PRESERVING KNOWLEDGE IN SIMULATED BEHAVIORAL ACTION LOOPS

by

Rachel St.Clair

A Dissertation Submitted to the Faculty of

Charles E. Schmidt College of Science

In Partial Fulfilment of the Requirements for the Degree of

Doctor of Philosophy

Florida Atlantic University

Boca Raton, FL

May 2022

PRESERVING KNOWLEDGE IN SIMULATED BEHAVIORAL ACTION LOOPS

by

Rachel St.Clair

This dissertation was prepared under the direction of the candidate's dissertation advisor, Dr. Elan Barenholtz and Dr. William Hahn, Center for Complex Systems and Brain Sciences, and has been approved by all members of the supervisory committee. It was submitted to the faculty of the Charles E. Schmidt College of Scienceand was accepted in partial fulfillment of the requirements for the degree of Doctor of Philosophy.

SUPERVISORY COMMITTEE:

Elan Barenholtz
Elan Barenholtz (Mar 29, 2022 11:10 EDT)

Elan Barenholtz, Ph.D.
Dissertation Co-Advisor

William Hahn (Apr 5, 2022 11:08 EDT)

William Hahn, Ph.D.
Dissertation Co-Advisor

Susan Schneider
Susan Schneider (Apr 5, 2022 11:37 EDT)

Susan Schneider, Ph.D.

Gary W Perry
Gary W Perry (Apr 5, 2022 11:53 EDT)

Gary Perry, Ph.D.
Chair, Center for Complex Systems and Brain Sciences

Teresa Wilcox

Teresa Wilcox, Ph.D.
Interim Dean, Charles E. Schmidt College of Science

Robert W. Stackman Jr., Ph.D.
Dean, Graduate College

April 8, 2022

Date

ACKNOWLEDGEMENTS

The author wishes to express sincere gratitude to her committee members for all of their guidance and support, and a very special thanks to my advisors for their persistence, patience, encouragement, countless pep-talks, and endless support during the preparation and typing of this manuscript. The author is grateful to the Rubin Gruber Sandbox for providing the research equipment and space to conduct the study. Last but not least, the author wishes to thank her lab mates for all of their support.

ABSTRACT

Author: Rachel St.Clair

Title: Preserving Knowledge in Simulated Behavior Action Loops

Institution: Florida Atlantic University

Dissertation Advisor: Dr. Elan Barenholtz and Dr. William Hahn

Degree: Doctor of Philosophy

Year: 2022

One basic goal of artificial learning systems is the ability to continually learn throughout that system's lifetime. Transitioning between tasks and re-deploying prior knowledge is thus a desired feature of artificial learning. However, in the deep-learning approaches, the problem of catastrophic forgetting of prior knowledge persists. As a field, we want to solve the catastrophic forgetting problem without requiring exponential computations or time, while demonstrating real-world relevance. This work proposes a novel model which uses an evolutionary algorithm similar to a meta-learning objective, that is fitted with a resource constraint metrics. Four reinforcement learning environments are considered with the shared concept of depth although the collection of environments is multi-modal. This system shows preservation of some knowledge in sequential task learning and protection of catastrophic forgetting in deep neural networks.

DEDICATION

This manuscript is dedicated to my family, particularly my supportful and patient mother, Kimberly, and father, Joe, who have encouraged me through these many years of research, and especially to my cat, Boopi, who is the joy of my life. I also dedicate this work to my friends who have allowed me to allot an extraordinary amount of my cognitive processing time to this work by offloading some basic human functioning to their own accord, and my grandparents, Phyllis and Don, both of whom believed in the pursuit of my dreams unequivocally.

PRESERVING KNOWLEDGE IN SIMULATED BEHAVIORAL ACTION LOOPS

LIST OF TABLES

LIST OF FIGURES

Chapter 1

INTRODUCTION

One basic goal of artificial learning systems is the ability to continually learn throughout that system's lifetime. The ability for continual learning, or life-long learning implies that the artificial learning system, or agent, can learn without forgetting prior learning. The potential utility of learning without forgetting, or 'knowledge preservation', is twofold: first, the system can return to previously learned tasks after learning something new. Secondly, bootstrapping previous knowledge may allow for faster learning of a novel task. Both of these potential benefits of preserving learned information may serve to conserve resources. Allowing the same computational hardware to be 'multiplexed' for another task, rather than requiring allocation of distinct resources, reduces the required energy resources involved in the learning process. Multiplexed information is stored in such a way that the system can switch between previously learned tasks while retaining performance. Information can then be accessed for learning new tasks in a contextually relevant manner. Multiplexed information is thus stored and retrieved in a way that facilitates the conservation and use of prior learning, broadening the ability of a learning system to multiple task domains by encoding high-level 'multi-use' concepts.

Although many metrics exist for characterizing desirable properties of artificial general intelligence (AGI), the term is commonly used to describe a software program that is highly adaptable to a variety of environments while displaying a range of functionality similar to, or beyond, human cognition. The term is distinct from traditional artificial intelligence research, which aims to make domain-specific prediction programs. AGI, or rather superintelligent AGI, which has the potential to

drastically improve the quality of human life, being able to solve problems at computational speeds that far exceed the capacity of several human lives combined and with perspectives on systemic societal issues currently unavailable to human bias. However, it is not without ethical consideration that the creation of such a program could be adversarial to human life if the sueprintelligent AGI does not hold the same values as it's biological counterparts [1]. Nevertheless, several approaches for AGI have been created and play an important role in understanding artificial learning and cognition.

Both approaches, AGI and AI, to creating artificial learning systems encompass a wide variety of approaches, all attempting to create an intelligence in which being capable of learning without forgetting to is desirable to some degree. Transitioning between tasks and re-deploying prior knowledge is thus a desired feature of artificial learning. Each of these approaches has a unique perspective to offer upon our understanding of intelligence as a whole, as it relates to human, animal, and machine 'life'. However, no approach thus far has been able to demonstrate the degree of knowledge preservation necessary to achieve life-long learning to the degree of biological counterparts.

In recent years, practical implementations in the field of artificial intelligence (AI) have largely converged on deep learning methodologies. The underlying mechanics of the majority of these approaches involve artificial neural networks constructed with weights which are tuned by backpropagating an error signal in accordance with some update rule (i.e. stochastic gradient descent). While deep learning has shown state-of-the-art performance in many domains such as computer vision, natural language processing, time-series forecasting, etc., one major problem is that of catastrophic forgetting, in which new learning overwrites prior learning. Challenges in overcoming catastrophic forgetting are bound by practical considerations such as resource constraints and model applicability. As a field, we want to solve the catastrophic forget-

ting problem without requiring exponential computations or time, while demonstrating real-world relevance. Learning without catastrophic forgetting would potentially allow an AI model to transfer its knowledge base between tasks. The study of transfer learning is still in its infancy in neural network models. More research is needed on how to create robust algorithms that employ prior knowledge without requiring excessive training time and previous learning rehearsal. Thus, new learning without overwriting prior learning provides a model with a broadened intelligence, rather than the narrow, domain-specific intelligence seen in current AI approaches. I suspect a broad intelligence basis, when paired with other adequate algorithms, might allow for the emergence of previously elusive cognitive behaviors such as causal inference, common sense, rational choice, etc.

The field of AGI encompasses many techniques and metrics. Most notable in recent years are described in [2], which describes symbolic, emergentist, artificial life, developmental robotics, universalist, and hybrid approaches to AGI. In symbolic architectures, which have seen the majority of adoption in the AGI community, have the challenge of designing memory components capable of strong learning. Other approaches exhibit strong learning, such as associative memory capabilities, but lack the ability for high-order concepts, i.e. complex language and reasoning. While still, others lack the ability to model the brain to the degree of which larger artificial learning systems can be constructed, failing to result in algorithms which exhibit multiple cognitive behaviors. Hybrid architectures attempt to leverage the various symbolic and connectionist approaches but lack in their ability to be understandable in terms of design and offer little insight into human intelligence. That, these approaches are often a hodge-podge of design elements taken from computational experiment, and not lack systematic design based on properties of human intelligence as we understand it from the perspective of neuroscience. Interoperability for understanding the resulting artificial learning system is important as it protects against adverse interpretations of

reality as we understand it, compared with reality as a computer may understand it; thus, protecting ethical considerations and real-world applicability [3]. While many of these techniques have resulted in useful, intelligent algorithms, we have yet to see algorithms that result in knowledge preservation close to the level in which vertebrate brains can perform. Even though our understanding of neuroscience is still incomplete, core features from foundational mechanisms of action can be recruited to better design AGI systems. Thus, it is foreseeable that AGI approaches would benefit from better understanding how knowledge preservation occurs in biological systems.

It can be inferred from biological intelligence that one core property of how biological brains learn is their ability to preserve knowledge within resource constraints. That the number of neurons for an animal is mostly fixed and animals with advanced nervous systems can learn multiple distinct tasks without forgetting prior tasks. Evolution designed vertebrate brains in a highly conserved manner in over about 3.5 million years [4]. This information is stored molecularly, mostly by way of genetic encoding. The modern human brain sequesters this information during early prenatal development before any online learning starts to take place. Advantageous features of the brain to be repairable, easily constructible, conserve resources, modifiable, and synchronous favored the highly conserved architectural design of biological brains [5]. In vertebrates, these evolutionary pressures resulted in nervous system topology structures with distinct condition detection devices (neurons) arranged in segregated populations (brain areas) with ambiguous roles which coordinate with the larger specific tasks of their respective area. From such architecture, emergent fundamental features arise allowing for higher cognitive processes, according to the recommendation architecture theory of cognition [6].

When neuron populations are segregated based on functional tasks (eg. visual vs auditory streams), they have to share information via some third party mechanisms of integration to signal to the overall system the appropriate interpretation of the current

condition. This area to area signaling is not exactly information compression, it's more like communication or control and may be a foundational principle of emergent cognitive phenomena. It is plausible that this communication between brain areas lends itself to the self-generated, "subjective", experience. The modular paradigm found in the human brain uses several tools to manage information flows through the system, including (but not limited to) inhibition, feedback, behavior recommendation and selection, indirect activation, receptive field expansion, and reward cascades [7].

Evolution is a slow process of optimization for which we can only observe the final products and few intermediate states. By contrast, researchers can design and implement computational architectures speedily and flexibly, creating the potential for new models that predict and test hypotheses about the evolution of the nervous system. Currently, most artificial network architectures are arbitrarily designed due to lack of systematic evidence for what types of connectivity topology best facilitates what type of learning. Here, connectivity refers to how information units are linked to other information units, whether it be nodes in a neural network, subject and predicate links in graphs, etc.. Vertebrate brains are highly conserved in that they have continuously evolved without discarding much of the previously adapted architecture. If neuron connectivity lends itself to learning environmental solutions, then some previously learned solutions are already stored in the earlier evolved parts of our brains.

The niche field of neuroevolution tries to replicate what occured naturally by allowing network topology to evolve using various genetic algorithms given input. Most of the best neuroevolutionary models investigate emergent architectures with specific inputs for task optimization. To date, no published approach uses knowledge preservation as an optimization metric. Likewise, there is little information on how connectivity architectures emerge from an evolving environment subject to similar pressures in which the human brain developed.

Significance

The importance of investigating what types of architectures are well suited for preserving knowledge lies in constructing artificial learning systems that can not only provide utility in our daily lives, but also give insight into our understanding of human learning and intelligence.

Currently, topologies of artificial learning systems are somewhat haphazardly designed. Informing our architecture designs with systematic research, will lead to better insight into how models of artificial cognition can be better constructed. The key idea is modeled from the human brain where it is likely that network topology is at least, if not more, important than later learning algorithms in achieving knowledge preservation. This allows us to avoid ending up with a system that is doing the bulk of the learning process after it has been configured. In a system that prioritizes initial connectivity, this manifests as understanding what is the appropriate topology to facilitate continual learning, over the model's lifetime. Our current artificial learning algorithms drastically reduce their biological counterparts to imitative mathematics that do not yet withstand the demands of real-world problem solving within resource conservative systems.

The major problem of catastrophic forgetting prevents artificial neural networks from learning in a similar fashion to the human brain. An abundance of evidence suggests that biological brains are capable of learning without forgetting and that they do so within resource constraints. Brains do not create a new neuron for every piece of information they need to learn. Instead, they multiplex existing learning in novel contexts. If we better understand how connectivity between neurons facilitates knowledge preservation, we can not only create more generally intelligent artificial agents, but also learn more about how the human brain is facilitating learning.

In a broader sense, knowledge preservation lends itself to general learning which

is useful for designing artificial intelligences with less computational resources and in better understanding how the organizations of neurons in the human brain may facilitate different types of life-long learning. The significance of having an artificial general intelligence is that if AGI is created with benevolent,ethical consideration, it is speculated to be useful in solving a range of everyday tasks as well as tackling some of humanity's toughest challenges. We will have created something that is capable of solving problems, just as a human would but with much more speed and processing power. The reason for these exponential returns is that silicon chips are inherently much faster than biological neurons. As semiconductor research progresses, it is foreseeable that any model of cognition that runs in a computer will be much faster than any biological counterpart. The difficulties in creating a computational model that has the ability to help humanity solve our toughest challenges, eve those we cannot foresee, is that the model must be ethical (sharing humanitie' values), capable of general intelligence (that it can learn in a exceedingly comparable fashion to human intelligence), and that it can understand the world as we understand it, or rather, better than we understand it.

Theoretical Basis

Lifelong learning, or continual learning, is the idea that a computational model should learn across tasks without being reset to an initialization state. In lifelong learning, a model is trained upon inception on a task and retains it's training so long as it is in use [8]. This means that the model weights are not reset, as they typically are in current AI models, between training tasks. Knowledge preservation concerns itself with lifelong learning models. Those that re-initialize the weights are inherently destroying all previous learning. Being able to switch between tasks, using prior knowledge, is a deserada of general intelligence. Thus, lifelong learning models play a critical role in understanding the functions of intelligence in biological brains.

