Exploring the Effect of Different Behavior Characterizations in Atari Games

Justin Nguyen
jhn545
2014justinnguyen@gmail.com

Suhaib Abdulquddos
sa39663
suhaib.abdulquddos@gmail.com

***Abstract*— The issue with using gradient-based approaches in the Atari game environment is that some games have deceptive gradient landscapes and sparse reward functions that often trap the agent in local minima. Although novelty search has been known to mitigate such deception, not much research has been done to apply novelty search toward the Atari game environment. In particular, little to no research has been done on the effect of different behavior characterizations (BCs) in Atari games. Thus, the main contribution of this paper is to study the effects of using custom, handcoded BCs for two different Atari games: Frostbite and Kangaroo. Additionally, we also study the impact of using similarity metrics other than Euclidean distance to compare BCs, such as normalized compression distance (NCD). As expected, our experiments showed that using BCs that are more aligned with performance goals results in better performance than using a generic, unaligned RAM state BC. This has led us to believe that many elements of the RAM state BC are extraneous and do not aid in the search of behavior space. Furthermore, we found that using NCD with a run length encoding for the compression scheme resulted in significantly worse performance. Finally, we also found that novelty search alone was not enough to drive performance given computing constraints and should be combined with some form of performance (quality) metric to achieve good results in the Atari game domain.**

I. INTRODUCTION

Atari games are a popular research domain for reinforcement learning (RL) since they provide a structured, modular learning environment [Mnih et al., 2013]. However, most reinforcement learning approaches use gradient-based approaches to maximize a reward function. The issue with using typical gradient-based approaches is that some games have deceptive gradient landscapes, sparse reward functions, and other local optima that can deceive agents and severely impede performance.

One of the most prominent methods of mitigating this deception is novelty search [Lehman and Stanley, 2011a]. Novelty search works by rewarding agents for discovering behaviors that are different from previous ones regardless of how well they perform. In order to determine the difference between behaviors, one must define a distance metric (e.g. Euclidean distance) and a "behavior characterization" (BC). Behavior characterizations are essentially task-specific behavior representations. They are analogous to state representations, except that they represent behaviors instead of states. For example, if an agent is trying to escape a maze and each behavior is a traversal through the maze for X timesteps, then the BC could simply be the ending location of the agent at the final timestep. Ultimately, by rewarding novelty instead of performance, novelty search encourages the agent to try new things instead of being driven toward performance and, in some cases, consequently being deceived into a local optima.

Novelty search has been shown to work well in certain domains with very deceptive gradient landscapes such as human locomotion and the maze environment in [Lehman and Stanley, 2011a]. Additionally, [Conti et al., 2018] have shown that a modified novelty search algorithm combined with evolutionary strategies can perform well in certain Atari games such as Frostbite when using the RAM state[1] of the game as the behavior characterization.

Despite existing research on novelty search, not much work has been tested using the Atari game environment. Specifically, to the authors' knowledge, it is not known how novelty search performs in Atari games when using custom, hand-coded BCs for each game. Instead of simply using the RAM state as the BC for each game, this paper will tackle the challenge of developing task-specific BCs for the high-dimensional Atari game environment.

We hypothesize that the RAM state BC used in [Conti et al., 2018] contains extraneous information that is unhelpful with respect to discovering novel behaviors. We believe that using custom, hand-coded BCs for each game will result in higher performance and better direct the search for novelty.

Additionally, we hypothesize that using similarity metrics other than Euclidean distance, such as normalized compression distance (NCD), when comparing BCs can improve performance as shown in [Gomez, 2009]. Gomez's work used the Tartarus problem, which has a relatively easy-to-define behavior space, but we propose to extend this work to the more complex Atari domain.

We tested our hypotheses by building off of the codebase[2] used in [Conti et al., 2018]. However, due to time and computational constraints, we were only able to test our hypotheses on two games: Frostbite and Kangaroo. We

[1]Each game's RAM state is simply a vector of 128 bytes that, albeit describes the state of the game, is uninterpretable to human eyes

[2]https://github.com/uber-research/deep-neuroevolution

performed experiments for three different behavior characterizations as well as a separate experiment to compare NCD versus Euclidean distance.

Overall, our results did not prove to be better than pure evolutionary strategies. However, as expected, our experiments showed that using BCs that are more aligned with performance goals, such as the path taken BC in Frostbite and the heatmap BC in Kangaroo, results in better performance than using a generic, unaligned RAM state BC. This has led us to believe that many elements of the RAM state BC are extraneous and do not aid in the search of behavior space. Furthermore, we found that using NCD with a run length encoding for the compression scheme resulted in significantly worse performance compared to the original code using Euclidean distance, though more experiments need to be done. Finally, we also found that, unlike in simpler domains such as the classic maze example and human locomotion [Lehman and Stanley, 2011a], novelty search alone was not enough to drive performance given computing constraints and should be combined with some form of performance (quality) metric to achieve good results in the Atari game domain. We believe that our results are promising and will help future work on exploring the effect of BCs and novelty search in the Atari game domain.

II. RELATED WORK

In this section, we discuss the works that are related to and inspired our research, including the works of [Conti et al., 2018] and [Gomez, 2009].

