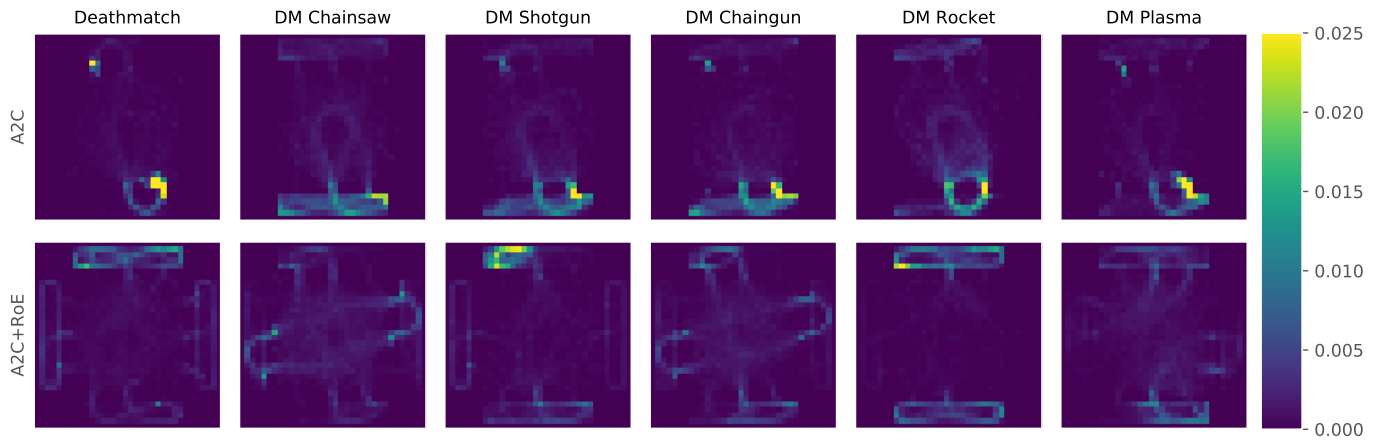

Fig. 5: Heat maps showing the proportional time spent at each location on the map in the *Deathmatch* scenario and its five variations. The values are averaged over 100 runs and clipped at 0.025. The heat maps show that the A2C-policy prefers to stay near the plasma gun, even in the map variations where it is not present, while the A2C+RoE-policy has learned distinct behaviors for each weapon type. The results in Table II shows that the A2C+RoE-policy is able to reach high scores in these variations event though it was never trained on them.

behaviors that explore the game's features are more useful when it comes to automatic testing and will also produce more human-like behavior for NPCs.

Regarding our implementation of the proposed method, future work will explore other methods to determine the episodic mean occurrence of events. Perhaps the approach would work even better if the mean occurrences are discounted over time such that it does not completely forget after N episodes (the event buffer only holds N mean event occurrences).

During our experiments, the best policy was save based on the mean extrinsic reward of all worker threads. Since only four threads were used in some of our experiments, it may have affected the evaluated performance due to randomness in the training rollouts. This approach is, however, more stable with 16 threads as we used in the *Deathmatch* scenarios.

VII. Conclusion

We presented *Rarity of Events* (RoE), a simple reinforcement learning approach that determines reward based on the temporal rarity of pre-defined events. This approach was able to reach high-performing scores in five tested VizDoom scenarios with sparse and/or delayed rewards. It used the same setup and did not have access to the extrinsic reward. The results are significantly better ($p < 0.05$ using two-tailed t-test) than a traditional A2C baseline in four of the five scenarios. Interestingly, the presented approach is able to not only receive a high final reward, but also discovers versatile behavior that can adapt to critical changes in the environment significantly better than the baseline ($p < 0.05$ using two-tailed t-test). In our experiments, the extrinsically motivated approach either fails in these challenging environments or learns a high-performing, but narrow, behavior that is unable to adapt. In the future this approach could allow more complex scenarios to be solved, where it is infeasible to learn from extrinsic rewards without manual reward shaping and curriculum learning.

VIII. Acknowledgements

We thank OpenAI for publishing accessible implementations of A2C and GitHub user p-kar for the integration of A2C to VizDoom[3]. We would also like to show our gratitude to the members of the Game Innovation Lab at New York University Tandon School of Engineering for their feedback and inspiring ideas.

References

[1] S. Alvernaz and J. Togelius. Autoencoder-augmented neuroevolution for visual doom playing. In *Computational Intelligence and Games (CIG), 2017 IEEE Conference on*, pages 1–8. IEEE, 2017.

[2] M. Andrychowicz, D. Crow, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. P. Abbeel, and W. Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5055–5065, 2017.

[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*, pages 1471–1479, 2016.

[4] Y. Bengio, J. Louradour, R. Collobert, and J. Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 41–48. ACM, 2009.

[5] D. E. Berlyne. Conflict, arousal, and curiosity. 1960.

[6] S. Bhatti, A. Desmaison, O. Miksik, N. Nardelli, N. Siddharth, and P. H. Torr. Playing doom with slam-augmented deep reinforcement learning. *arXiv preprint arXiv:1612.00380*, 2016.