The question then becomes, how do we structure models to facilitate learning across a vast range of tasks, storing knowledge in such a way that it can return to previous tasks later and use prior learning to learn new tasks efficiently. Connectionism proposes that the brain is composed of units, or neurons, which act on other units to create a functional dynamical network of total units active at one time [9]. Activity denotes excitation or inhibition. This idea has been extensively extended to computer science where simulations of nodes activating other nodes within a network occur. Thus, neural networks rely extensively on the connectionist paradigm by incorporating connected nodes updated with backpropagation. In backpropagation, node weights are learned by propagating and error backwards through the hierarchy of connections [10]. In these approaches, knowledge is built by iterating over environmental input which is forced through hierarchical encodings [11]. Here, hierarchy enforces decomposition of higher concepts into parts, which relies on the combinatorial expression of the network to make sense of. The 'mind' results from the environmental input's impression on the neural substrate. Parallel processing enables the network to create a dynamical system which in turn, takes on new functions for the network itself, interacting with other networks as a complex system [12]. The main point of connectionism is that the connectome, or connections between neurons, construct the mind, or rather, cognition. The architecture of the brain results in learning which transfers input to output in a series of parallel computations of units activating other units in a network. In the field of artificial intelligence, recent research has only just begun to look at what types of architectures facilitate which types of learning. While many deep learning approaches already use inductive biases of biological brains, such as visual cortex hierarchies in the convolutional neural network, there is much that remains to be understood on the impact of architectures on inductive biases of neural networks. The topology of various networks considers how computational units, or neurons, are connected in such a way that facilitates

learning appropriate information and transcoding that information into appropriate output, or behavior. Connectionism in artificial neural networks measures learning by stimulus-response pairs, or rather how well the model can predict an output from an input. Artificial learning models, especially those using deep learning methods, suffer in the ability to learn new tasks without forgetting prior-learning. In the artificial intelligence community, this problem is known as catastrophic forgetting (CF), or the plasticity/stability trade-off. This is in part due to the fundamental nature of back-propagation in deep learning models to rewrite learning; where weights are updated most heavily towards current learning at the expense of weights learned in more distant tasks [13]. The result of backpropagation, even when assuaged by CF mitigation, is narrow feature learning that will eventually reach a resource constraint asymptote as learning increases throughout the model's lifetime. In recent years, practical implementations in the field of artificial intelligence (AI) have largely converged on deep learning methodologies. The underlying mechanics of these approaches involve artificial neural networks constructed with weights which are tuned by backpropagating an error signal in accordance with some update rule (i.e. stochastic gradient descent). While deep learning has shown state-of-the-art performance in many domains such as computer vision, natural language processing, time-series forecasting, etc., one major problem is that of catastrophic forgetting (CF), in which new learning overwrites prior learning. Challenges in overcoming CF are bound by practical considerations such as resource constraints and model applicability. As a field, we want to solve the catastrophic forgetting problem without requiring exponential computations or time, while demonstrating practical relevance. Learning without CF would potentially allow an AI model to transfer its knowledge base between tasks. Thus, new learning without overwriting prior learning provides a model with a broadened intelligence, rather than the narrow, domain-specific intelligence seen in current AI approaches. We suspect a broad intelligence basis, when paired with other adequate algorithms,

might allow for the emergence of previously elusive cognitive behaviors such as causal inference, common sense, rational choice, etc. Embodied cognition describes a theory of cognition in which the brain needs the ability to execute actions in an environment in a cyclical fashion. Behavioral outputs react with the environment, producing the next set of inputs, aiding in higher cognitive functions, such as learning and agency. In embodied cognition, representations of the mind are manipulated to interact with nature in a way that produces a dynamical system [14]. The being, or agent of the brain, is directly entwined with the interaction of it's situation. Physiology cannot be distinctly separated from goals of the agent, and thus, must be viewed as part of the overarching nature of cognition [15]. Embodied cognition describes a system in which an agent takes input from the environment, computes that information, and provides output which in turn creates it's next input. Reinforcement learning (RL), a subset of artificial intelligence and dynamic programming, models embodied cognition in the form of state, action, and reward scenarios. The core foundation of RL is that most behaviors are sequential, requiring active engagement in sequestering the next input to the sequence [16]. A balance between exploration and exploitation is critical in successful RL algorithms as the first results in new solutions and the former in reliance on previous learning. Therefore, retaining prior learning and being able to use it in new situations is imperative for RL architectures. The mathematical foundations of RL rely upon markov decision making [17]. Excessive trial-and-error is needed to iterate through effective policies for decision making before an optimal solution is converged upon. This results in the need for an abundance of computational resources; unlike biological brains which are bound by the number of neurons and biochemical processes available to the system.

The evolution of the human nervous system has been speculated to be a product of darwinian fitness. The central dogma of Darwinian evolution is that adaptations which are advantageous get passed down via genetic traits [18]. Here, adaptations

are behaviors encoded in genes and advantage, or fitness, refers to the organism's increased survival and reproduction. The class of computational algorithms, genetic algorithms (GA), capitalize on this fundamental feature; that a fitness function is used as a metric to determine which genes should propagate to future individuals in a population. GAs are mathematical constructs to embody natural evolution [19]. While most GAs use the strategy of transferring genes, or rather computational instructions, to the next generation, or rather iteration, of the population (i.e. program), according to a fitness function. Together, the fitness function and the mathematical code creates an optimization tasks, which with enough computational resources and time, is generally successful at deriving solvable solutions. This approach, although human derived, is quite similar to natural selection in Darwinian evolution since the program's behavior results in advancing the system forward, by recombination and mutation of genetic material that provides an increase in fitness. The fitness function thus becomes instrumental in advancing the program's outcome and constitutes the evolutionary pressures of development.

Resource constraints are considered as a key indicator of multiplexing information in life-long learning systems here based on the theoretical framework of the Recommendation Architecture (RA) as theorized in [6]. Understanding recommendation architecture (RA) requires thinking about the brain as a system in which different anatomical structures like the cortex, basal ganglia, hippocampus, thalamus, amygdala and cerebellum are subsystems that perform different, distinctive types of information processes, each contributing in key ways to the brain's conscious activity. This framework for understanding higher cognition describes brain function in a similar mechanism to understanding functions of a computer; by relating levels of understanding to levels of anatomical description. The key feature of RA is that it describes foundational properties of cognition in terms of core system interactions. This means that properties of the brain can be somewhat abstracted from

their biological properties of neurochemical interactions and dynamical systems into artificial systems which embody the system interactions by mechanisms outside of physiology and anatomy. Here, we rely on the RA's description of evolutionary pressure and constraints of biological brains to provide support for artificial cognition being constrained within resource conservative systems. This approach is very useful in understanding the problem of knowledge preservation. Thus, we can define the concept of knowledge preservation as the ability to learn information for later multiplexing under resource constraints. The constraint of limited resources enforces the multiplexing of information units. That is, in order to keep the number of neurons needed from growing exponentially, neurons need to be used in various ways without degrading prior learning. Once information is stored appropriately, it can then be bootstrapped to inform novel contexts. With this consideration, learning can then be measured, in part, in terms of using prior information in novel context and returning to previous context without information degradation, all within minimal requirements of the amount of computational resources available.

Relevant Literature

Recent literature suggests that the intersection of continual deep reinforcement learning with genetic algorithms may be useful for preserving knowledge. State of the art learning algorithms in reinforcement learning rely on deep-learning based Deep Q Networks (DQNs) and Markov decision based processes, namely Proximal Policy Optimization (PPO) and Soft Actor-Critic (SAC) models. In both of these cases, an agent uses one of these decision making models to perform tasks in an environment. Some decisions in the tasks provide the agent's model with a reward value. The reward value is used to calculate the appropriateness of the decision making policy within the model. Over time, the single-task or multiple-task objective is completed, often in a continual learning process. In genetic algorithms, a continual learning

process spans over a period of time where populations of agents learn according to a fitness function, providing the 'reward' for agents individually and/or as a whole population of agents. Thus, both RL and GAs are similar in their overall design for continuous learning, but the mechanisms by which policies are constructed and rewards are given are specific to the model design.

In the works of [20], a single-task continual reinforcement learning objective is used to test how context-dependent relevance to tune the information in the models in such a way that the task performance is increased and memory requirements are decreased compared to other state-of-the-art RL models. The key insight to their work is that the sequential nature of learning is often dependent on prior contexts. They successfully showed that by incorporating some metric of context, the model is more robust to changes in input states because the distribution can adapt more smoothly by incorporating gradient information of recent context. Thus, by distributing prior knowledge into the current state, the dynamically leverages the plasticity/stability. However, their approach did not explicate how the information is managed in the weights of the network, or how long the continual learning tasks could be performed before the benefits of the model reach resource constraints. This work suggests that incorporating sequential contextual information across the continual learning process of the model has benefits for mitigating catastrophic forgetting.

The works of [21] further suggest that continual deep reinforcement learning can be modified to mitigate catastrophic forgetting using another technique called policy reuse, where task-specific policies are cached and switched between tasks. Their findings support this approach for lifetime learning. The key to their work is that multiple-task learning objectives can be learned independently over a continual learning process while minimizing catastrophic forgetting. Here the limitation is that the tasks are finite and pre-determined, leaving little room for adaptability and generalization to different categories of tasks and resource constraints for storing policies.

Nonetheless, the authors showed that task-capacity in multi-task continual learning benefits from dynamic policies that limit rewriting of task-specific weights.

[22] showed that evolutionary algorithms to tune hyperparameters, the operator chosen variables, of a RL DQN single-task objective allows the policy to traverse the appropriate decision space while adapting to changes in the input state. The key here is that a global optimization using GA allows the model to adapt to changes in the environment. Furthermore the authors showed supporting evidence that the method could be used to control a swarm of drones to collaboratively clean up oil spills. This implies that the global optimization accounts for individual task differences for the global task objective.

Synthesizing the findings from those key articles, this work argues that a continual learning process for a RL model could benefit from a multi-task objective that shares policies to mitigate catastrophic forgetting. They also suggest that a GA could be employed to help tune the individual tasks differences to coordinate a shared global objective. Yet, there remains a central theme amongst these sources, as well as others, that points to resource constraints as a critical component of the life-long learning design.

Artificial intelligence algorithms and genetic algorithms to date have not found a work-around for managing the resources in such a way that solves the stability-plasticity dilemma indicative of catastrophic forgetting. However, the work of [23] details one such system that takes resource constraints as a central consideration, the human brain. In this work, the authors build a framework for understanding the implications of resource constraints as it relates to cognitive modeling and consciousness. The key point of their work is that evolutionary pressures of resource constraints are responsible for designing the topology, or architecture, of the brain. From the connectionist perspective, topology of this sort directly allows neurons to multiplex information, being used for storing and acting upon multiple information

sequences. Combining the previous relevant literature with this framework suggests that employing a GA to simulate the evolutionary resource constraint pressure may give rise to a new mechanism to protect against catastrophic forgetting.

Dilemma

Biological brains are highly skilled in learning and adapting to novel contexts. Yet, computational models seeking to employ a similar level of intelligence in the human brain are limited in their learning capabilities and even more so in their adaptive abilities. Specifically, computational models to date lack the ability to preserve knowledge for later use, which ultimately keeps resource requirements low. Any computational or biological system is bound to limited resources. This key constraint is largely unconsidered in current AI and AGI architectures. This work seeks to understand how architecture plays a role in preserving knowledge. Although many studies have been conducted on topology of intelligent algorithms, life-long learning, and learning without forgetting, an investigation on how these variables play a role in preserving knowledge is not available. Furthermore, when designing an AI or AGI system, there is little scientific methodology on how to construct the topology for the types of inductive biases and learning tasks at play. Encoding higher concepts, or concepts built on-top of other concepts, is not well studied in current literature. Here, an evolutionary approach is employed which subjects emerging architectures to pressures of resource constraints, similar to those under which the human brain evolved. More specifically, this work investigates how information is multiplexed between computational neurons in neural networks to encode high-level concepts that can be used between multimodal learning environments.

Research Questions and Hypotheses

Research Question One

How does changing architecture preserve knowledge?

Hypothesis 1 Architectures that self-modify neuron weights in a way that manages the trade-off between existing information and new information in relation to sequential specific goals preserve knowledge.

Hypothesis 2 Architectures that self-modify by rewriting existing neuron weights are able to adapt to changing modalities by optimizing for consecutive tasks.

Research Question Two

How do evolutionary pressures of knowledge preservation affect architectures for learning and using high-level concepts?

Hypothesis 1 Architectures with neurons which are shared across different computational abilities display more multiplexing of learned information.

Hypothesis 2 Emergent architectures from evolutionary pressures of knowledge preservation aid general learning and use of a high-level concept.

Chapter 2

LITERATURE REVIEW

History of Computing

The term 'computer' has morphed over this history of computing devices, but has generally evolved to represent a singular instrument of digital, mathematical computation. Yet, the word computer didn't always mean what it means today. The first use of the word computer is mentioned in 1613 in The Yong Mans Gleanings by Richard Braithway [24]. The term was originally used to refer to a person who carried out arithmetic calculations up until the 19th century.