Similar to our research, [Conti et al., 2018] have tried to apply novelty search to the Atari game environment. In particular, they combined novelty search with evolutionary strategies (ES) to form new quality diversity (QD) algorithms that not only search for performance, but also novelty. Their results showed that these new QD algorithms (NS-ES, NSR-ES, and NSRA-ES) were able to outperform classical RL algorithms such as DQN and A3C on certain Atari games. Their results showed that the success of these QD algorithms is not ubiquitous across the entire suite of Atari games, rather that some games are more conducive to QD algorithms than others. To expand on their work, our research focuses on using more appropriate (domain-specific) behavior characterizations to enhance the efficiency of the behavior space search and ultimately result in agents with greater overall fitness.

Research by [Gomez, 2009] has shown that alternative similarity metrics, such as normalized compression distance (NCD), can perform better than Euclidean distance. Furthermore, [Liwei Wang et al., 2005] have shown that Euclidean distance is a poor choice for comparing images and that different similarity metrics that take into account the spatial relationship of the pixels perform much better. As a result, we decided to research the effect of using NCD to compare behavior characterizations and compare the results to those of Euclidean distance.

III. OUR APPROACH

In this section, we discuss our approach to designing behavior characterizations for two Atari games (Frostbite and Kangaroo) as well as our approach to implementing the NCD similarity metric.

First, we will give an overview of how each game works so that the reader can understand the intuition behind our BCs. Then, we will discuss each of the following ideas in turn.

- Three different BCs for Frostbite: pixel counting, heatmap, and traversed path
- One BC for Kangaroo: heatmap
- NCD vs. Euclidean distance for Frostbite

A. *Frostbite: Game Overview*

The goal of the game Frostbite is to build an igloo for warmth before time runs out and the player freezes to death. To build the igloo, the player must acquire ice from the ice blocks that are floating horizontally across the screen as shown in Figure 1. Each time the player jumps onto a piece of ice, the entire row changes from white to blue and the player is awarded an ice block for their igloo. Once all of the ice block rows have turned blue, they will turn white again and the player must repeat the ice collection process until the igloo is complete. Additionally, there are enemies such as crabs, birds, and polar bears that the player must avoid or else lose one of his/her 3 lives.

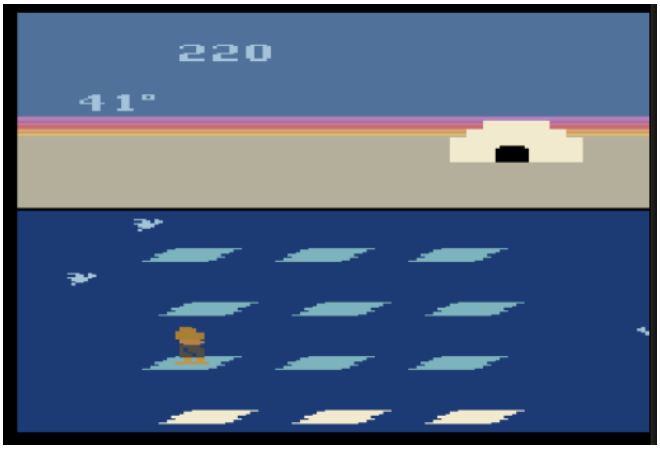

Fig. 1: A simple view of the Frostbite game. Note that the ice blocks turn from white to blue after stepping on them and that the blocks move from left to right. Also note the two birds on the left that must be avoided. Finally, note the igloo that has been built for shelter

We hypothesized that Frostbite would be a good candidate for novelty search since the reward function is rather sparse and the game has plenty of local minima where the player can get trapped.

For example, jumping onto an ice block rewards the player with ten points, but this is almost nothing compared to the few thousands of points rewarded for building and then entering the igloo. Furthermore, in order to actually build

the igloo, the player has to already be pretty good at the game, making it hard for agents that receive almost no initial rewards to learn a good behavior in classical reinforcement learning. This intuition is backed by [Conti et al., 2018]'s results in Figure 2. The sparsity of rewards makes Frostbite a reasonable candidate for testing novelty search.

Game	ES	NS-ES	NSR-ES	NSRA-ES	DQN	NoisyDQN	A3C+
Alien	3283.8	1124.5	2186.2	**4846.4**	2404	2403	1848.3
Amidar	322.2	134.7	255.8	305.0	924	**1610**	964.7
Bank Heist	140.0	50.0	130.0	152.9	455	**1068**	991.9
Beam Rider†	871.7	805.5	876.9	906.4	10564	**20793**	5992.1
Freeway†	31.1	22.8	32.3	**32.9**	31	32	27.3
Frostbite†	367.4	250.0	2978.6	**3785.4**	1000	753	506.6
Gravitar	1129.4	527.5	732.9	**1140.9**	366	447	246.02
Montezuma	0.0	0.0	0.0	0.0	2	3	**142.5**
Ms. Pacman	4498.0	2252.2	3495.2	**5171.0**	2674	2722	2380.6
Q*Bert†	1075.0	1234.1	1400.0	1350.0	11241	15545	**15804.7**
Seaquest†	960.0	1044.5	2329.7	960.0	**4163**	2282	2274.1
Zaxxon	**9885.0**	1761.9	6723.3	7303.3	4806	6920	7956.1

