Evolving Robust Robots

Connor Keane

January 24, 2018

Abstract

In real world applications, AI controlled robots must safely and effectively operate within unexpected circumstances — including damage to robot systems. Powerful algorithms have been developed to allow robots to estimate the uncertainty in the control predictions made by their control networks, detect physical damage, and quickly learn new behaviors. In none of these approaches, however, have the underlying algorithms and structures been evaluated. In this experiment, we investigated the intrinsic robustness of NEAT evolved networks, where robustness is defined as the capability of the network to function when placed in situations beyond the training of the network. The four evolutionary conditions considered were Feed Forward and Recurrent networks evolved with and without noise in simulation. These networks were then evaluated in simulation on fully functional robots, robots missing a light sensor, robots missing half their sonar sensors, robots missing the entire left sensory field, robots with random noise added to their output, and robots with random noise added to their input. There were significant differences in performance between networks evolved in different conditions for each evaluation condition. In general, recurrent networks outperformed or were equivalent to feed forward networks — with the exception of the robot with one damaged light. What type of network is best for a given situation then depends on the expected conditions; choosing to design for normal operating conditions or catastrophic failure lead to different solutions. Generally, the fact that significant differences were observed suggests that considering robustness independently from task performance when designing robot control systems could improve performance outcomes in either case. We also found that adding noise to simulation either did not affect or improved outcomes, so designing for the reality gap problem and for robustness are not at cross purposes in this domain.

1 Introduction

Some challenges within adaptive robotics, such as the reality gap problem [4], where simulated performance doesnt translate to the real world, can be framed as design problems. These design problems involve choice rather than invention; rather than using novel approaches, design problems, as we consider them, are solved by selecting existing approaches based on a set of known and unknown benefits and risks. Common design challenges also include those relating to maximizing training performance given the constraints of available data. Different available data leads us to consider different approaches, such as supervised or unsupervised learning, that also have implications for other elements of the designed system. While it is possible to develop a novel system to solve a problem, these new approaches become possible solutions that may be selected when designing a robot for a particular task. Framed this way, the role of robotics research is to maximize the known information about different approaches to facilitate educated design choices.

The problem of maximizing performance during error states is an analogous domain. Rather than a gap that emerges from training, like the reality gap, this is a harsh-reality gap. In real applications, we desire continued functioning of the robot even when it encounters novel environments or is functionally damaged.

Table Making educated choices in this domain requires research into how control systems are impacted by and respond to damage to either their central or peripheral components. In this project, we chose to study the response of NEAT evolved control networks to damage to external systems [8]. Neglecting the engineering challenges that go into choosing sensors and focusing solely on the capabilities of control systems within these error states, we seek an answer to the question: are there evolutionary conditions that maximize robustness without retraining?

2 Background

Our chosen domain, neuroevolution of evolving topologies (NEAT), evolves neural networks from a population of simple starting networks. This starting population is evaluated on a task, after which some of the worst performing networks are dropped from the population and new networks are created by crossing elements of the best performing networks. Elements that may be crossed between networks are nodes, connections, or weights, all controlled by a genomic coding within NEAT. Each of these evaluations and repopulations is one generation. As more generations are evaluated, the networks gradually increase in complexity as crossed networks gain elements and random mutations add both new nodes and new connections. This method has proven capable of creating efficient representations of difficult problem spaces [8]. In general, NEAT is useful for domains in which a general, learned representation of a goal is desired and the size of the network needed to create this representation is unknown. Thus, NEAT and its progeny are relevant to robotics problems where adaptation to a complex environment is desired, as these complex changes are represented by the structures evolved by NEAT. At the same time, the ways in which these networks work is somewhat inscrutable, as the evolutionary process gives no insight into how networks encode features of the environment. By studying robustness, we seek to understand one dimension of this representation and how it might impact physical robots.