Computers actually have a huge, vast history, dating all the way back to around 200 BC with the first analogue computing device, the Antikythera, which was used to predict astronomical positions and eclipses decades in advance [25]. An analog computer is a device that works off properties of nature, physiological properties, either based on liquid or based on mechanical power. Analog computers are mechanical devices that carry out mathematical sequences within a continuous number space using physical structures (instead of digitally, as we've come to think of them). Analog computers have a long history going back to the Mesopotamian Abacus device (from as early as 2300 BC) which calculated basic arithmetic [26]. The Abacus was used to calculate basic addition and subtraction. This device was found in many other regions in Egypt, Persia, Greece, China, Rome, and India. Abacus were used into the early 20th century in schools to teach lessons. More complex versions are capable of solving square and cubic roots.

Computers as we know them today were inspired by the Jacquard Loom, an analogue device for creating textiles using punch-cards to carry out a sequence of

sewing operations [27]. In 1822, Charles Babbage prototyped a mechanical computer called the Difference Engine, capable of solving polynomial equations [28]. Later, in 1936, Alan Turing developed a series of technical articles which described computing devices known as Turing machines [29]. These along with other historical works inspired the first general-purpose digital computer, ENIAC, built in 1943 [28]. The ENIAC was a computer the size of a 1,500 square foot room used to calculate ballistics for the U.S. government. John Von Neumann helped develop the ENIAC and his work in applied mathematics led to the creation of classical computers as we know them today [30].

Shortly after the invention of the digital computer, the brilliant mathematician Alan Turing, who studied cryptography on the famous Enigma machine in WWII, hypothesized intelligent machines [31]. His work still serves as a reference point for intelligent algorithms today. Yet, the birth of artificial intelligence can be attributed to John McCarthy and Marvin Minsky in 1956 at the Dartmouth Summer Research Project on Artificial Intelligence [32]. However, the neuroscientific work of Hebb in 1949 [33] inspired the work of McCulloch and Pitts in 1943, which provided the modern neural network in [34] This work was shortly followed up by the influential perceptron model of Rosenblatt in 1957 [35], which most current neural networks model in some form.

Notably, Turing, McCulloh, and Pitts' work all demonstrate the connectionist paradigm. Although Turing's work in [36] is of particular interest to this study as he detailed the design of machines that could enable and disable connections between neurons (i.e. unorganized machine), and is linked to modern day reinforcement learning [29]. Furthermore, Turing elucidates that the human cortex can be thought of a sort of unorganized machine, that through evolution and punishment/reward feedback, organizes into an intelligent machine, similar to human intelligence.

This work and many influential scientists cultivated the machine learning field

that flourished from the late 1950's through the 1980's [32]. At this point, machine learning was largely of systems which were explicitly programmed to execute tasks that mimicked intelligence, however the computational complexity of computers at that time were underwhelmingly capable of producing intelligence as we understand it in by human comparison. By the 1980's expert systems and deep learning by John Hopfield, David Rumelhart, and Edward Feigenbaum [37, 38, 39]. These advancements were critical for igniting today's intelligent algorithms.

While the history of computing and artificial intelligence is rich enough to write a multiple novel series on, the revolution of deep learning seen today can be largely attributed to Rumelhart's error propagation techniques coupled with accelerated hardware of parallel processing units and audio differentiation [40, 41]. Current AI is largely overtaken by deep learning methodologies that use auto differentiation to backpropagate errors on accelerated parallel processing hardware. A particular subset of machine learning is reinforcement learning which originates from Markov decision making processes (MDP) [17]. However, state-of-the-art reinforcement learning employs deep learning neural networks within the MDP. One of the best performing models across tasks is that of Deepmind's Deep Q Network [42, 43]. This particular model is robust in solving a variety of tasks by relying on the backpropagation methods of deep learning.

Backpropagation

Backpropagatiaon was first implemented as a method for learning patterns. Weights in an artificial neural network are repeatedly modified to minimize the difference between the ground-truth label and the network's prediction. This difference, known as the loss, is used in a partial derivative step which designates the direction the weights should change to create a smaller loss during the next iteration. Backpropagation relies on an optimization objective [44]. As part of the weight update process, a

gradient is partially calculated from the environmental signal, whether it be reward, error, etc.. In theory, a complete gradient exists for any one set of inputs for a given task. If the task or input changes, the gradient changes. Backpropagation affords little room for encoding patterns from the input that could be used across different gradient landscapes. This is in part because the features learned from one gradient's optimization are specifically tuned to be useful for the current task. Thus, features learned to minimize one error surface are unlikely to minimize another without careful construction of a unified objective [45] or task-specific finetuning [46]. These limitations result in narrow, domain-specific learning and inefficient learned representations, as demonstrated by the success of post-hoc weight pruning algorithms [47, 48] that ablate a majority of learned weights in a network with minimal degradation to performance on the training task. Without the constraint of adding new neurons to neural networks for encoding new information as tasks capacity increases, backpropagation is limited by resource constraints.

Genetic Algorithms

Genetic algorithms (GA) may provide a unique solution by affording generalization while making use of existing resources. In [49], authors highlight GA used in RL as a means for overcoming the constraints of backpropagation caused 'forgetting' of prior learned information. GA models which engage in self-organizing, autonomous processes employ two processes, top-down (global, population feedback to the objective) and bottom-up (mutations and crossover of information components), for a more in-depth discussion see [50]. The causal relationship between the two is partially decoupled, which allows for local components to evolve somewhat independently of the overarching top-down metrics. The result is that the history of the system is interjected from the top-down process, while novelty is introduced from the bottom-up processes. Generalization to novel circumstances demonstrates resource constraints

by repurposing existing knowledge with few examples and few updates to learned representations. Indications of this claim are apparent in the evolutionary mechanism which employs a long-range temporal success metric in [51]. However, we suspect more rigorous analysis of how to curate general, emergent adaptability is needed. The overall emergent nature of these systems may show some degree of extrapolation, which would abet deep neural networks in some degree of general learning under resource constraints.

Artificial General Intelligence

Meanwhile, expert systems from the 1980's paved the way for a different type of artificial intelligence, namely symbolic architectures. Ultimately, this path has led to the body of work delineated from AI as artificial general intelligence (AGI). Spawning from the ideas of thinking machines from Turing, science fiction took a particular interest in humans matching and exceeding human intelligence. First in 1872 [52], and later for the android character, Data, in Star Trek: The Next Generation [53] and fully in Stanley Kubrick and Arthur C. Clarke's Space Odyssey [54], although there are many others. Albeit a plethora of science fiction has cultivated the idealization of AGI, the core scientific pursuit has a rich history in an alternative approach to the connectionist paradigm aforementioned. Here, the key difference between AI and AGI is that AI relies on connectionism while AGI relies on symbolism.

In practice, AGI systems have largely converged on graph knowledge based architectures [55]. These approaches do not suffer from the same rewriting of information as backpropagation AI systems do. Instead, they add new nodes to correlate new information in relation to existing information. Some multiplexing is involved by modifying types and meta-information in the graph architectures [56]. Although these models are more robust to task switching and continual learning, they suffer from resource limitations. If the AGI is expected to learn the vast depth of knowledge that

is available to the current generation, then at best it would understand the meaning of all words and the relationships between those words. In the English dictionary, this amounts to 171,146 different information objects, or words [57]. The combinations of these objects is combinatorial intractable in current computing architectures. In the prominent AGI architecture of Hyperon, the knowledge graph manages resource constraints by learning to 'forget' or learn according to the current state, past prior information, current tasks, and referencing priors for recovering future information. Although this model is projected to be formidable in its ability to tackle real-world problems, AGI is still limited in resource constraints as we consider the vast explosion of information to be known has increased exponentially over the last decade.

Knowledge Preservation

One defining characteristic of knowledge preservation is resource conservation. Intuitively, practical models are those which can mature without allocating new, dedicated resources in each novel learning case. Thus, resource constraints force the underlying learning mechanisms to learn new information without degrading prior knowledge. One mechanism such a model might employ is the multiplexing of information units for multiple tasks in such a way that finite resources can be used to contend with human level intelligence.

Thus, a major constraint of intelligent algorithms is resource constraints and life-long learning, both in artificial intelligence in the form of catastrophic forgetting and in AGI in life-long learning for reconciling current knowledge and future knowledge within hardware constraints. We can then surmise that the problem of catastrophic forgetting in deep learning is more than a plasticity-stability management problem, it is rather a resource allocation challenge.

Theory Relevant Literature

The work proposed here is thus an approach informed by the history of intelligent algorithms in both AI and AGI with a critical need to overcome catastrophic forgetting (CF), and more broadly, the need to manage a finite number of resources. However, there are different existing approaches for designing computational models of intelligence that can overcome CF or allocate resources appropriately.

Connectionism

Connectionism is a theory that frames learning as an incremental process. This framework relies on connections between neurons to encode information from an input stimulus and produce an output response [58]. While the overall schema of connectionism has many implications, the most important here is that of topology. Biological neurons are not arbitrarily exchanging information. They are particularly constructed in space and time. Their morphology largely dictates how information is allowed to travel and be utilized throughout the brain. Effectively, what other neurons any one neuron is receiving inputs from and what neurons it can output too will constrain learning. This theory has built the foundation for modern deep learning approaches.

Symbolism

is different from connectionsims; that is, unique, indirect representations, or symbols, are developed in the brain, which concretely surmise to the knowledge of the mind [59]. The 'mind' is composed of symbols, which can be called upon to interpret incoming information. Here, language is especially important as it holds the symbol which can take on various meanings, depending on the circumstances [60]. This concept is often referred to as the 'Language of Thought' hypothesis, which states

that symbol manipulation is a product of mental language, a type of symbolic information representation, that once combined provides meaning with syntactic and semantic structure [61]. Mental language is not exactly the same as grammatical language, taking on a unique representation of information that is only meaningful once context is provided by other symbols and signals. The main idea here is that cognition results from manipulation of mental representations [62]. Symbolism is directly embedded in classical cognitive science.

Artificial General Intelligence

Existing models of artificial general intelligence, which mostly use cognitive science architectures. Most prominent approaches in AGI rely upon symbolic and subsymbolic models. For a more detailed description, see [63]. Of these works many employ the techniques of graph networks and predicate logic [64]. In this case, information is learned symbolically and manipulated through nodes and edges in the graph to produce a construct of knowledge. Nodes denote subjects and objects, while edges denote relationships (i.e. predicates) between them [65]. Symbolic learning in these cases is mostly measured in terms of similarity between simulation results and human behavior [66]. The most notable of these approaches are still in early development and much remains to be determined in terms of how similarly they model human intelligence in terms of knowledge preservation.

Both the connectionist and symbolic frameworks for artificial intelligence and artificial general intelligence share the element of combinatorial expression of information processing units as a key feature of learning. What is largely implied in these frameworks by their metrics of learning is the varying degrees to which the model can retain information for later use, or knowledge preservation. What is missing is a unifying understanding of architecture, independent of the artificial cognitive framework, as it relates to life-long learning without forgetting.

Neuro-Symbolic Architectures

Neuro-symbolic architectures are those which take input information and create symbolic representations. Their approach draws from the original symbolic architectures first employed for artificial intelligence in the 1950's and combines it with newer techniques in gradient optimization [67]. Concept embeddings are often stored in knowledge graphs which can be manipulated to produce some artificial semblance of logic and understanding. Graph neural networks are a particular kind of architecture that aim to integrate the successes of deep neural networks for non-grid data [68]. In these cases, the issue is either CF or the need to expand resource requirements to incorporate higher-level concepts as knowledge is transferred across domains. New information has to be learned on each occasion and it is difficult to utilize prior knowledge in a completely novel context.

Backpropagation as a Plausible Brain Mechanism

Meanwhile, there is little evidence to support that backpropagation is a plausible mechanism for the human brain. For further explanation of various learning models and their implementation of backpropagation in a biologically plausible fashion, we refer to the works of [69]. Here we offer some additional concerns to those already presented. In speculating if backpropagation could be used by biological neural networks, the first concern is that brains are not provided a ground truth for which to calculate a loss. Very few data-points in any one moment of sensory perception receive a true label (e.g. teacher signal). Yet, animal and human brains still afford large degrees of generalization amongst similar instances and in many cases, context is used to extrapolate new inferences and predictions for which categories (i.e. labels) have not been provided. Secondly, the amount of data and teacher signals needed for even a small child to generalize to a vast amount of other out-of-distribution data is far

less than what is currently employed in any deep learning technique. Lastly, we turn the topological structure preserved amongst many vertebrate brains. Lack of direct connectivity between top layer to lower layer neurons in any one sensory processing stream suggests that an error signal, or any sort of backpropagated information, is unlikely. In nearly all biological vertebrate brains, evolution has preserved a modular hierarchy of neurons [70]. To date, there is no evidence of a physiological mechanism in the human brain which would provide individual neurons with direct feedback from the environment, other than those involved in stimulus-response coding. Otherwise, either long-range connections from the higher cortical regions detecting categories or a separate molecular mechanism to propagate information backwards would need to be in place to tune the synaptic connections between neurons at every cortical level involved in the sensory processing stream. Yet, this is precisely the procedure used in backpropagation. To further make this point clear, the feed-foward information in a DNN is distinct from the bakpropagated information, but in biological neurons, the information from one neuron cell to another is the same chemical-electric process. These considerations suggest that backpropagation is not the primary learning mechanism of biological brains, for which we've seen copious amounts of broad intelligence.