Fig. 2: Comparing novelty search and RL performance in Frostbite. The novelty search algorithms combined with reward (NSR-ES and NSRA-ES) perform much better than traditional RL algorithms such as DQN and A3C+. See [Conti et al., 2018] for further details as these results were pulled from their paper

Furthermore, the agent cannot simply follow the gradient and just jump onto the nearest ice block to get a small reward each time since doing so can result in the player getting stranded on an ice block in the middle of the ocean as shown in Figure 3. Following the gradient as such will obviously lead to local minima and result in suboptimal behavior. Instead, the agent also has to consider whether there is a path back to shore after jumping down. The need to avoid deception makes Frostbite a reasonable candidate for testing novelty search.

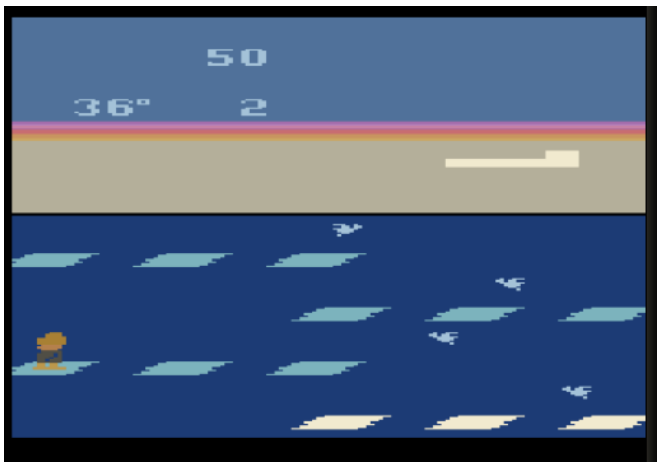

Fig. 3: Getting trapped by local minima in Frostbite. Here, we can see that the agent followed the reward gradient and jumped onto the 3rd row of ice blocks to receive 10 measly points each time. However, the agent is now trapped since there are no ice blocks above or below it. As a result, the agent can only jump up into the ocean, down into the ocean, or move right to fall into the ocean

B. Frostbite: Pixel Counting BC

The first BC we designed for Frostbite simply consisted of a single integer: the number of light blue ice pixels on the screen for each episode. The agent needs to step on ice blocks and turn them from white to light blue in order to make progress toward building the igloo. Thus, by making the BC the number of light blue ice pixels on the screen, we hoped to encourage the agent to find new behaviors that would result in collecting ice for the igloo.

We hypothesized that storing a single integer BC could be superior to the 128 byte RAM state BC. This is because the majority of the RAM state is likely to be useless in terms of discovering new behaviors. For example, one of the bytes might simply be the number of gray pixels on the shore (note that the RAM state is uninterpretable and we could not find documentation on what each byte value represented so we do not know for sure what is in the RAM state). Additionally, we knew that having a smaller behavior space would help focus the search and hopefully result in faster discovery of relevant, novel behaviors.

C. Frostbite: Heatmap BC

Next, we made a heatmap of the locations visited by the agent during each episode and used that as the BC. After every observation, we ran a filter on the image looking for the agent's RBG colors to find its position and then incremented that location in the heatmap by a constant value.

For example, in Figure 4 you can tell that the agent spent some time in the shore area where it was spawned (see the first row of non-black pixels). Then it seemed to jump to the first row of ice blocks toward the right (see the second row of non-black pixels). In this episode, it seemed to spend more time jumping down to the right of the screen. You can also tell that it very rarely made it to the 3rd row of ice blocks and never seemed to make it to the 4th row for this particular episode.

We hypothesized that this BC would perform better than the RAM state BC because this BC keeps track of where the agent has been and encourages it to visit other parts of the screen. This information is very relevant to solving the game. Eventually, if the agent has explored all parts of the screen, it should visit each of the ice block rows and collect enough ice to build the igloo.

D. Frostbite: Traversed Path BC

The last BC we tried for Frostbite consisted of a vector of (x,y) coordinates that represented the path taken during an episode. The difference between this BC and the heatmap BC is that the heatmap simply records where the agent traversed but not how it actually arrived at those locations. For example, in the heatmap in Figure 4, it is impossible to exactly tell which path the agent took. Did the agent go right, down, down, or did the agent go right, left, right, left, right, down, down or something else? The idea behind this BC was to encode the sequence of locations visited and encourage the agent to try new path trajectories.

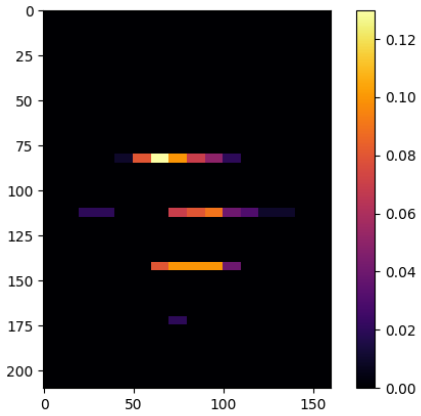

Fig. 4: An example Heatmap BC. You can see that the agent spends more time jumping down the right side of the screen and does not even make it to the 4th row of ice blocks this run

We hypothesized that this BC would perform better than the RAM state BC because this BC keeps track of the paths taken by the agent and encourages it to try new paths. This information is very relevant to solving the game. Eventually, if the agent has tried all the paths, it should visit each of the ice block rows and collect enough ice to build the igloo. Additionally, this BC is much smaller than a heatmap and should help direct the search better.