A survey of semi-autonomous robots deployed in real world in search and rescue and military operations, such as the world trade center response, found that robots required operator intervention approximately every 2 minutes [2]. This presents a serious barrier to extended autonomous operation and compromises the effectiveness of such response efforts. Failures were distributed over several categories, the foremost among these being control system, effector, and sensor system failures, with the first two categories making up over half of all recorded failures [2]. Though these categories are fairly non-specific — and this field has likely improved since 2005 — the failures due to sensor and effector damage form the focus of the following discussion and experiment.

The dominant approaches to solving this problem and improving autonomy have been active learning approaches. Since the states we are studying occur after training, these algorithms have explicit self calibration and the ability to re-explore the behavior space under new conditions. Intelligent Trial and Error is a prominent algorithm that uses Bayesian guessing to re-explore the space of possible behaviors, converging to a new optimal behavior in around 2 minutes [3]. This is much more efficient than retraining with traditional methods, such as reinforcement learning, that could also be used to re learn behaviors. This learning can also be done without simulating error states in advance or gathering additional data with which to train the robot. Still, this method is

both time and computation intensive.

Other effective approaches have given robots the ability to create a explicit proprioceptive models by self experimentation to find correlations between actions and sensor states [1]. Independent experimentation allow the robot to build an internal model of its body plan and simulate the dynamics of this body. By updating this model as conditions change, such as by damage to legs of the robot, the robot is able to use internal simulation to converge to new optimal behaviors. Again, this method is highly computationally expensive and relies upon the identification of an error state by the robot.

Some systems also use hand designed control systems to solve this same problem. A relatively recent and highly visible example of this approach is BigDog, a robot built by Boston Dynamics for use in military applications. In several convincing videos, BigDog is able to keep its balance when subjected to difficult conditions. This robust behavior is due, in large part, to walking algorithms that were written and tested by teams of engineers [6]. Other similar methods often use function compositions with Gaussian or Bayesian methods for improving performance and ensuring behaviors are viable. Such systems are very effective, but restrict our use of methods, such as deep learning and genetic algorithms, that can't be directly verified.

Methods built around neural networks and genetic algorithms are an increasingly active area of research and, in our reading, little attention has been paid to the performance of these methods without direct training or design for error compensation. The first mention we find of the robustness of deep learning methods is from a June 2017 paper [7] that uses uncertainty prediction methods to choose when to switch to more conservative behaviors. This study proposed approaches for overcoming the limitations of deep learning methods but didnt look at factors that influence the robustness of these methods in isolation.

Uncertainty estimation methods have seen dramatic advancements in the last few years and have started to incorporate data similarity into estimates of uncertainty arising from mismatch between training and real data [9] [5]. This is an active area of research and the connections between prediction confidence and performance are not yet well understood. Comparisons between equivalent network architectures on the grounds of robustness is also an open question. Generally, our ability to predict the capabilities of a network decrease as we provide inputs farther beyond the training set [7]. In particular, the behavior of uncertainty estimation methods has not been thoroughly investigated for input far outside the training set.

In addition to implications for robot effectiveness, robustness has implications for AI safety. A robot that fails to act as expected or as trained is, definitionally, in an error state. Minimizing these error states, understanding the behaviors that arise from them, and designing systems that fail gracefully can all be considered aspects of robustness. Active approaches may prevent or resolve most failures, but implicit robustness passively maintains behavior within tolerable bounds. We dont expect implicit calibration, as we might term the ability for a system to continue functioning without explicit representation of error, to match the abilities of any of the above active approaches in most domains. In limited cases, robust systems could prevent the need for active approaches or allow the robot to function until it is able to apply more complex methods to learn new behaviors. Robustness is also a factor in the design of these more complex systems, as these higher level algorithms are expected to produce meaningful solutions even when encountering novel situations. By treating the problem of robustness similarly to the reality gap problem, we show that simple changes may produce significant changes in adaptability that may continue to be significant even for robots that employ more complex planning and adaptation algorithms.