[7] M. M. Botvinick, Y. Niv, and A. C. Barto. Hierarchically organized behavior and its neural foundations: a reinforcement learning perspective. *Cognition*, 113(3):262–280, 2009.

[8] D. S. Chaplot and G. Lample. Arnold: An autonomous agent to play fps games. In *AAAI*, pages 5085–5086, 2017.

[9] Y. Chebotar, K. Hausman, M. Zhang, G. Sukhatme, S. Schaal, and S. Levine. Combining model-based and model-free updates for trajectory-centric reinforcement learning. *arXiv preprint arXiv:1703.03078*, 2017.

[3] https://github.com/p-kar/a2c-acktr-vizdoom

[10] A. Dosovitskiy and V. Koltun. Learning to act by predicting the future. *arXiv preprint arXiv:1611.01779*, 2016.

[11] L. Festinger. *A Theory of Cognitive Dissonance*. Stanford University Press, 1957. ISBN 9780804709118.

[12] S. Gu, T. Lillicrap, I. Sutskever, and S. Levine. Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838, 2016.

[13] J. M. HUNT. Intrinsic motivation and its role in psychological development. *Nebraska symposium on motivation*, 13:189–282, 1965. URL https://ci.nii.ac.jp/naid/20001159493/en/.

[14] N. Justesen, P. Bontrager, J. Togelius, and S. Risi. Deep learning for video game playing. *arXiv preprint arXiv:1708.07902*, 2017.

[15] F. Kaplan and P.-Y. Oudeyer. Intrinsically motivated machines. In *50 years of artificial intelligence*, pages 303–314. Springer, 2007.

[16] M. Kempka, M. Wydmuch, G. Runc, J. Toczek, and W. Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *Computational Intelligence and Games (CIG), 2016 IEEE Conference on*, pages 1–8. IEEE, 2016.

[17] T. D. Kulkarni, K. Narasimhan, A. Saeedi, and J. Tenenbaum. Hierarchical deep reinforcement learning: Integrating temporal abstraction and intrinsic motivation. In *Advances in neural information processing systems*, pages 3675–3683, 2016.

[18] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman. Deep successor reinforcement learning. *arXiv preprint arXiv:1606.02396*, 2016.

[19] G. Lample and D. S. Chaplot. Playing fps games with deep reinforcement learning. In *AAAI*, pages 2140–2146, 2017.

[20] A. D. Laud. Theory and application of reward shaping in reinforcement learning. Technical report, 2004.

[21] J. Lehman and K. O. Stanley. Abandoning objectives: Evolution through the search for novelty alone. *Evolutionary computation*, 19(2):189–223, 2011.

[22] M. Lopes, T. Lang, M. Toussaint, and P.-Y. Oudeyer. Exploration in model-based reinforcement learning by empirically estimating learning progress. In *Advances in Neural Information Processing Systems*, pages 206–214, 2012.

[23] P. Mirowski, R. Pascanu, F. Viola, H. Soyer, A. J. Ballard, A. Banino, M. Denil, R. Goroshin, L. Sifre, K. Kavukcuoglu, et al. Learning to navigate in complex environments. *arXiv preprint arXiv:1611.03673*, 2016.

[24] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529, 2015.

[25] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning*, pages 1928–1937, 2016.

[26] A. Y. Ng. *Shaping and policy search in reinforcement learning*. PhD thesis, University of California, Berkeley, 2003.

[27] P.-Y. Oudeyer. Computational theories of curiosity-driven learning. *arXiv preprint arXiv:1802.10546*, 2018.

[28] P.-Y. Oudeyer and F. Kaplan. What is intrinsic motivation? a typology of computational approaches. *Frontiers in Neurorobotics*, 1:6, 2009.

[29] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. Curiosity-driven exploration by self-supervised prediction. In *International Conference on Machine Learning (ICML)*, volume 2017, 2017.

[30] R. M. Ryan and E. L. Deci. Intrinsic and extrinsic motivations: Classic definitions and new directions. *Contemporary educational psychology*, 25(1):54–67, 2000.

[31] T. Salimans, J. Ho, X. Chen, and I. Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[32] J. Schmidhuber. Formal theory of creativity, fun, and intrinsic motivation (1990–2010). *IEEE Transactions on Autonomous Mental Development*, 2(3):230–247, 2010.

[33] J. Schmidhuber. Powerplay: Training an increasingly general problem solver by continually searching for the simplest still unsolvable problem. *Frontiers in psychology*, 4:313, 2013.

[34] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[35] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

[36] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser, et al. Starcraft ii: a new challenge for reinforcement learning. *arXiv preprint arXiv:1708.04782*, 2017.

[37] J. X. Wang, Z. Kurth-Nelson, D. Tirumala, H. Soyer, J. Z. Leibo, R. Munos, C. Blundell, D. Kumaran, and M. Botvinick. Learning to reinforcement learn. *arXiv preprint arXiv:1611.05763*, 2016.

[38] Y. Wu and Y. Tian. Training agent for first-person shooter game with actor-critic curriculum learning. 2016.

[39] Y. Wu, E. Mansimov, R. B. Grosse, S. Liao, and J. Ba. Scalable trust-region method for deep reinforcement learning using kronecker-factored approximation. In *Advances in neural information processing systems*, pages 5285–5294, 2017.