E. Kangaroo: Game Overview

In Kangaroo, the goal for each level is to rescue your kangaroo child at the top of the screen without being hit by enemy projectiles from both above and to the side. The agent needs to move left and right and climb the ladders to the top of the screen while also jumping and ducking to avoid enemy projectiles.

We hypothesized that Kangaroo would be a good candidate for novelty search since the reward function is rather sparse and the game has plenty of local minima where the player can get trapped.

For example, collecting fruit rewards the player with 100 points, but this is almost nothing compared to the 2000 points received for reaching the top and rescuing the kangaroo's child. In order to actually see this large reward, however, the agent has to be very good at the game. It has to know when to jump to avoid enemy projectiles from the side, when to duck because sometimes the side projectiles are thrown at your head instead of your feet, when to dodge left or right when projectiles or coming from above, and when to punch the enemies out of the way.

Furthermore, the agent cannot simply follow the gradient and go up the ladders as quickly as possible. This is because enemies are often waiting to throw the projectiles at you as soon as you are going up the ladder as shown in Figure 5. In this scenario, you are completely vulnerable to their attack since there is no way to dodge once you are on the ladder. Instead, the agent needs to bait the enemy's attack before trying to go up the ladder. The need to avoid such deceptive local minima makes Kangaroo a reasonable candidate for testing novelty search.

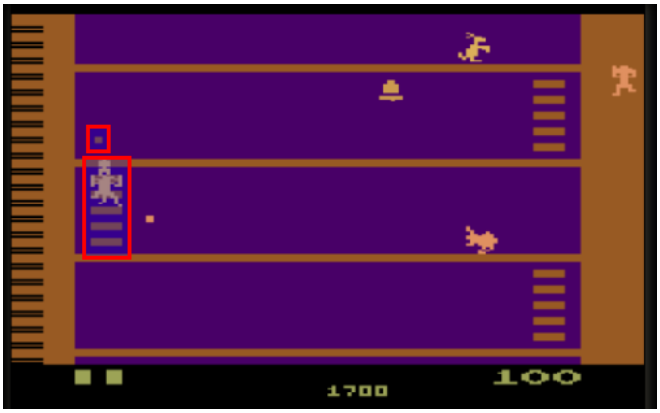

Fig. 5: Getting trapped by local minima in Kangaroo. Here, we can see that the agent followed the gradient to start climbing up the ladder. However, the agent is now trapped since it is on the ladder and cannot dodge the enemy projectile. Note that the top red box was drawn to show the projectile and the bottom red box was drawn to show the agent climbing the ladder.

F. Kangaroo: Heatmap BC

Unlike the simple maze problem explored in [Lehman and Stanley, 2011a], the amount of time that the agent spends at each location in the map is important. Specific locations may be dangerous at certain times in the episode, whereas at other times those same locations might be crucial stepping stones for reaching the goal.

We hypothesize that using a frequency heatmap BC to keep track of the number of frames spent at each 2D location in the map will encourage greater exploration of the map while also encouraging the agent to spend varying amounts of time in different areas.

As in Frostbite, the agent's location is computed by applying a filter over the image and looking for specific RGB values. For the heatmap BC, the screen is discretized into a 4x7 grid and each region of the grid has a value corresponding to the amount of time spent in that location. A visual aid of the heatmap is shown in Figure 6.

G. New Similarity Metrics: NCD

Normalized compression distance is a similarity metric that has proven to be quite effective at comparing discrete pieces of data represented as bytes [Cilibrasi and Vitanyi, 2005]. Given the prior success achieved in [Gomez, 2009] for the Tartarus problem, a natural extension is to apply this NCD metric to the results of [Conti et al., 2018] and compare RAM state BCs using NCD instead of Euclidean distance.

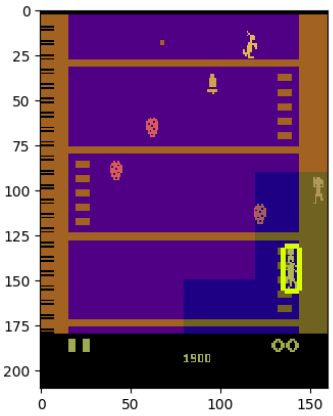

Fig. 6: Kangaroo heatmap. The darker regions of the game (purple regions) correspond to locations where the agent spent the most time.

NCD is defined as

$$NCD(x,y) = \frac{C(xy) - min(C(x), C(y))}{max(C(x), C(y))} \quad (1)$$

where x and y represent the pieces of data being compared, xy represents the union of x and y, and $C(a)$ is the length of a after being compressed by a compression function, where the compression function used is up to the user [Gomez, 2009].