3 Methods

There isnt an established procedure for researching the robustness of control systems in this way, as stated by researchers at MIT [7]. It is clear that deep learning and genetic algorithms have limits to their performance in novel situations. What these are and how to best research them is still an open question. In this experiment, we tested networks evolved over fifty generations with NEAT in several different challenging environments. Four different evolutionary conditions were tested across each of these environments. All four evolutionary runs used one of two sets of NEAT parameters; we evolved networks with only feed forward connections and with the possibility of recurrent connections. Taking inspiration from reality gap research [4], we repeated these two evolutions with 10% noise in the simulator. Aside from the listed parameters, all conditions were held constant and left at default values (Table 1).

Table 1: NEAT parameters

Input nodes	9	Connection probability	.1
Output nodes	2	Add node probability	.05
Activation	tanh	Mutate bias	.2
Hidden nodes	0	Bias mutation power	.5
Population size	50	Mutate weight probability	.9
Species size	10	Weight mutation power	1.5
Survival threshold	.2	Toggle link probability	.01
Elitism	1		

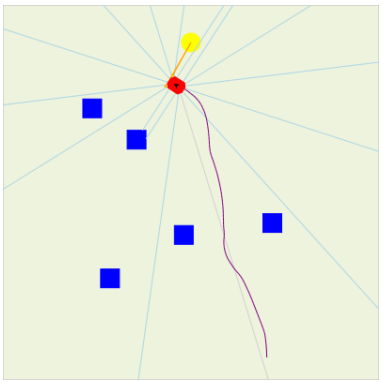

Figure 1: Example of a random world and the behavior of a recurrent network evolved with noise

The evolved networks were tested on a pioneer robot in the jyro simulator. Each individual was assessed on its ability to find the light in a 10m x 10m environment with five randomly placed square blocks .5m x .5m between the robot and the light (Figure 1). A light was placed in the top center of the environment and the robot was started from the bottom of the environment in a random position along the bottom of the environment in the center third. The fitness of an individual was the normalized sum of light readings averaged across five different random environments. In each run, the robot was given 200 time steps to act in the environment; if the robot stalled for more than 5 consecutive time steps, the trial was cut off. Each generation consisted of 50 individuals

and each evolution ran for 50 generations. In each generation, the best individual was saved for later analysis.

Once the evolution of all fifty generation completed, the saved best individuals were reassessed with the same fitness function as during evolution and the top five individuals of this test selected as representative of the evolutionary run. These conditions were: random environment with no noise, no data from left light sensor, no data from left sonars, no data from left light sensor or left sonars, 10% noise added to all motor output and 10% noise added to all sensory input. The experimental conditions are intended to represent real world error states, such as broken, detached, or blocked sensors, unpredictable sensor error, and damage to drivetrain components. The robot was run through each of these conditions in the same environment before changing environments for a total of twenty iterations in each condition. The overall fitness for each condition was the average over the twenty runs, with fitness for a single run again being the normalized sum of light readings over 200 time steps in the jyro simulator. The presented results are taken from twenty NEAT evolutions with five individuals pulled from each, for a total of 2,000 individual tests in our sample for each experimental condition.

This large sample of tests and large number of individual trials used in the assessment of each individual was chosen to overcome the randomness inherent in NEAT. Additional randomness was introduced by the use of random environments, so several runs were also used to ensure that the fitness of the robot reflected its ability to navigate and not the difficulty of the world it was placed in. Preliminary experiments using a smaller number of evaluations produced robots far more sensitive to the choice of environment and made trends — though present — much less clear. Even with improved fitness metrics, there is still substantial randomness from differences between NEAT runs.

4 Results

Table 2: Table of selected t values for two-tailed t test. The top row presents the t score between the top two networks; the bottom row presents the t score between the best and worst networks for a given condition. A t value greater than 1.983 implies statistical significance ($p < .05$).