Furthermore, while this work agrees with [67] in their conclusion that equilibrium propagation (EP) is a unifying backpropagation framework, this work doubts that it will lead to knowledge preservation. EP is proposed as a method to integrate various DL models into one by constructing the global optimization objective as minimizing free-energy. However, the current method of which free-energy is optimized still relies on some form of backpropagation, which has no constraint or initiative to conserve past learning. The exceptions to this are predictive coding models, which can still be optimized in terms of free energy and do not suffer from the concerns previously mentioned for backpropagation. While the preliminary results of the predictive coding model put forth by Ororbia, et.al. [71] shows a different approach

to CF mitigation, known as the Sequential Neural Coding Network (SNCN); it has yet to demonstrate knowledge preservation without explicit tasks descriptions and target labels, a concern previously shared with backpropagation's role in CF. Due to the dual-optimization technique of SNCN, the gradient is large in comparison with standard artificial neural networks, suggesting that resource constraints play a role in converging on a minimum as knowledge is accumulated throughout the life-long learning process. The result of these constraints is increased probability of inciting prior learning re-writes. Indications of this concern are addressed in detail in the discussion on limitations in Ororbia, et. al.'s work.

While minimal changes to brain organization are possible, the majority of neuron connectivity is already established by age three [72], where more connections between neurons are present than necessary. As part of early developmental learning, an extensive process of synaptic pruning removes unnecessary connections. This process further supports the connectionist theory; that neurons have to have the ability to fire onto each other to facilitate learning. Thus, the initial topology of any brain, biological or artificial, is crucial in dictating future learning. Furthermore, this feature has largely been overlooked in artificial models that 'scale-up' as they learn instead of 'scaling down', as in biological brains.

In [73], the authors offer insight that rather than relying on the previously thought-to-be primary mode of extrapolation learning, it is rather likely that the brain is capable of using interpolation for generalization due to the large observation sampling size. They underline this point with evidence from big data models which are increasingly gaining human-level performance. While we candidly agree on the overall premise of their work, we suggest here that while interpolation may serve as a means for generalization in the brain, the mechanism by which it does so is not directly comparable to interpolation in deep neural networks (DNN). In the latter case, excessive computational resources are required to learn from copious amounts of data in order

to produce human-level performance.

The scalability of DNNs to human-level tasks has recently been scrutinized in several works [74, 75]. While the human brain has 86 billion neurons, some language models (e.g. GPT-3) have more computational nodes; yet we do not see the same retention of knowledge. GPT-3 is particularly suited for language learning, but does not transfer knowledge between tasks in the same proficiency as the brain. It is clear that models such as these are constrained to pattern storage and retrieval tasks. Applying GPT-3 to seemingly unconstrained tasks, such as question answering, requires careful, prompt engineering and problem constraints to avoid erratic predictions [76]. These models fail in comparison to, even in the case of animals with considerably fewer neurons, the ability to perform a range of goal-directed behavior and learn new tasks without forgetting old tasks.

Simply put, a brown mouse, with 200 million neurons and an estimated 4.48X10$\hat{1}$1 synapses [77], shows a higher degree of knowledge retention than any known DNN. This is .001 the number of neurons and 25.6 times the number of synapses to computational neurons of GPT-3 [78]. In the case of neurons, we see much more knowledge preservation in the mouse and in the case of synapses, we suspect that we do not see 1/25 the amount of knowledge preservation in GPT-3. Here, a mouse is able to use information previously stored in its brain for a variety of cognitive functioning without being re-trained on every task; while GPT-3 requires re-training for each individual task and cannot generalize out-of-distribution. We suspect big data paired with traditional deep learning alone is not enough to create the type of knowledge preservation we see in biological counterparts and suggest that extrapolation is the important difference between the types of general learning occurring in both DNNs and highly preserved vertebrate brains.

Contending Frameworks and Theories of Brain Activity

There are several leading frameworks for constructing brain-inspired models from. This work examines the most prevelate models to date in relation to their success in cognitive modeling as it relates to intelligent algorithms.

Global Workspace Theory Global Workspace Theory for consciousness, was first proposed by Baars in 1988 [79]. The theory has been adapted to Global Workspace Dynamic (GWD) that provides a theory of conscious and unconscious brain events [80]. While the theory is expansive in the amount of literature and scope of detail, in summary, Baars describes brain activity as a series of activations that collaborate in ensembles with emergent wave forms of cascading activations. Early works implementing GWD can be found in Stan Franklin's LIDA architecture [81], demonstrating learning and memory as an autonomous agent, requiring built-in task-based priors. Another notable implementation by Dahene and Changeux's DCM model [82]. DCM demonstrated GWD applicability to visual consciousness. Most prominently, the work of Bengio [83] that indicates a shared workspace implemented in a deep learning model can encourage specialization and facilitate synchronization between specialized components. Amongst these implementations, the problem of task-based resource allocation persists. Integrated Information Theory Integrated Information Theory (IIT) postulates the physical systems of the brain under the axioms of consciousness and was first surmised in by Tononi in 2004 [84]. This mathematical theory of consciousness includes a quantitative measure, phi, of the integrated information in a physical system. The larger the value of phi, the more conscious the system is. IIT is more inclusive than just this measurement of phi, but several prominent scientists, such as Scott Aronson, have scrutinized the measure of phi as a central component of IIT [85, 86]. According to the mathematical formulas for phi, an arbitrary complex mathematical system can be constructed in such a way that it would

clearly not be regarded by most i individuals, but would have a significantly high phi value. Implementations of IIT include a toolkit for measuring phi and various reports of using phi in existing models [87, 88, 89, 90]. However, there are no known reports for a IIT cognitive computational model at the time of this work. Predictive Processing Predictive processing, first proposed by Roa and Ballard in 1999, is a neuroscientific framework for describing the core function of the brain as an optimization task to minimize errors between real sensory input and predicted input [91]. There are multiple accounts of human behavioral phenomena that are not captured in the previously mentioned models that can be described by predictive processing, such as auditory hallucination during low decibel listening, inability to self-tickle, pre-motor movement firing, and binocular rivalry [92]. The most successful application of these models is in perception-action loops [93], which demonstrates that task learning is well suited for prediction based modeling in ways that rival reinforcement learning. Other notable works in [94, 95, 96, 97] show strong results in figure-ground segregation, speach, image, text representation learning in 3D environments, and most prominently, video representation learning. While these reports show predictive processing as a useful computational tool, their ability to generalize and adapt to new context over a continual learning process is not currently supported. Predictive coding models thus suffer from catastrophic forgetting since they are backpropagation based methods [98]. Furthermore, the physiological basis of predicting error is questionable when considering the infant brain and how the mechanism may first occur (i.e. how is an error predicted in the first instance of neuronal firing). The Recommendation Architecture The recommendation architecture (RA) is particularly interesting in relation to the work proposed here, since a core feature of the framework is resource constraints. Proposed by Coward in 2001 [99], although the preemptive work can be dated back to 1995 in a pattern extraction hierarchy framework [100]. The Recommendation Architecture is a conceptual framework for understanding how cognition

occurs in terms of anatomy and physiology. In RA, the overall function of the brain can be divided into two systems: condition detection and definition comprised of the cortex, hippocampus, and amygdala; and the behavioral selection system comprised of the basal forebrain, thalamus, dorsal basal ganglia, and ventral basal ganglia [101]. The cerebellum, brain stem, and spinal cord manage behavior sequences and implementation.

RA is particularly interesting because it relies on multiplexing information in the cortex for behavioral response. This occurs when the cortex encodes abstract hierarchical patterns that are later interpreted as recommendations for behavior by the behavior system. This topology and corresponding learning algorithms allow neurons to generalize and bootstrap prior information, ultimately leading to the ability of the brain to have a somewhat fixed number of neurons and connections [102]. Thus, we adopt RA as the appropriate leading cognitive framework of which to inform our cognitive modeling.

In the work proposed here, RA applies to the problem of resource constraints as demonstrated by catastrophic in deep learning models. As the theory suggests, if the topology of the computational brain is organized according to the evolutionary pressures of resource constraints, it is likely that the result will be similar multiplexing of computational units as neurons in the brain.

Current Empirical Literature

Catastrophic Forgetting

Catastrophic forgetting is observed as a failure to perform and to generalize to previous goals due to changing weights for the sake of the current goal by re-writing the previous weights learned during a previous goal [103]. CF is also observed when the input data distribution critically shifts during same tasks problems [104]. Besides

that of Ororbia, et. al., most successful work on overcoming CF involves slowing the weight-update process during the model's lifetime [105], known as elastic weight consolidation (EWC). While these are effective strategies, there are still resource limitations constraining generalizability. Excess resources are needed in either the number of neurons to encode patterns of the data and/or in the amount of data needed. Other methods for alleviating CF involve sharing learned representations of past inputs during a downstream processing state, sharing weights between tasks, and pseudo-rehearsal where past tasks are passed through the gradient again [106, 107, 108, 109, 110]. Another alternative method is asynchronous updating as in [111, 112, 113].

Dropout is another technique which tries to allay overwriting of weights by periodically 'freezing' random populations of neurons at various weight-update iteration steps [114]. The effect here is similar to EWC in that only a partial set of neurons are tuned to the current input batch, allowing the non-tuned set to persist down-stream until further update. Catastrophic forgetting is typically measured by accuracy and speed in learning a recurrent sequence of a task or tasks [104, 115, 116, 117, 118] Some more advanced metrics can be implemented to determine tasks overlap, layer-wise stability (i.e. rewriting percentage), and rate of forgetting [119, 120, 121].

The theme of works involving CF mitigation is to disperse learning over temporal ranges, which has afforded such models to learn some sequences of tasks and avoid over-fitting. However, these models are still limited by the foundational crux of backpropagation: that narrow features are inherently encoded from the optimization process and sooner, rather than later, the model will reach resource constraints. There remains the problem of storing prior general learning to be used in a broad manner.

Transferring Knowledge

Transferring knowledge from one task to another is formally known as transfer learning [122]. In these techniques, weights are pre-trained in a neural network on a broad task and then imported and fine-tuned to a specific task. One such promising approach is that of Google Brain's PathNet which uses a modular architecture to swap out computational units depending on the tasks [123]. However, the idea of bootstrapping a-priori knowledge is the larger goal of knowledge preservation that uses existing architecture to provide information, without requiring additional computational resources. The metrics for transfer learning are how quickly and accurately the new model can learn with pre-trained weights.

In general, the ability to preserve knowledge should manage resources in such a way that the system is able to learn continuously over its lifetime without continuously needing new computational resources. Part of this ability is due to the fact that prior information can be sequestered in a novel context. Transfer learning techniques are similar, but they rely on additional resources to transfer to the existing system. While the methods for catastrophic forgetting also attempt to alleviate the resource constraint problem, the current methods have resource asymptotes that quickly reach peak learning abilities far below that of human and animal memory abilities.

Neuroevolution

Neuroevolution is another potential approach to preserve knowledge that has thus far been largely neglected is to consider the topology of the learning system — that is, its 'architecture'. Some approaches in neuroevolution of artificial networks attempt to answer this question, but none have shown emergent architectures in simulated environments systematically. While there is a significant body of work on architecture optimization for other metrics of learning [124, 125, 126, 127, 128, 129], to

date there has been little work considering architectural choices in relation to CF and/or transferring knowledge. Fortunately, millions of years of evolution has generated highly conserved, vertebrate and mammalian brain architectures that have been optimized with regard to intelligent functions including, it is reasonable to assume, the preservation of learning without CF.

The work proposed here combines deep reinforcement learning and genetic algorithms within a resource constrained environment for the goal of preserving knowledge amongst sequential tasks.

Continual reinforcement learning (RL) is an emerging field that describes an autonomous agent that can learn multiple tasks over time [130]. Within the framework of reinforcement learning, several works have shown interesting results in relation to catastrophic forgetting (CF). Likewise, the works of [131] employs a meta-learning algorithm to tune an RL model to extract knowledge between similar tasks in multi-tasks environments. In [132], a framework with a simple feed-forward neural network is used in conjunction with pseudo-rehearsal to overcome CF with some resource constraints. In [133], a decomposition-based reinforcement learning environment with evolutionary strategy is used to explore CF in a sequence of tasks. The results show that the first tasks become less optimized and the system favors plasticity over stability in the model's weightsk, leading to increased learning time and overall performance. Furthermore, critical knowledge was replaced due to the greedy behavior of the genetic algorithm (GA). The work of [134] describes a path to AGI that relies on reinforcement learning and Hebbian learning by constraining decision-making that aims for satisfaction of multiple physiological needs in sequence-based tasks. The work here borrows from the theoretical framework proposed that Hebbian learning within the reinforcement learning paradigm described can aid in adaptive generalization for sequential learning.

Chapter 3

METHOD

The work proposed here is adopted from previous works that use a continuous RL model for recurrent sequences of multiple tasks in simulated behavior-action loops. Similar to meta-learning in which a second optimization protocol is used to optimize the original neural network optimization tasks [135], an approach is used to tune the network on a global level beyond the local model optimization methods using a (1+1) evolutionary algorithm and a hyperparameter search technique. The methodology uses a basic framework by employing a simple DQN-based model. The DQN adopted from [42] is fitted with experience replay, a type of pseudo-rehearsal, but uses a simplified fully-connected network. In the work proposed here, the evolutionary algorithm fitted with a resource constraint metric is used to manage the imbalance between plasticity and stability, and mitigate the effects of greedy optimization in the GA. The overall optimization goal of the system is to maximize reward across all tasks in the sequence.

Environment

The direct mapping of an agent's action to the resulting environment behavior is imperative for simulating human-like cognition in the embodied-mind approach. Unity3D offers easy to construct simulation environments that allow flexible and rapid changes during experiment testing of artificial agents. The built-in physics simulating feature provides constraints of physics as we would experience them in the natural world. The built-in temporal component of these environments also offers an opportunity to study simulated environments within the fourth dimension that may

not be easily accessible by standard computer vision, natural language processing, and extracted time-series datas alone. Unity3D has another built-in package, ML-Agents that offers state-of-the-art reinforcement learning algorithms easily deployable on the custom simulated environment [136].