The compression function used in the present work is called a run length encoding (RLE) [Pountain, 1987]. The idea behind run length encoding is that consecutive repeated elements of a feature vector can be combined into two elements: the first element represents the frequency and the second element represents the identity of the element being repeated. When appending two different RAM states together, we chose to interleave the RAM states such that the i'th element of the first feature vector is next to the i'th element of the second feature vector. Thus, if the two corresponding elements of each vector are the same, the RLE score will reward this similarity and yield a better (shorter) encoding length of the appended feature vector.

Furthermore, we provide an additional extension to RLE: a threshold value for similarity within the RLE routine. That is, we only create a new frequency-value pair when the next element is different from the preceding element by some threshold value Δ. For the purpose of actual encoding, this would result in lossy compressions, but for our purposes, this gives some leeway when comparing RAM states that are only slightly dissimilar. Whereas standard Euclidean distance penalizes all non-zero differences regardless of magnitude, with a sufficiently large threshold value, our RLE scheme will not penalize two feature vectors for being only slightly dissimilar. We believe this distinction to be important when comparing discrete RAM states.

IV. EXPERIMENTS AND RESULTS

In this section, we will show the results of each of the approaches mentioned in Section III. Below is a bulleted list of our overall results

- The handcrafted BCs aligned with performance perform better than the generic RAM state BC
- The pure novelty-based approaches (NS-ES) performed poorly
- The NCD metric performed poorly compared to the Euclidean distance metric

Additionally, here are some of the details of the experiments performed. We kept these parameters the same across each of the experiments for consistency. Each iteration consisted of 500 episodes and the mean episode reward is the reward averaged across each set of 500 episodes. We used 10 nearest neighbors in the distance calculation for novelty search. We used the Adam optimizer provided by Tensorflow with a step size of 0.01.

The NSR-ES and NS-ES algorithms are the same as those used in [Conti et al., 2018] and stand for "novelty search with rewards and evolutionary strategies" and "novelty search and evolutionary strategies", respectively. The important distinction between NSR-ES and NS-ES is that NSR-ES integrates reward into the calculation to drive the search of behavior space, whereas NS-ES solely uses novelty.

A. NSR-ES for Frostbite

Figure 7 compares the performance of different BCs in the game of Frostbite using the NSR-ES algorithm. It shows that only the path taken BC performed well and that the rest of the BCs performed poorly. We believe this is because the path taken BC was very aligned with performance (i.e. exploring all of the paths will eventually result in building the igloo and result in high rewards). Though the heatmap BC was also well aligned with performance, we believe that its much larger BC dimensionality prevented the agent from properly exploring the behavior space. Though not properly shown in Figure 7, the log files in our code repository[3] back this hypothesis by showing that each iteration of the heatmap BC takes about 6 times longer to compute than that of the path taken BC. We believe that discretizing the heatmap into bins, as in the heatmap approach for Kangaroo, would have resulted in better performance. We believe that the pixel count BC did not perform well because it is unable to capture the complexity of each behavior. Using a single integer to describe the behavior of an episode is intuitively unlikely to succeed. After all, there are only so many different light blue pixels that can appear on the screen. Once the agent has run through all of those possibilities, there is no more novelty to be discovered. As a result, the pixel counting BC eventually discouraged the agent from exploring more of behavior space. Despite the different results in [Conti et al., 2018], we believe that the RAM state BC also performed poorly because of the extraneous information that diluted the behavior space search.

[3]https://github.com/justin-nguyen-1996/deep-neuroevolution

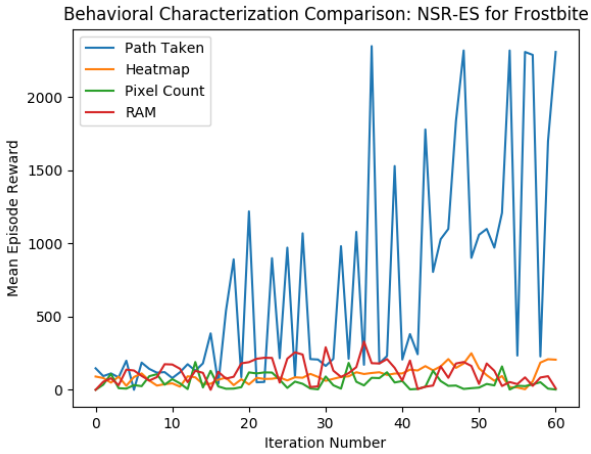

Fig. 7: Comparison of different BCs in Frostbite using NSR-ES. This figure shows that the path taken BC performed very well compared to the baseline, generic, RAM state BC as well as the other BCs. This is likely due to the path taken BC being well aligned with performance.