All sensors	One light	Half sonars	No left Sensors	Noisy motors	Noisy sensors
1.413	0.0352	1.439	1.555	1.809	1.927
3.990	4.534	4.195	4.195	2.214	2.514

In these experiments, we expected recurrent networks and networks evolved with noise would, overall, show more robust behavior. Additionally, the robustness of a network would be independent of its performance on the task in the control condition.

This last assumption did not hold true, which complicates the clarity of the subsequent results. As we see in the graph of absolute performance in Figure 2, the recurrent networks generally outperformed the feed forward network, with the feed forward network evolved with noise performing worse than any other network ($p < .054$ relative to recurrent with noise, two-tailed t test). While comparison on the basis of absolute behavior is still useful, we also present fractional loss in average fitness, so that each network is compared against its own baseline. In order to determine the significance of the results, we have plotted a 95% confidence interval on the plot of absolute performance and performed two-tailed t tests between selected pairs of results. In these measurements,

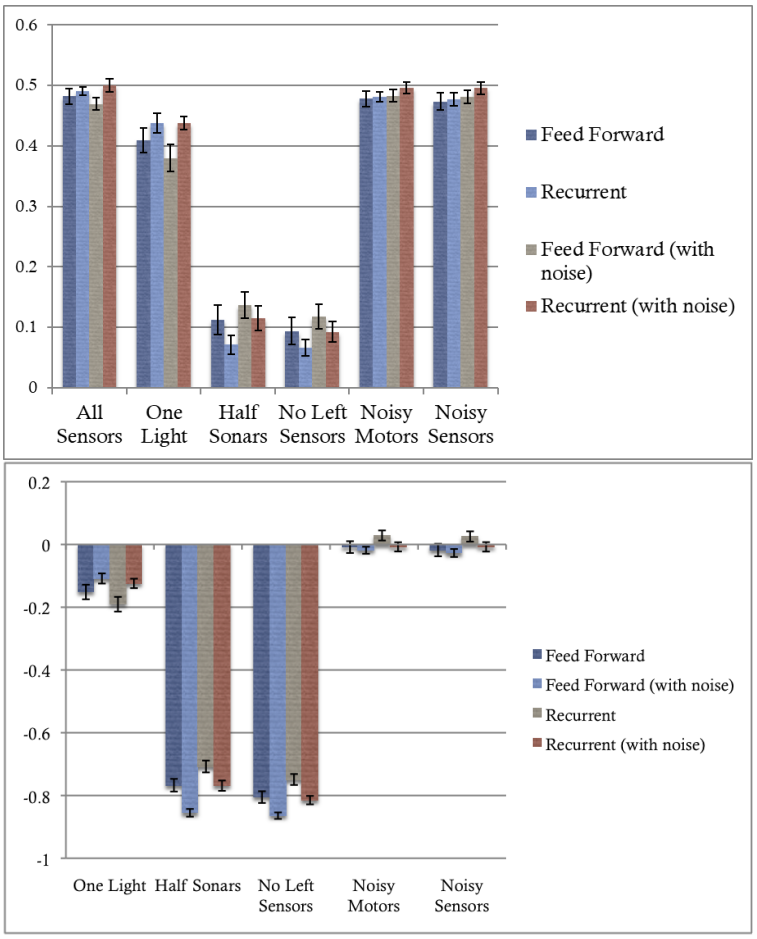

Figure 2: At top, the average fitness of each evolutionary approach plotted with a 95% confidence interval on the mean. At bottom, the fractional change in performance for each behavior relative to the control condition. Error bars on this plot indicate one standard deviation.

we consider the evaluation of a single individual one data point and our data as having 100 degrees of freedom as a result.

The evolved robots were consistently able to reach the light within 200 time steps, taking paths such as that shown in Figure 1. Recurrent networks outperformed all other other networks in all test conditions but those lacking sonars. For the control and one light conditions, the best recurrent network was significantly better than the best feed forward network ($p < .05$, two-tailed t test). In the remaining two cases, the difference in performance is not statistically significant, by a narrow margin ($p = .07$). In the noisy cases and cases lacking sonars, the evolutionary run with noise performed significantly better that the evolutionary run without noise. In all cases, the best performing network was significantly better than the worst performing network ($p < .05$).