In this work, a custom environment is constructed as the basic environment with x, y, and z planes. The environment consists of a fox-like artificial agent, a cylinder, a cube, and a plane, as seen in Figure One. This comprises the base environment and remains the same for all sub-environments, unless otherwise noted. The fox, or rather agent, looks similar to a real-life fox in that it has four legs, a face, and a tail. However, the agent moves as a sliding cube; physical coordination between body parts is assumed. The environment affords the agent one of six discrete actions: left one unit, right one unit, up one unit, down one unit, and rotate of a unit left or right. A reinforcement learning model is used to control the agent's movements. This agent has a mass of 10. Gravity is enforced on the agent such that if it moves off the edge of the plane, it will decrease it's coordinate in the y plane. The plane, cube, and cylinder are massless objects that cannot move on the x,y,z plane; they are static objects. A light source is cast from above the environment, directly onto the plane. Unity3D version 2020.3.21 with mlagents version 0.26, 2.0 was used for all environments in this study.

Four sub-environments are created by enabling different sensory organs on the fox: vision, position, raycast, and audio. For vision, the fox receives a camera attached to the fox's eyes with a focal length of 14.4 units and field of view of 79.5 units, giving a state vector input shape of 24 by 24 pixels for the red, green, and blue channels. For position, the fox is given a state vector of its x,y,z position in the environment, it's x and z velocity, the cube x,y,z coordinates, and the cylinder x,y,z coordinates. For raycast perception, the fox has one raycast object (a built-in mlagents sensor similar to lidar). This object extends in front of the fox's face from the eye level pointed

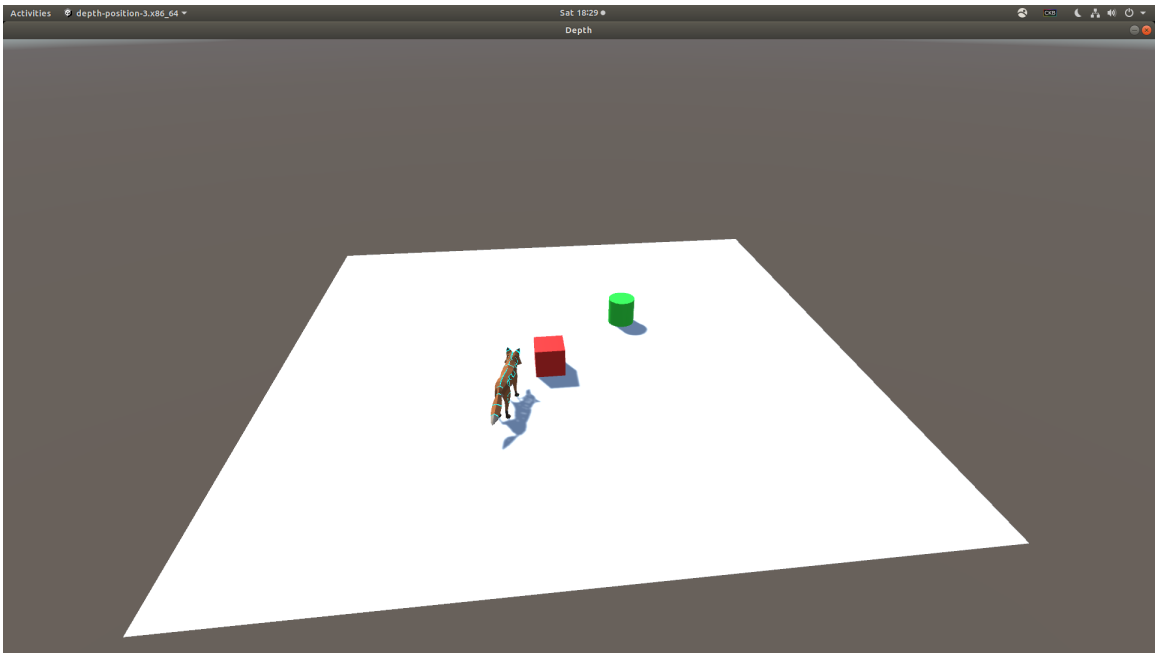


Figure 3.1: The simulated environment includes a fox agent, a cylinder and cube as target objects, and a plane serving as the floor.

towards the plane at a 45 degree angle, with a ray length of 3 units. When the sphere of radius .34 units comes into contact with an object, that object's x,y,z position is reported as the state vector input. For audio, the cube and cylinder broadcast a recording of a cat's meow. This auditory signal decays according to the protocol set forth in [137]. The state input vector is the signal once it reaches the fox's location. These four sub-environments are evaluated independently of one another such that the model controls the agent for only one sub-environment at a time, where only one sensory organ is placed on the agent. A fifth sub-environment is created by using the base environment. However, in this environment, no sensory organ is given. This allows for a control environment to be formed in which no state input is given.

A sixth sub-environment is formed without using the base environment. Here, we use a new environment, Match, which is customly crafted as a control. In this environment the agent must navigate to a target position. The state input is a 40,000 vector input of the target position statement. The agent is a cube suspended

without gravity. Action behaviors are to move up one, down one, left one, right one, forward one, or back one unit. The target position state is a number, one through six, denoting each of the possible actions, but repeated in the state input vector for 40,000 dimensions. If the target is to move up, the input value is zero and the state vector is a vector of 40,000 zeros. To get a reward of one, the agent must choose action output 0. The environment can be seen below in Figure 2 where the square is the agent which has received a target signal and the objective is to select an output which corresponds to the target signal. In each episode, the target signal is chosen at random. This environment was chosen as a control to the depth tasks in previous documents since it does not share core concepts.

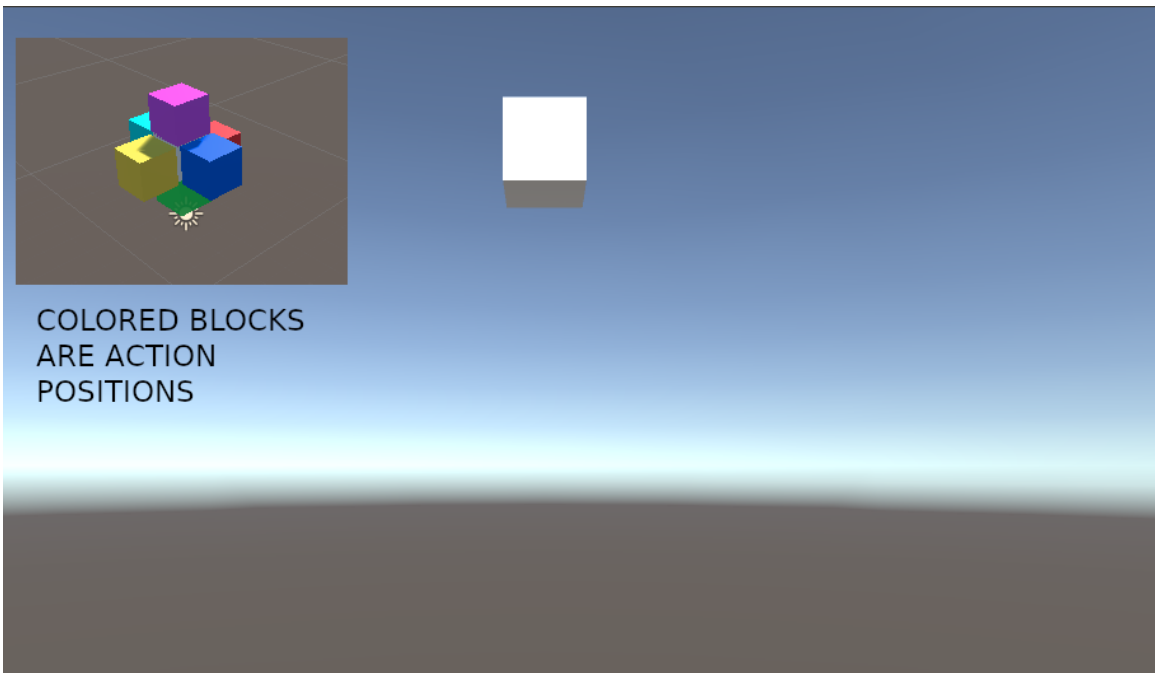


Figure 3.2: A custom mlagents environment where the cube agent is given a target vector. The agent selects the action output corresponding to the target vector value.

A table of all environments is given below.

Environment Name	State Input	Action Output	Reward	Goal
None-Fox	None	Rotate left, rotate right, move up, move down, move right, move left	-1 falling off plane -1 moving to further target +1 moving to target -.01 for each step	Move to closest target
Visual-Fox	24 by 24 visual input from agent view camera	Rotate left, rotate right, move up, move down, move right, move left	-1 falling off plane -1 moving to further target +1 moving to target -.01 for each step	Move to closest target
Audio-Fox	40,000 state vector of audio signals from target and non-target sources	Rotate left, rotate right, move up, move down, move right, move left	-1 falling off plane -1 moving to further target +1 moving to target -.01 for each step	Move to closest target
Position-Fox	Agent x,y velocity, agent x,y,z position, target x,y,z position, non-target x,y,z position	Rotate left, rotate right, move up, move down, move right, move left	-1 falling off plane -1 moving to further target +1 moving to target -.01 for each step	Move to closest target
Raycast-Fox	Vector input of received raycast object, size 9	Rotate left, rotate right, move up, move down, move right, move left	-1 falling off plane -1 moving to further target +1 moving to target -.01 for each step	Move to closest target
Match	Target vector of desired action out value repeated for 40,000 dimensions	Select up, down, left, right, forward, or back	-1 selecting incorrect action +1 selection correct action	Match target vector input value to action output value

Table 3.1: Described here is each environment used throughout this work. A name, the state input, the action output, reward, and goal for optimization is given.

Computational Framework

A variety of reinforcement learning models are used for the computational framework of the agent. Which model controls the agent is dictated according to the experimental procedure described in the procedures section below. For baseline test of each environment set-up, the built-in mlagents reinforcement learning algorithms, proximal policy optimization (PPO) and soft actor-critic (SAC), are used with default hyperparameter configurations found in Appendix A and B [138, 139]. In the variable testing, a reinforcement learning with a built-in replay buffer, DQN, is used adapted from [42], for which exact configurations can be found in Appendix C. The DQN consists of three fully connected layers. Layers one and two are followed by the ReLU activation layer as described in [140]. The DQN uses smooth L1 loss as described in [141] and RMSprop optimization as described in [142] The DQN code is implemented in Pytorch version 1.8.1 with Cuda version 10.1 [143, 144]. Figure 3.3 is adopted from [145] to show the computational flow of a DQN.

The genetic algorithm used is a modification of a (1+1) evolutionary algorithm [146]. The hyperparameter tuning algorithm is used from Weights and Biases, a program for tracking model variables and performance, according to a random search as described in [147].

The motivation behind using mlagents built-in PPO and SAC algorithms was to

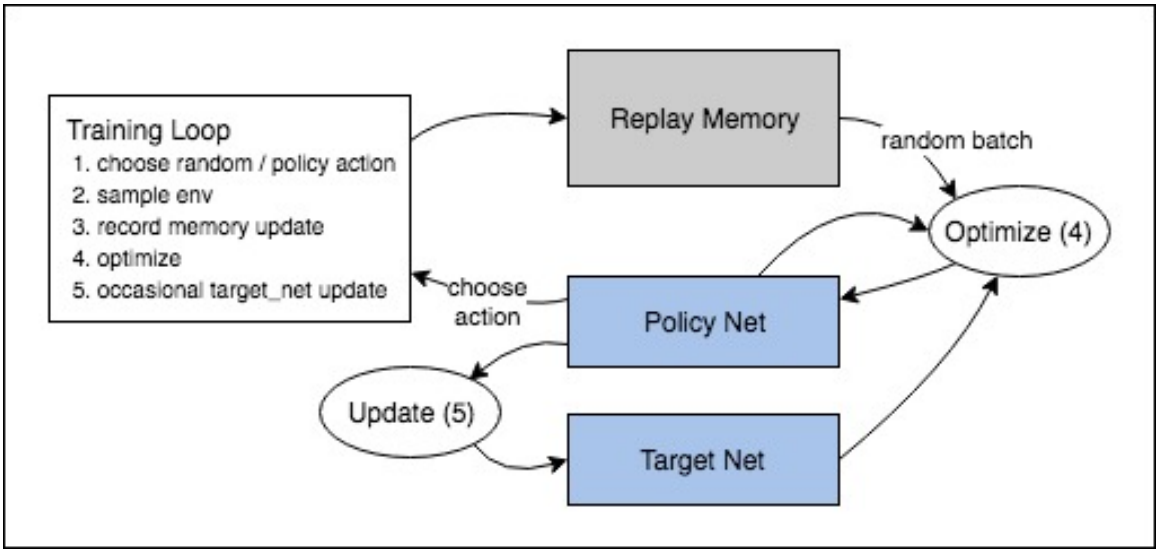


Figure 3.3: DQN Model Flow.

use state-of-the-art reinforcement learning policies as control measures to compare against the novel DQN approach and to identify if the custom environments are learnable. Supporting works for using these algorithms with mlagents can be found in [148, 149, 150, 151, 152]. The choice to use the DQN is supported by the ease of use in the evolutionary strategy, since weights in a fully connected system can be easily recombined and mutated and as previously mentioned, DQNs have a built-in pseudo-rehearsal to protect against catastrophic forgetting, further supported by works of [153, 154, 155, 156, 157, 158]. The work of [159, 160, 161, 162] motivated the use of the evolutionary algorithm as well as the desire to simulate evolutionary pressures of resource constraints as described in the Recommendation Architecture in [99]. The evolutionary algorithm used is a 1+1 GA, where through evolutions, mutation and crossover within each layer of the DQN weights occurs according to the resource constraint metric. If the number of non-zero weights is above the resource constraint, a random selection of weights are mutated to zero, else zero value weights are mutated to a random value between 0 and 1. The number of crossover weights is controlled by a single hyperparameter while the number of mutation weights is

controlled by three hyperparameters for each layer. Hyperparameter tuning with Weights and Biases was used due to its comprehensive model tracking and ability to quickly search for optimal hyperparameters.