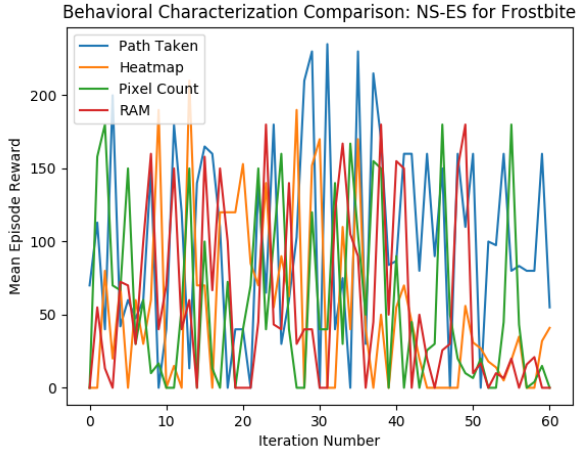

Fig. 8: Comparison of different BCs in Frostbite using NS-ES. This figure shows that all of the BCs perform poorly when only using novelty to drive the behavior space search. Note the difference in scale from Figure 7

B. NS-ES for Frostbite

Figure 8 compares the performance of different BCs in the game of Frostbite using the NS-ES algorithm. The results show that each of the different BCs performed rather poorly, even the path taken BC that was successful with NSR-ES. This seems to suggest that novelty alone is not enough to traverse behavior space, given a comparable amount of computational resources.

C. NSR-ES vs. NS-ES for Frostbite

Figure 9 compares the performance of NSR-ES and NS-ES for the path taken BC in Frostbite. Again, this plot seems to suggest that combining rewards with novelty (quality with diversity) is necessary to perform well in higher dimensional domains like Atari games.

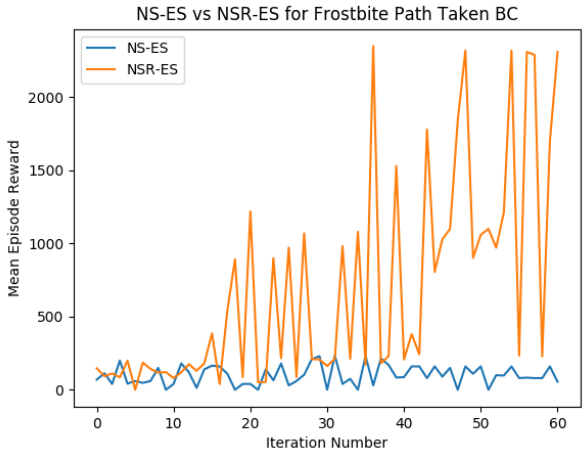

Fig. 9: Comparison of NSR-ES and NS-ES for the path taken BC in Frostbite. This figure suggests that only using novelty to drive the behavior space search does not work on high dimensional spaces such as Atari games.

D. NSR-ES for Kangaroo

In figure 10 we observe a similar trend which shows that our handcoded BC performs better than the RAM state BC for the same number of iterations. It should be noted that the reason behind there being more data for the heatmap BC is simply that we ran out of compute time when generating results for the RAM state trial. However, we can still analyze differences in the training curves until around 44 iterations, during which time data for both experiments is available. One major advantage of the heatmap approach is that the BC is a fixed length 28 element vector, and does not depend on the number of timesteps that the agent survives per episode. We believe this much smaller BC contributed to the superior performance observed in the heatmap experiment.

E. NCD vs. Euclidean Distance

We observe in Figure 11 that NCD provides greater consistency in the mean reward achieved per episode. Although euclidean distance achieves a greater maximum mean episode reward than NCD over 50 episodes, we note that this comes at the cost of inconsistency. We believe the consistency of NCD can be attributed to the more lenient approach that NCD has with regards to comparing two behavior characterizations. As mentioned in the similarity metrics section of this paper, NCD only looks at the differences in compression length between two BC's. This means that two BC's are only considered different if they have different underlying structures. For our purposes this means that for a given pair of BC's a and b, NCD may have considered these BC's to be more similar than euclidean distance would have. This may have resulted in increased difficulty with regards to making progress with respect to discovering more novel behaviors. Unlike euclidean distance, NCD is not

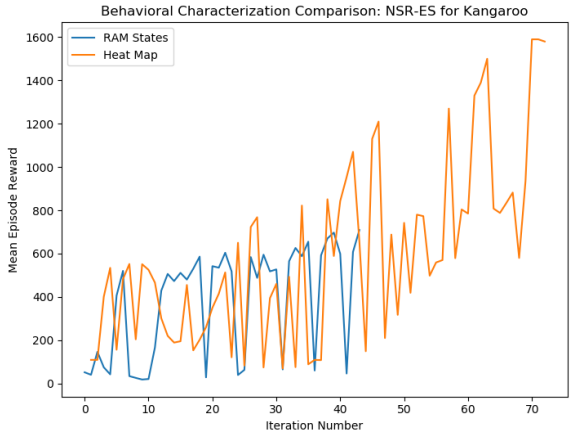

Fig. 10: Comparison of different BCs in Kangaroo using NSR-ES. This figure shows that the heatmap BC performs much better than the generic RAM state BC

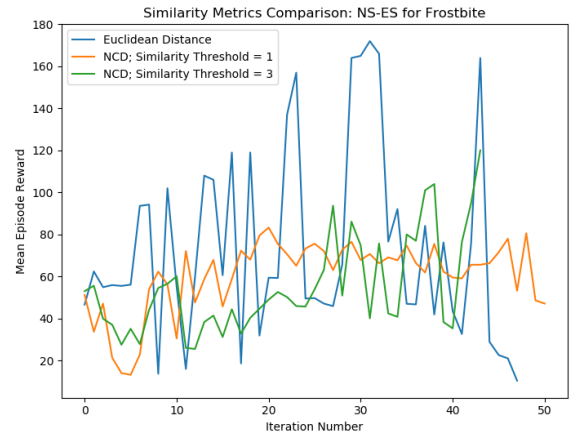

Fig. 11: Comparison of different NCD similarity thresholds in Frostbite using NS-ES. This figure shows that NCD performs similarly, but still poorly, compared to Euclidean distance and that the threshold does not impact performance that much

a differentiable function, therefore following a gradient is much more difficult. We see in Figure 12 that NCD used with the RLE compression function results in significantly worse performance than euclidean distance. It should be noted however, that NCD could still yield better results with a different compression function. Due to compute limitations we were ultimately unable to test compression functions other than RLE with NCD. This remains an avenue for future work.