Possibly the most interesting type of network was the feed forward network evolved with noise. Notably, this was the only network that improved its performance relative to the control condition. This network performed the worst in the control and one light cases, but performed the best in the two cases lacking sonars. This difference was not significant compared to the next best case

($p = .07$) when we consider our data as having 100 degrees of freedom. Though this does not meet our criteria for significance, the change in ordering from worst to best is notable and the failure to reach statistical significance is by a narrow margin. The difference is easily significant in comparison to the recurrent network evolved without noise, which showed a precipitous drop in performance on the conditions without sonars.

When we look at the graph of fractional difference in performance, this drop can be clearly seen. On the cases without sonars, the best and worst performing networks showed the least and greatest fractional loss in performance, respectively. We have defined fractional difference as the difference between the performances of the control and experimental networks divided by the performance of the control network, and calculated our error as the maximum and minimum values within one standard deviation of the relevant quantities. In all cases, the absolute performance and percent loss in performance were directly correlated, which is significant considering the difference in absolute performance in the control condition — the best networks for each condition were also the most robust networks within that condition. We reserve the feed forward network evolved with noise from this last statement, as it alone gained performance when subjected to a noisy environment; such a result doesn't readily fit within our understanding of robustness.

5 Discussion

Our results do not clearly support or reject our hypothesis across all conditions. When there was sonar damage, the feed forward networks performed more strongly, indicating not all damage is equal. Our assumption that each of the evolutionary methods would perform equivalently on this task was also not true of our results; in the noisy conditions this was closer to true, but the control condition showed clear differences between recurrent and feed forward networks. This potentially means that less robust networks could outperform more robust networks when we consider robustness solely as the difference in performance of a network in an error condition relative to the performance of that network in the control condition. This was not the case in our results, and the dramatic loss in performance by the recurrent network and unexpected performance of the feed forward network evolved with noise suggests that the different performance on the control condition does not significantly influence our ability to evaluate robustness.

This particular result, and this particular network, is the most surprising. The gain in performance relative to the control condition on the tasks with noise suggests that the network evolved to exploit noise, causing it to perform poorly in the relatively simple control and one light conditions. The fact that this drop in performance was caused by the addition of noise to the evolution is concerning, because it implies that the effect of adding noise is more complex than merely improving robustness. While this is likely unimportant when considering something like the reality gap problem, since we expect the robot to operate in noisy conditions, it may mean that noise in simulation creates unexpected features in the robots behavior. For our purposes, these unexpected features led to more robust behavior in the most difficult conditions. While the difference was not statistically significant by the $p < .05$ standard, it is clear from the graph that this network — as well as others — has a unique response to damage. This is supported by the correlation between similar conditions.

The conditions in which all networks performed most poorly were the conditions in which the networks lacked sonar sensors. While there is some variation between the tests lacking just sonars and the tests lacking both sonars and a light sensor, the error is dominated by the lack of sonars.

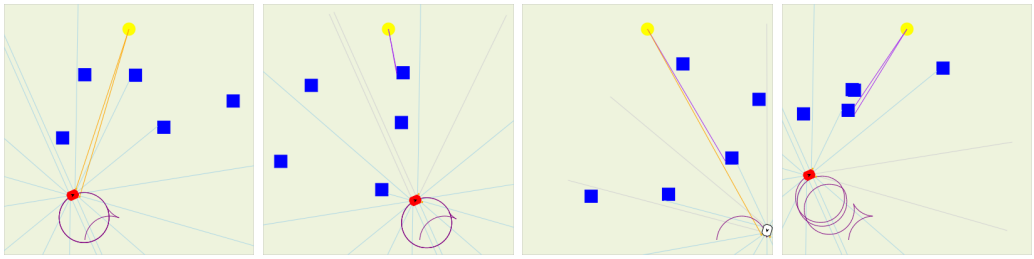

Figure 3: Sample behaviors of NEAT evolved networks. From top left, a feed forward network evolved with noise evaluated with no left sensors. A feed forward network evolved with noise evaluated with no left sonars. Here, we see that lacking sonars dominates the behavior of the robot the left sensor field. At top right is a recurrent network evaluated without the left sensor field. At bottom is a recurrent network evolved in noisy conditions evaluated with no left sensors.