The table below describes all models used in the computational framework.

Model Name	**Purpose**	**Brief Description**
PPO	Control	State-of-the-art proximal policy optimization
SAC	Control	State-of-the-at soft actor-critic
SAC+LSTM	Control	Same as SAC with additional final LSTM layer
DQN	Test – PPO, SAC, SAC+LSTM	Standard deep Q network with 3 fully connected layers
DQN+GA	Test – DQN-GA, DQN+GA-RC	Same as DQN with evolutionary algorithm and resource constraint metric
DQN+GA-RC	Control-Test – DQN-GA	Same as DQN+GA without the resource constraint
DQN-GA	Control	Same as DQN+GA-RA without crossover and mutation

Table 3.2: Each model used in this work is listed by name. The purpose of the model (control, test, or control-test) for analysis in relation to other models and a brief description of each model's set-up is given

Measures

In order to assess a model's knowledge preservation, it must be able to demonstrate that it can learn a sequence of tasks in a variety of task specific goals and return to those tasks with improvements in computational costs and/or learning time. Below we describe initial tests to measure generality in terms of learning without forgetting and bootstrapping capacities. However, as generality scales, more advanced testing is likely necessary. Metrics for testing knowledge preservation are novel, first introduced by the author in prior works of [70].

As described, it is also important for a knowledge preserving model to do so within resource constraints. Here, the evolutionary algorithm is used to employ a bottom-up and top-down optimization method for preserving knowledge within resource constraints. Curating a bottom-up novelty process should be done in a way that preserves historic learning. If the novelty mechanism, often performed by mutations, transitions the overall global state rapidly, historical information is changed in such a way that a recurrent success metric wouldn't have time to select for broad

adaptability. Another key consideration for the bottom-up process is where mutations occur. The global state should contain components that retain information from prior states. If mutations occur throughout the global state simultaneously, the ability for the system to re-introduce a particular configuration of a local component, for the sake of returning to a prior task, is degraded. Likewise, if the success metric is too recurrently focused that it penalizes most novelty, the system won't be able to adapt to novelty introduced by the other evolving components. Thus, there is a critical balance between the bottom-up and top-down mechanisms. Here, bottom-up processes are controlled by minimizing the loss metric of the model and crossover and mutation hyperparameters in the evolutionary algorithm.

Learning without catastrophic forgetting is typically measured by performance on a sequence of tasks. Here, we propose that a recurrent sequence of sub-environments can be used to assess knowledge preservation. Where task A is performed, then task B then task A again. However, in order to test whether higher-order concepts can be encoded in the pursuit of multiplexing information, as occurs in the human brain, the tasks chosen share the common concept of depth perception. Depth can be ascertained through various sense modalities. In this work, depth can be perceived by the artificial agent visually, through distance via position, through auditory clues, or through a distance measure via raycast. One main goal of the study is to determine if the concept of depth can be encoded and used across lower-level sensory representations.

The metric used in this study for measuring knowledge preservation how well the model preforms on the current task. Thus, this measure can be related to the accuracy of a single task or across a sequence tasks. Here, accuracy is a measure by the reward received by the agent in each tasks. For single tasks and sequences of tasks, as well as the type of model used, accuracy can be measured differently. For PPO, SAC, SAC+LSTM, and DQN models on single tasks, a single-task accuracy meassure is used. This measure is collected after the total number of iterations for each the

final episode rewards. An average is then taken across all iterations. This results in the rewards at the end of each episode, during which the model is updated. The advantage here, is that the speed at which the model learns is somewhat inherently captured, since the reward is gathered across the entire training. For the DQN+GA, DQN+GA-RC, and DQN-GA, A total episode reward is calculated by adding the rewards incurred during and episode. An average across all episodes is taken to result in the sequence-task accuracy measure. This measure is similar to the single-task accuracy with the exception that it is measured across the sequence of tasks. A special accuracy measure is given in addition to the previous sequence task accuracy during testing. A total episode reward is calculated by adding the rewards incurred during and episode. An average is taken of the last five total episode rewards. From this reward list, the last value is taken as the prime reward. This resulting prime reward is essentially the final training model reward for a particular individual model. An average is taken across model iterations, resulting in the prime accuracy measure. The advantage of this measure, is that the final reward is captured which is indicative of how well the training procedure performed. Higher reward values indicate better performance on the task.

Below is a table of the accuracy measures used in this work.

Name	Single-task Accuracy	Sequence-task Accuracy	Prime-task Accuracy
Models	PPO, SAC, SAC+LSTM, DQN	DQN+GA, DQN+GA-RC, DQN-GA	DQN+GA, DQN+GA-RC, DQN-GA
Train/Test	Training and Testing on Single Tasks	Training on Sequence of Tasks	Testing on Sequence of Tasks

Table 3.3: Rewards in each environment and model pair can be calculated according to single-task or the sequence-task objective and type of training/testing metric desired.

Research Design

In each of the 4 sub-environments for the fox base environment an agent is fitted with the respective input capturing device (i.e. sensory organ) and navigates the environment to collect input data for the corresponding model to use as information

to produce an actionable output. In each environment, there are two sources, A and B, or rather cylinder and cube. The goal of the agent is to move to the closest source, moving to the incorrect source results in a negative outcome, moving to the correct target results in a positive outcome. The agent is always placed in the scene at the same fixed geographical location while the location of A and B are randomized. Once the agent gets within a predetermined range of a source, a reward of 1 unit is given for the target or -1 unit for the non-target and the environment is terminated and the prediction of which agent is closer is captured. In each environment, a time-step consists of one navigation decision, rendering a new scene depending on how the agent moves around. This process is iterated a number of times in order for the agent to learn the environment effectively, using the total reward as information to update the neural architecture.

Since the space of all possible modular architectures is too large to investigate by design, the authors propose an evolutionary strategy to explore emergent architectures. Resulting topologies are selected by a fitness function that rewards models on the basis of performance in three metrics: computational execution time, accuracy on the task, and resource requirements. Topologies which are faster at learning the tasks, produce more accurate predictions, or require less resources are more likely to be rewarded with the ability to pass on their genes to the next generation. Genes describe the network topologies in terms of the number of nodes, how those nodes are connected in the form of the weight matrix of each layer in the network. The weight, or connection value, is learned through traditional back propagation techniques, specified by the aforementioned connections. However, the ability to drop weights, or add weights, is determined by the resource constraint hyperparameter. Hyperparameter tuning is used to configure the optimal number of add and drop neurons within the fixed resource constraint metric. Figure Two describes the overall flow in the novel model proposed.

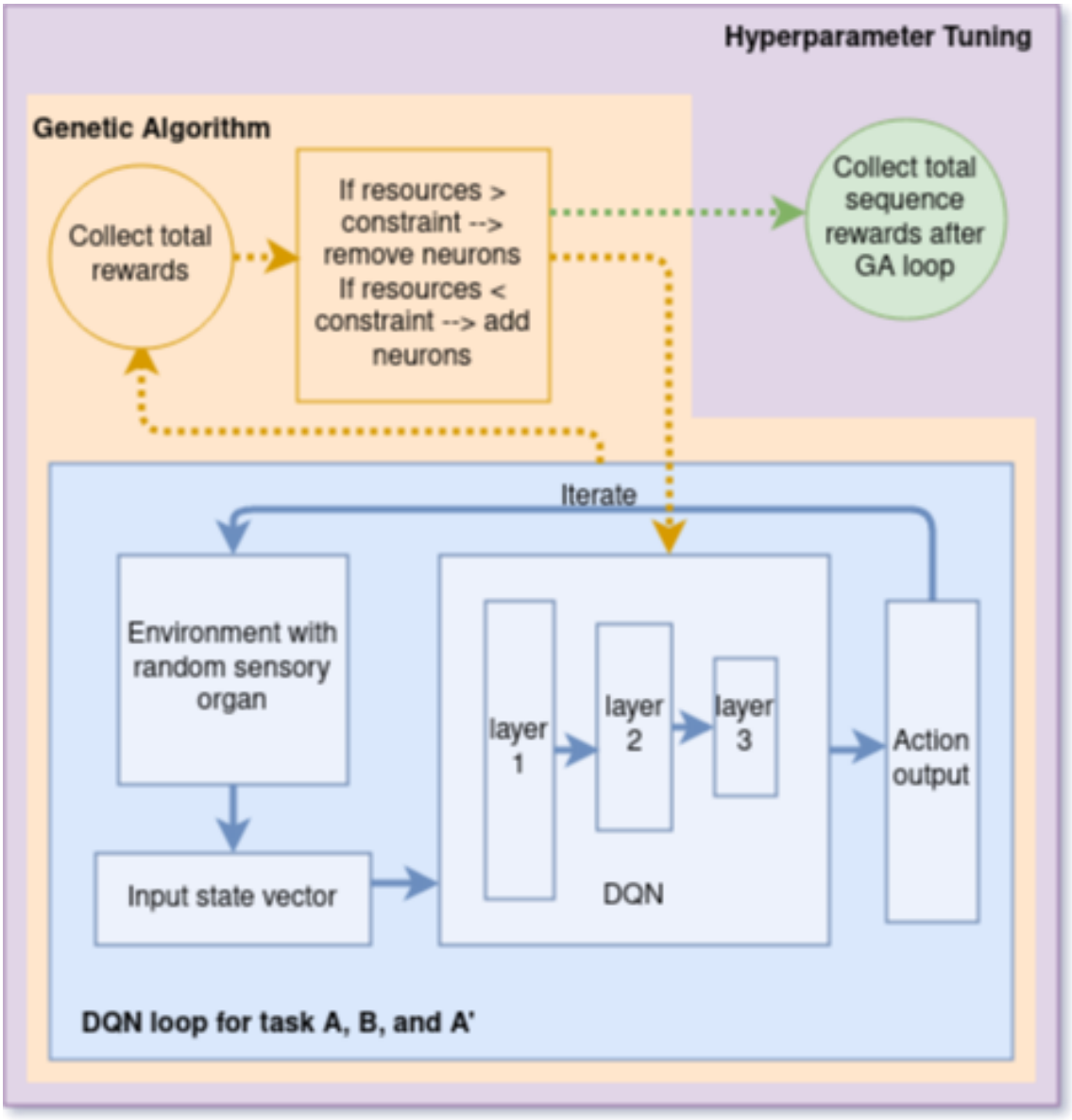


Figure 3.4: A loop iterates through different sub-environments with different sensory organs for DQN training until task A, B and repeat task A (i.e. A') have been completed. The total average reward is collected and used as the fitness function for a genetic algorithm to remove or add weights in the DQN layers at random, percentages calculated by hyperparameter tuning. Final results are then collected after one recombination, mutation, and implementation loop of the genetic algorithm.

Procedures

First, each environment was constructed and trained using PPO, SAC, and SAC+LSTM models. Results were recorded with Tensorflow's Tensorboard program [163], which tracked single-task accuracy and loss. This step determines the viability of each environment in the reinforcement learning, state-action loop. In this step, each model is trained on one environment at a time. Between training sessions, each model is re-created from scratch. These metrics serve as control measures for the DQN on each environment as a single task. The DQN was trained using the hyperparameters derived from the procedures below.

In each of the six environments, the input signal (e.g. sensory information) needs to be of the same length. This allows the same brain to be recycled in sequences of environments, which is necessary for comparing models equally. Each DQN based model is constructed with this constraint in mind by forming the input vector to be of a predefined length. The largest length of the four fox environments was that of the auditory organ, at a length of 40,000. Thus, all inputs for DQN, DQN+GA, DQN+GA-RA, and DQN-GA were padded to 40,000 dimensions. Since the audio model is the largest, in terms of real-valued input state, a hyperparameter sweep was performed according to the procedure below. To determine the impact of padding, the DQN was trained on each environment with padding to 40,000 and with an alternative padding to the visual signal of 1,752. In the alternative padding, compresses the audio signal by stacking the audio images into a single images, linearizing it, and then padding to the visual signal length.

To determine the DQN hyperparameters, the audio environment was used for a hyperparameter sweep using Weights and Biases hyperparameter random search process. Here, the Weights and Biases program was used to run a random search on the DQN hyperparameters in order to determine optimal values according to maximiz-

ing the total average reward (for a list of hyperparameters tuned and tuning results, see Appendix D and E). The model has 10 hyperparameters with three values each, resulting in 59049 possible combinations. The runtime to calculate all possible combinations was outside the scope of this work. Therefore, only a subset of hyperparameter tuning runs were used to calculate the optimal hyperparameter values. 280 runs were taken into account, although the selection of the hyperparameters is optimized according to the Weights and Biases program's protocol which optimizes for most viable hyperparameter values (i.e. an intelligent search of values instead of random search) [147]. From here the mode value was collected from the total values sorted by maximum reward for all average reward values less than one and greater than -10. The mode was then calculated for all average rewards less than one and greater than -5. The mode was then calculated for average rewards sorted by reward and then by runtime for the rewards for the first 100 runs, excluding runs that received an average reward of one. The mode was then calculated for the average reward sorted by reward and then sorted by run time for reward values between 2 and 10.

From these four mode calculations, the mode was used as final hyperparameter values. This procedure was used because the model procedures allow a random chance of spawning into the environment and automatically hitting the target without any learning or computation of the input. In short, by pure chance, the agent could spawn into the reward or close enough to it that the first random action it chooses gets the agent to the target. In order to account for this, we remove the low runtime and high average rewards models The environment training process was then repeated with the DQN model for which weights and biases log single-task accuracy. These hyperparameters were used throughout all future DQN and DQN+GA training and testing.