V. DISCUSSION

In this section, we discuss some of the issues we faced along the way and the limitations of our results.

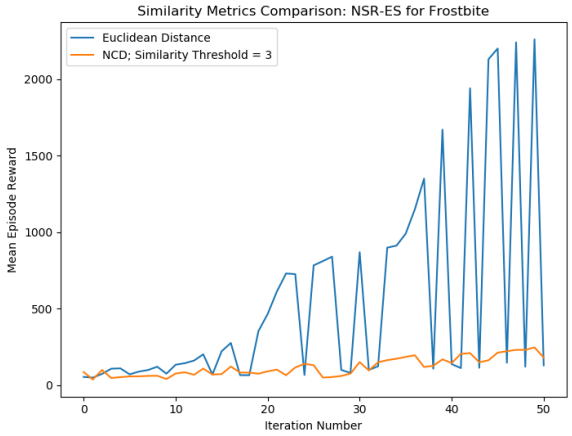

Fig. 12: Comparison of NCD vs. Euclidean distance in Frostbite using NSR-ES. This figure shows that NCD performs much worse than the Euclidean distance metric

A. *Irreproducibility of [Conti et al., 2018]'s Results*

Though we started our project from [Conti et al., 2018]'s codebase, we were never able to reproduce the results reported in their paper. For example in Frostbite, their results reported an average score of 367.4 for ES, but we achieved an average score of about 1800 using their same code. Contrastingly, they were able to achieve a score of 2978.6 using NSR-ES with their RAM state BC while we only got an average score of about 200. Note that we checked the experiments and codebase and tested with other games as well only to find similar differences.

One of the main differences between our results and theirs is the amount of computational power available to them. Though [Conti et al., 2018] were able to train their models for 2 billion frames in about 2 hours, we estimate that it would take approximately 12 days to achieve a similar amount of training time on our machines (which was infeasible for obvious reasons).

Though we did try to replicate their results by training the models for over 12 hours, due to the difference in training time, we cannot conclusively say that their code works or not.

B. *Problems With Results*

As mentioned in the previous subsection, not being able to trust the underlying starter code in [Conti et al., 2018] means that it is hard to trust our own results. Despite our BC results being better than the RAM state BCs in our own experiments, the results were not better than those reported in [Conti et al., 2018]'s paper. As a result, it is hard to draw a definitive conclusion.

Ultimately, a lack of computing resources were a major reason we were unable to troubleshoot the differences in results. Similarly, such lack of resources is also the reason we were unable to test our approaches on more than two Atari games. Though we did end up using external resources such

as Google Cloud Platform in addition to our own personal devices, it would have been helpful to have more computing resources.

Despite the discrepancy in results, we do believe that our experiments were valid and will be able to help future research regarding novelty search and behavior characterizations in the Atari game domain.

VI. FUTURE WORK

In addition to the work already done in this paper on researching the effects of custom, handcoded BCs for different Atari games, there is much future work that can be done. This section will discuss the following potential extensions to our research:

- Tuning novelty search parameters
- Different methods of combining novelty and fitness
- Combining multiple BCs
- Implementing actual quality diversity algorithms
- Driving search for novelty using human input
- Learning BCs instead of handcoding them

First of all, there are several parameters of novelty search that can be tuned for better performance as shown in [Gomes et al., 2015]. These parameters include the size of the archive, the number of nearest neighbors used, and varying the underlying evolution algorithm used (e.g. genetic algorithms, NEAT, etc.).

Secondly, one could try different methods of combining novelty and fitness scores. For example, the results of [Conti et al., 2018] combined novelty and fitness by simply weighting each score by 50% for their NSR-ES implementation. For their NSRA-ES implementation, they dynamically weighted the two based on the stagnation of performance. In their implementation, they start with a 100% weighting for performance until performance has not increased in the past X iterations. At this point, they decrease the performance weighting and increase the novelty weighting until performance starts to go back up again and then reset the performance weighting back to 100%. Instead of these arbitrary weighting schemes, one can try using a multi-objective optimization approach with pareto-based ranking such as NSGA-II [Deb et al., 2002].

Thirdly, one can try a different approach to BCs. Instead of painstakingly altering the underlying BC to make it more aligned with performance, one can accept the fact that BC are likely to be unaligned anyway and just try to drive the search using multiple combined BCs. The results of [Pugh et al., 2016] have shown that combining multiple BCs can improve robustness against deception while still maintaining diversity provided that one BC targets diversity and the other BC drives the search.