This is borne out not just in the similarity of the quantitative data, but in the qualitative behavior of the robots — as exhibited in Figure 3. Crucially, when we look at the tests with and without sonars as separate groups, we see that the relative performance of all of the networks stays relatively constant within a group. For the conditions without sonars, this effectively doubles the number of tests for each network. To the extent that the tested networks are reflective of the NEAT method as a whole, the correspondence of the results of these two test conditions increases our confidence that the observed robustness of the feed forward with noise method is due to a true difference between these approaches by increasing our confidence that we have truly measured the fitness of each individual in our sample and represented that evolutionary method as well as we can within reasonable statistical bounds.

Within these same groups, another consistent trend can be seen in the recurrent network, this one passing into statistical significance. Despite our hypothesis and despite strong performance on the easier tasks, this network suffered the greatest loss in performance. Considering that adding recurrence is a reasonable second approach to solving problems in which feed forward networks fail or more complex representations are justified, this dramatic drop off in performance paints this choice as having consequences for the ability of such an approach to succeed when faced with a larger range of situations. The addition of noise to the simulator improved the outcomes for the recurrent networks, so mitigating this cost may be simple; further investigation is needed to determine if recurrent networks show similar features as feed forward networks did in this experiment.

Any statement as to the causes of the patterns in our data do not reach far beyond speculation. The general better performance of recurrent networks on the easier tasks is not exceptionally surprising, since recurrent networks can encode additional features compared to feed forward networks. This may mean that the recurrent network relies more heavily on representations that integrate multiple sensors and so failures in part of the system more heavily impact other aspects. This effect, if indeed this explanation approaches the real cause, only involved sonars and was disrupted by the addition of noise in simulation. The difference in outcome between damage to sonars and damage to light sensors may also be due to the different levels of the task each type of sensor contributes to. The light sensors allow gross planning of movement through the environment but provide little information about obstacles. Since stalling on obstacles cuts short trials, sonars are more immediately significant and are not as heavily implicated over the longer arc of a trial. Recurrence may hamper the ability of the network to react to obstacles as they are presented in the

damaged condition. Aside from this, the meaning of the damage to the different sensory systems was not equivalent. A value of 0 from a sonar sensor indicates an obstacle immediately blocking the sensor while a value of 0 from a light sensor indicates that the light is indeterminately far away. While both are valid forms of damage and the value of the result is not strictly dependent on the meaning of the particular type of damage as much as on consistency between evaluations for each network, comparing negative and positive error states such as these may obscure deeper insight into the causes of the different behavior.

Another cause we considered was the complexity of networks; perhaps the best performing networks were simply more complex and these results are merely a measure of complexity — though this does not easily account for our results in the conditions lacking sonars. To the first order, there was no difference in complexity between the different runs and breaking down analysis by network complexity rather than network type did not show any clear correlation. Across two evolutionary runs, the top five networks from each run showed an average of 3 hidden nodes and 26 connections across all evolutionary conditions. In this average, there was a spread of networks with as few as 0 as many as 7 hidden nodes, with no clear systematic differences in performance between networks with more or fewer nodes.