A hyperparameter search was performed to investigate the resource constraints by adding four new hyperparameters: number of crossover weights, layer one mutation

rate, layer two mutation rate, and layer three mutation rate. For hyperparameter tuning of the DQN+GA, the previous hyperparameters were frozen so that they were left out from the new hyperparameter tuning. The total resource constraint metric was set to 80%, such that if the number of total zeros in the DQN model is over 80%, the GA is instructed to remove weights. If the total of zero weights is below 80%, the GA is instructed to add random weights. Which layer to add/remove in crossover is chosen at random. Which weights to add/remove in mutation is chosen at random. Yet, once a layer is chosen to add or remove weights to in crossover and mutation, the percentage of weights in that layer is chosen according to a hyperparameter. This hyperparameter is optimized according to the aforementioned Weights and Biases hyperparameter tuning random search process.

Here, there are 16 hyperparemeters for the DQN+GA model. Six of these hyperparameters are additional for the GA model, of which, four are tuned using hyperparameter optimization described above. Each tunable hyperparameter has three values each, resulting in 81 possible combinations. Testing all possible combinations was outside of the scope of this work, thus only 14 runs were considered for hyperparameter tuning during this phase. Results for hyperparameter tunning were concluded using run that received the best final reward accuracy. These hyperparameters were used for further training and testing of the DQN+GA, DQN+GA-RC, and DQN-GA models.

In order to evaluate a model's ability to preserve knowledge during a sequence of tasks, two sub-environments out of the four fox environments (visual-fox, audio-fox, position-fox, and raycast-fox) were randomly chosen, reffered to as depth-sequence tasks. The agent is evaluated first on task A, then task B, then task A again. Between tasks, the model weights are saved and re-loaded for the next tasks. Results are collected using the Weights and Biases application. In order to asses if having a shared concept is useful in storing information for later learning, another sequence

task was constructed in which task B is always the match environment while tasks A is chosen at random from the four fox sensory environments, reffered to as mixed-sequence task.

The DQN+GA, DQN+GA-RC, and DQN-GA models were then depth sequence task three times. Further results were collected for the DQN+GA and DQN-GA by training both models on the mixed sequence for three separate iterations. The last trained weights for each DQN+GA and DQN-GA model were then chosen to preform a test. In this test, the weights are loaded and evaluated on a each sensory environment (excluding the none environment). This test is repeated for each of the 10 individuals in each of the model's population for the mixed and depth sequence task separately. The population count was chosen as a hyperparameter in the DQN+GA model based on compute resource availability.

In order to determine how model weights change during training. The DQN+GA and DQN-GA are trained for three evolution iterations. At this point, models are saved between task A, task B, and the recurrent task A. Another evolution iteration is performed for which the resultant model weights are saved between each task similarly. Model weights are then loaded into arrays, normalized, and the absolute value difference between each model weights between the third and fourth evolutionary iterations is recorded.

Finally, a hyperparameter tuning according to previous DQN+GA hyperparameter tuning protocol is repeated for the DQN+GA model in which all other hyperparameters are frozen for various RC weights for which a sequence-task accuracy is recorded.

Data Analysis

Research Question One

How does changing architecture preserve knowledge?

Hypothesis 1 Architectures that self-modify neuron weights in a way that manages the trade-off between existing information and new information in relation to sequential specific goals preserve knowledge.

This hypothesis can be tested by comparing the accuracy between the DQN+GA model, the DQN+GA-RC, and the DQN -GA models on the sequential depth objective. Furthermore, an analysis of the accuracy between DQN+GA models with various levels of crossover and mutation rates can test how resources might be managed between layers. The rationale here is that the stability-plasticity trade-off is directly managed by the resource constraint measure and the percentage of neurons added or removed in each layer. Since information propagates from layer to layer, the trade-off between which layer can afford more node modification, either mutation or crossover, is also a viable statistical analysis. The limitation of this measure is that the hyperparameters are not tuned for each set-up individually. The hyperparameters were tuned for the DQN, since it is the base model. Extra hyperparameters for the GA were tuned separately according to the procedures above.

Hypothesis 2 Architectures that self-modify by rewriting existing neuron weights are able to adapt to changing modalities by optimizing for consecutive tasks.

Hypothesis two can be analyzed by comparing the training and test accuracies between DQN+GA models and DQN-GA models for sequential tasks that share objectives (i.e. depth perception) and sequential tasks that don't share an objective. Analysis between sequences of tasks with shared concepts for models that use standard backpropagation and the DQN+GA with resource constraint feature are compared to test whether standard backpropagation methods or the DQN+GA method is better

suited for adapting to changing tasks. Here, limitations of the analysis are in the type of tasks chosen. Since reinforcement learning is used, there is still a shared procedure between the depth sequence objective and the non-depth sequence, in that the agent has to learn state-action-reward loops. To account for this, the task of non-depth was chosen to be a visual matching objective. Furthermore, the action outputs are shared, which allows switching between depth tasks and visual matching tasks possible within the same model weights.

Research Question Two

How do evolutionary pressures of resource constraints affect architectures for learning and using high-level concepts?

Hypothesis 1 Architectures with neurons which are shared across different computational abilities display more multiplexing of learned information.

Degrees of change in weights at between model checkpoints for the DQN+GA model and DQN-GA model between different tasks in the sequential depth objective during training can be compared to asses which nodes are static, being used across tasks, and which nodes are being re-learned to encode new information. In order to test if neurons are being multiplexed, we would need to see if neurons remain somewhat constant across tasks in a sequential task objective. The limitation here is that since backpropagation forces automatic weight update, all neurons will be changed by some small degree. To account for this, we take the degree of change in terms of percentage normalization where a complete change, or reversal of value, would result in a degree of one. Smaller changes in weight values result in a fraction of one. Direction of change is not accounted for, as a positive or negative weight change can be treated equally in terms of degree change and ultimately has no difference in determining which nodes are multiplexed, or more static.

Hypothesis 2 Emergent architectures from evolutionary pressures of resource con-

straints aid general learning and use of a high-level concept.

The overall accuracy in sequences of depth tasks for testing between the baseline models, DQN, and DQN+GA can be compared. As a standard measure of how evolutionary pressures of resource constraints play a role in preserving knowledge, the total reward is calculated over multiple test runs, using hyperparameters selected by the aforementioned procedures. The limitation here is that the test models are trained on a variety of environments, which may obscure the model's accuracy on a specific environment sequence. To account for this, multiple test models are performed an a variety of depth sequences.

Chapter 4

RESULTS

Model and Environment Analysis

The results in figure 4.1 describe which hyperparameters in the DQN model have the most impact on the overall single-task accuracy for the fox-audio environment. The importance measures in relationship to all runs performed, the impact on the accuracy measure, while the correlation states how increases the value of the hyperparameter will effect the accuracy value. For example, the number of nodes in layer two (FC2) is anti-correlated with the accuracy measure, producing a large red bar. This can be interpreted as, when the number of nodes increases, the accuracy decreases. During hyperparameter tuning, Weights and Biases used the random search procedure to determine the appropriate number of episodes from a possible selection of 25, 50, and 100; possible learning rate values of 0.001, 0.001, and 0.005; possible layer one number of nodes 64, 128, and 256; possible layer two number of nodes 32, 64, 128; batch sizes of 64, 128, 256; gamma values of 0.3, 0.5, 0.999; epsilon starting values of 0.7, 0.9, 0.99; epsilon ending values of 0.01, 0.05, ; epsilon decay values of 50, 100, 200; and target policy network update iteration values of 5, 10, 20. The final hyperparameters chosen were learning rate of 0.001, 64 layer one number of nodes, 32 layer two number of nodes, batch size of 256, gamma rate of 0.3, epsilon starting at 0.7 with a decay of 200 ending at 0.05, the target net updated every 5 iterations, and 25 episode iterations.

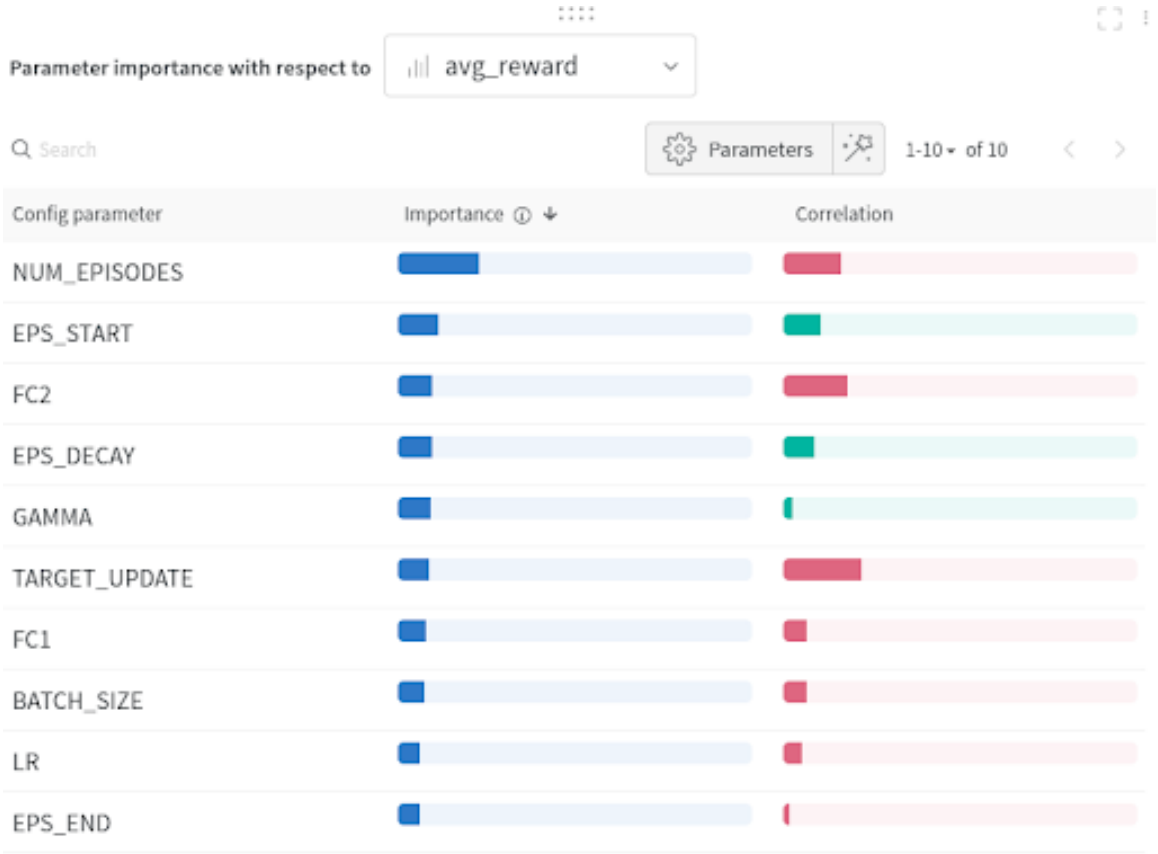


Figure 4.1: DQN Hyperparameter Importance
Each hyperparameter in the DQN is ranked in terms of importance and corresponding correlation in relation to the hyperparameter optimization of average reward of the last 5 total episode reward averages.

Figure 4.2 shows the single-task accuracy on grouped by each environment for the DQN model during training. The two-tailed P value equals 0.0411 and is statistically significant for alpha values of 0.05 for the none environment compared to the fox sensory organ environment single-task accuracies. The mean of none environments minus organ environments equals -4.02500046221. 95% confidence interval of this difference: From -7.88032033078 to -0.16968059364. Intermediate values used in calculations: t = 2.1015, df = 46, standard error of difference = 1.915. This result is captured from across six separate runs for the audio, visual, raycast, and position fox environments and compared to 24 separate runs for the none environment. The top

bar in the graph is the match environment for the DQN included for comparison. In each bar, the line represents the min and max values across the total runs for each environment. The x-axis is average single-task accuracy reward across alll runs for that environment.

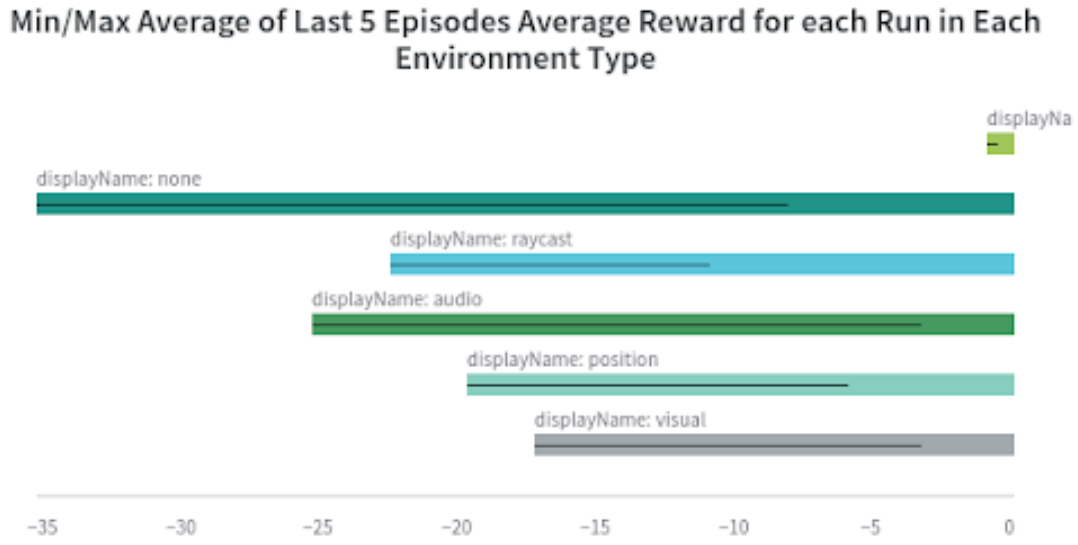


Figure 4.2: Training accuracies grouped by environment type for average of last 5 episode types. Min and max lines are included. The top bar is for the match environment. None environment vs. fox environments is statistically significant via t-test. p=0.0411.