Fourthly, the implementations in [Conti et al., 2018] simply hack together novelty and performance to approximate quality diversity through weighted averages. It would be very interesting to see how well-known quality diversity (QD) algorithms [Pugh et al., 2015] such as novelty search with local competition (NSLC) [Lehman and Stanley, 2011b] and multi-dimensional archive of phenotypic elites (MAP-Elites) [Mouret and Clune, 2015] perform instead.

Fifthly, one could try using human input to direct the search for novelty instead of fully automating it. The results of [Woolley and Stanley, 2014] are similar to typical interactive evolutionary computation (IEC), where the user helps direct the evolution process by choosing from an on-screen population of behaviors, except that the user can now decide to choose to produce 'novel' descendants. Their work showed that human input can help novelty search find good solutions significantly faster than purely automated searches.

Finally, one can even try learning the behavior characterizations instead of handcoding them. One approach is by using autoencoders to automatically learn which state representations actually matter (though of course they can also learn things that do not matter or even worse things disregard important features). Or, one could try the approach used in [Meyerson et al., 2016]. This approach uses stochastic policy induction (SPIRIT) to learn new BCs from generic BCs. The results were shown to work well on the classic maze domain, but it would be interesting to see how this approach fares on the Atari game domain.

VII. CONCLUSION

Despite the popularity of the Atari game domain in the reinforcement learning community and an abundance of games with deceptive gradient landscapes, not much work has been done in applying novelty search to this domain. Our work helps expand the state of the art by researching the effects of different behavior characterizations in the Atari game domain.

Per our experiments, the results obtained in Figures [7 and 10] support our initial hypothesis that using generic BCs, such as each game's RAM state, results in suboptimal performance. We believe that this is because generic BCs contain additional data that dilute the behavior space traversal. Our BCs that were aligned with performance (such as the path taken BC in Frostbite and the heatmap BC in Kangaroo) were able to perform better than the generic RAM state BC, at least in our own experiments. Additionally, our results in Figure 9 showed that novelty alone is not enough to result in good performance in the Atari game environment, given limited computational resources.

We also experimented with using NCD instead of Euclidean distance as the similarity metric for comparing BCs. Ultimately, this approach yielded results that were significantly worse than those obtained from using Euclidean distance. However, more experiments need to be done to test other compression schemes with NCD before making further conclusions.

Although the data collected for this paper is rather limited due to time and computational constraints, we believe that the experiments were valid and provide a decent starting point for future exploration of behavior characterizations in the Atari game environment.

REFERENCES

[Cilibrasi and Vitanyi, 2005] Cilibrasi, R. and Vitanyi, P. M. (2005). Clustering by compression. *IEEE Trans. Inf. Theor.*, 51(4):1523–1545.

[Conti et al., 2018] Conti, E., Madhavan, V., Such, F. P., Lehman, J., Stanley, K. O., and Clune, J. (2018). Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents. In *Proceedings of the 32Nd International Conference on Neural Information Processing Systems*, NIPS'18, pages 5032–5043, USA. Curran Associates Inc.

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197.

[Gomes et al., 2015] Gomes, J., Mariano, P., and Christensen, A. L. (2015). Devising effective novelty search algorithms: A comprehensive empirical study. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 943–950. ACM.

[Gomez, 2009] Gomez, F. J. (2009). Sustaining diversity using behavioral information distance. In *Proceedings of the 11th Annual Conference on Genetic and Evolutionary Computation*, GECCO '09, pages 113–120, New York, NY, USA. ACM.

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

[Lehman and Stanley, 2011b] Lehman, J. and Stanley, K. O. (2011b). Evolving a diversity of virtual creatures through novelty search and local competition. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, pages 211–218, New York, NY, USA. ACM.

[Liwei Wang et al., 2005] Liwei Wang, Yan Zhang, and Jufu Feng (2005). On the euclidean distance of images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(8):1334–1339.

[Meyerson et al., 2016] Meyerson, E., Lehman, J., and Miikkulainen, R. (2016). Learning behavior characterizations for novelty search. In *Proceedings of the Genetic and Evolutionary Computation Conference 2016*, pages 149–156. ACM.

[Mnih et al., 2013] Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. In *NIPS Deep Learning Workshop*.

[Mouret and Clune, 2015] Mouret, J.-B. and Clune, J. (2015). Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*.

[Pountain, 1987] Pountain, D. (1987). Run-length encoding. *Byte*, 12(6):317–319.

[Pugh et al., 2016] Pugh, J. K., Soros, L. B., and Stanley, K. O. (2016). Searching for quality diversity when diversity is unaligned with quality. In *International Conference on Parallel Problem Solving from Nature*, pages 880–889. Springer.

[Pugh et al., 2015] Pugh, J. K., Soros, L. B., Szerlip, P. A., and Stanley, K. O. (2015). Confronting the challenge of quality diversity. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, pages 967–974. ACM.

[Woolley and Stanley, 2014] Woolley, B. G. and Stanley, K. O. (2014). A novel human-computer collaboration: combining novelty search with interactive evolution. In *Proceedings of the 2014 annual conference on genetic and evolutionary computation*, pages 233–240. ACM.