As discussed previously in relation to the effects on behavior caused by damaged sonars, qualitative features of the behaviors produced by different error conditions show substantial difference that appear small when judged solely by the fitness score. The most striking of these is from the no sonars condition. As we see in Figure 3, the standard recurrent network very immediately crashes into the wall. The best performer in this condition, however, circles endlessly. In terms of fitness, neither behavior comes near to accomplishing the task. From a potential safety perspective, however, there is a world of difference between an immediate crash and a stable failure pattern. We can also see in this example that the less robust approaches often displayed this stable behavior. In all of the networks we observed, the feed forward with noise condition did not show a catastrophic failure like that shown in the example, leading us to conclude that the difference in fitness in the robots lacking sonars seems to reflect the frequency of catastrophic failures rather than an absolute difference in behavior. If we were to use this network on a robot equipped with a higher level retraining algorithm, like those presented in our review of the literature, a greater probability of graceful failure could minimize further damage to the system and allow this more powerful method more opportunities to intervene.

Choosing a network structure based on these results does not guarantee that any individual network will be more robust or hold a particular error state; there is substantial local variance between networks as a result of randomness inherent in the evolutionary process. This local variance is roughly equivalent across all categories, as shown by the confidence intervals in Figure 2, so the overall trends describe tendencies for each evolutionary method. It is unclear whether more complex domains will increase the disparity between approaches or level them, but the results of the most complex conditions in this experiment lead us to think that some approaches will suffer dramatically.

6 Conclusion

Evolving feed forward networks with noise may be the best for some domains where extensive damage is likely, particularly given the greater probability of graceful failure exemplified in Figure 3. The deficit in performance by this method on the easiest conditions is an interesting result but

is also an artifact of the study — networks in real situations consistently encounter noise. Further experiments could examine the impact of missing sensors within a noisy environment, more closely simulating reality and allowing a stronger statement of this conclusion.

When such significant damage is not expected, recurrent networks evolved in our conditions show both better overall performance and better performance with moderate damage. In more dangerous situations or situations in which there is potential risk in AI failure, a feed forward networks may be called for. More generally, the type and extent of damage expected within our problem domain direct our choices. More details of what types of damage imply what approaches we leave to further work. It is clear that such research is warranted for robots seeking the highest levels of autonomy. Subtle differences in our choice of approach make substantial differences in the robustness of our evolved networks. These differences, for several of the networks, are differences between otherwise interchangeable approaches — at least within this domain. Across dissimilar approaches, these differences may become more pronounced. Were further research to show this to be the case, robustness becomes an increasingly important foundational choice. This choice serves as a simple opportunity to improve outcomes without substantial effort.

In a more philosophical sense, we are interested in expanding this work to investigate the robust properties of biologically motivated systems — in particular developmental approaches. Given their strength in learning representations and actions in order of increasing complexity, these approaches may have an advantage in robustness over traditional training. This project took a small start in this direction by looking at evolved networks rather than more directly designed approaches; comparison to classical machine learning approaches seems the most direct next step to this more philosophical question.

References

[1] Josh Bongard et al. Resilient machines through continuous self-modeling. *Science*, 2006.

[2] J. Carlson and RR Murphy. How ugvs physically fail in the field. *IEEE*, 2005.

[3] Antoine Cully et al. Robots that can adapt like animals. *Nature*, 2015.

[4] Oraziov Miglino et al. Evolving mobile robots in simulated and real environments. *MIT Press Journals*, 1995.

[5] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In D. D. Lee, M. Sugiyama, U. V. Luxburg, I. Guyon, and R. Garnett, editors, *Advances in Neural Information Processing Systems 29*, pages 4026–4034. Curran Associates, Inc., 2016.

[6] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, and Rob Playter. Bigdog, the rough-terrain quadruped robot. *IFAC Proceedings Volumes*, 41(2):10822 – 10825, 2008. 17th IFAC World Congress.

[7] C. Richter and N. Roy. Safe visual navigation via deep learning and novelty detection. *Proceedings of Robotics Science and Systems*, 2017.

[8] Kenneth Stanley. Competitive coevolution through evolutionary complexification. *Journal of Artificial Intelligence Research*, 21, 2004.

[9] Lingxue Zhu and Nikolay Laptev. Engineering uncertainty estimation in neural networks for time series prediction at uber. *Uber Engineering Blog*, 2017.