For each of the baseline models, PPO, SAC, SAC+LSTM and DQN models, single-task accuracies were then compared for each of the six environments, as shown in figure 4.3. Here, each run for each model in each environment records the last single-task accuracy An average is then taken per model per run for that run's last five episode accuracies during training. A final average is calculated over all of runs per model per environment to produce the results shown in the figure.

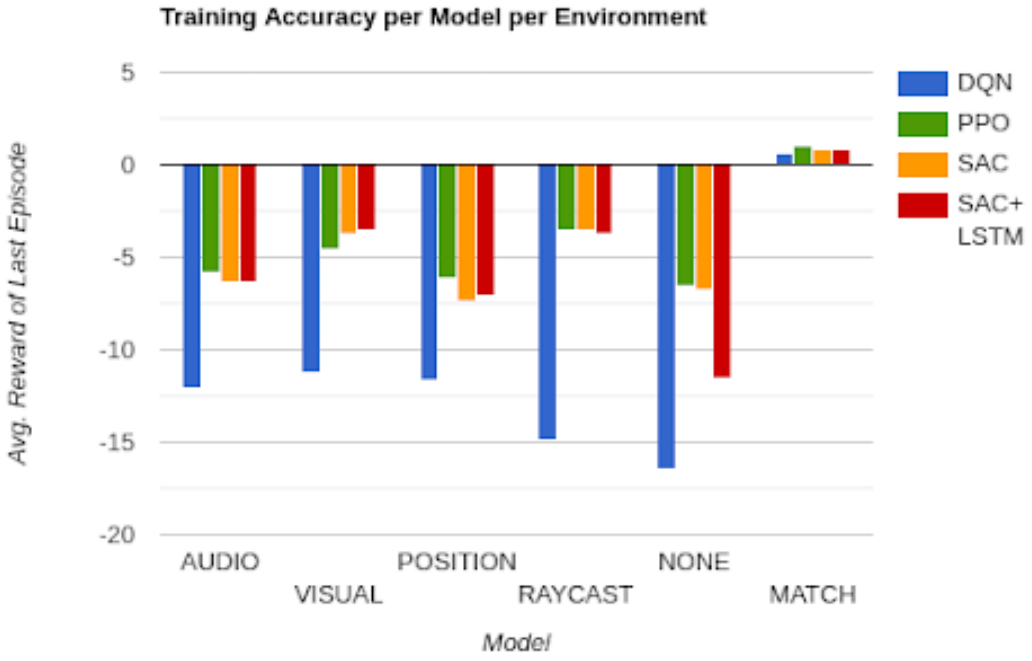


Figure 4.3: Average reward of last episode for DQN, PPO, SAC, SAC+LSTM models on each single environment.

The DQN+GA model hyperparameter tuning used the already tunned hyperparameters from the DQN with the addition of hyperparameters for the number individuals in the evolutionary algorithm population, the number of nodes for each layer to be used in crossover, the number of nodes in layer one to be used for mutation, the number of nodes to be mutated in layer two, the number of nodes in layer three to be mutated, and the number evolutionary episodes. Out of these hyperparameters, the evolutionary episodes were fixed to 5 and the number of individuals in the population were fixed to 10 to consereve compute requirements since each model run took between three and 12 hours to train during the hyperparameter sweep. Figure 4.4 shows the importance of each searchable hyperparameter in relation to average prime accuracy reward. Figure 4.5 shows the hyperparameter search across DQN+GA runs.

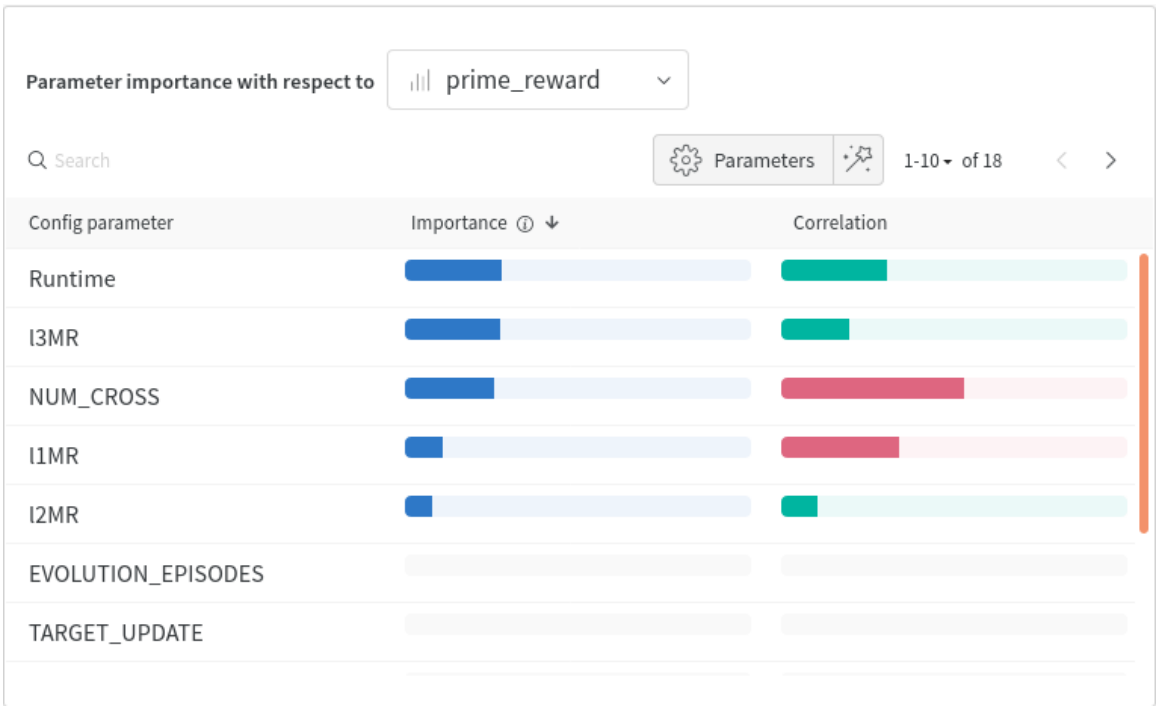


Figure 4.4: Each hyperparameter in the DQN+GA is ranked in terms of importance and corresponding correlation in relation to the hyperparameter optimization of average reward of the last 5 total episode reward averages.

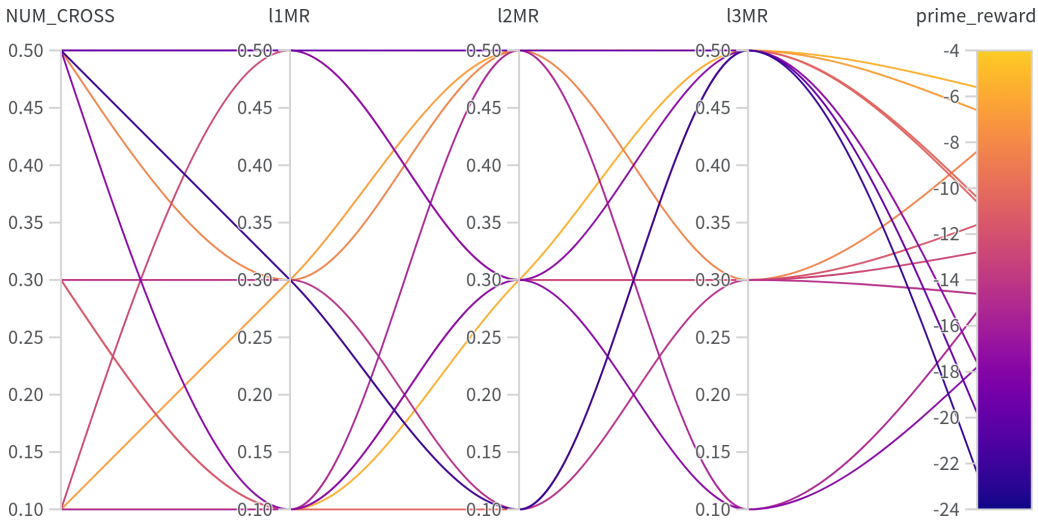


Figure 4.5: Runs across the hyperaprameter search are recorded in terms of prime accuracy.

Statistical Analysis

For all statistical analysis, an unpaired, two-tailed t-test is used, with p values less than .005 being statistically significant. The proceeding results compare different model and environment training and test with reported accuracy measures to support or deny the proposed research questions and corresponding hypotheses.

Research Question One

How does changing architecture preserve knowledge?

Hypothesis 1 Architectures that self-modify neuron weights in a way that manages the trade-off between existing information and new information in relation to sequential specific goals preserve knowledge.

Here, the training prime accuracy is reported for the DQN+GA, DQN+GA-RC, and DQN-GA, as shown in figure 4.6. The sequence-task accuracy for the DQN+GA during training on the depth tasks is -8.81, for the DQN+GA-RC is -14.73, and -18.80 for the DQN-RC. This result is calculated from the average of three separate runs per model. Between the DQN+GA and DQN-GA, each sequence-task accuracy was used in a t-test, resulting in a p-value of 0.0456. The mean of DQN+GA minus DQN-GA equals 9.98666785800. 95% confidence interval of this difference: from 4.09245857416 to 15.88087714184. Intermediate values used in calculations: t = 4.7042, df = 4, standard error of difference = 2.123. By conventional criteria, this difference is considered to be very statistically significant. Between the DQN+GA and DQN+GA-RC, each sequence-task accuracy was used in a t-test, resulting in a p-value of 0.0093. By conventional criteria, this difference is considered to be statistically significant. Confidence interval: the mean of DQN+GA minus DQN+GA-RC equals 9.30666775800. 95% confidence interval of this difference: from 0.29264323328 to 18.32069228272. Intermediate values used in calculations: t = 2.8666, df = 4,

standard error of difference = 3.247.

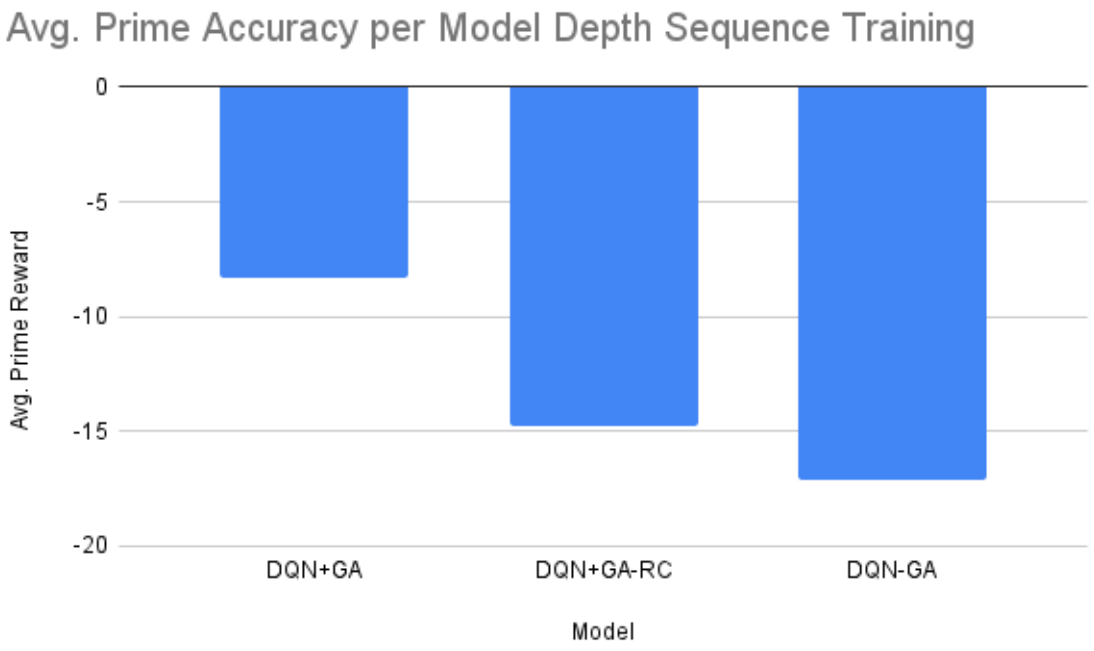


Figure 4.6: Average prime accuracy results across runs for DQN+GA, DQN+GA-RC, and DQN-GA for depth sequence training.

Hypothesis 2 Architectures that self-modify by rewriting existing neuron weights are able to adapt to changing modalities by optimizing for consecutive tasks.

Training prime accuracies are compared for the DQN+GA and DQN-GA for the mixed sequence and depth sequence tasks during training as shown in figure 4.7. The resultant value is an average across three model runs per environment for the mixed and depth sequences. A t-test comparison between the DQN+GA and DQN-GA for the mixed sequence task across three separate runs resulted in a p-value of 0.0693. By conventional criteria, this difference is considered to be not quite statistically significant. Confidence interval: the mean of DQN+GA minus DQN-GA equals 12.13333479333. 95% confidence interval of this difference: from -1.52999074279 to 25.79666032946. Intermediate values used in calculations: t = 2.4655, df = 4, standard error of difference = 4.921. A t-test comparison between the DQN+GA and

DQN-GAa for the depth sequence task during training across three separate runs was previously reported in research question one, hypothesis one, with a p-value of 0.0093. This result is considered statistically significant.

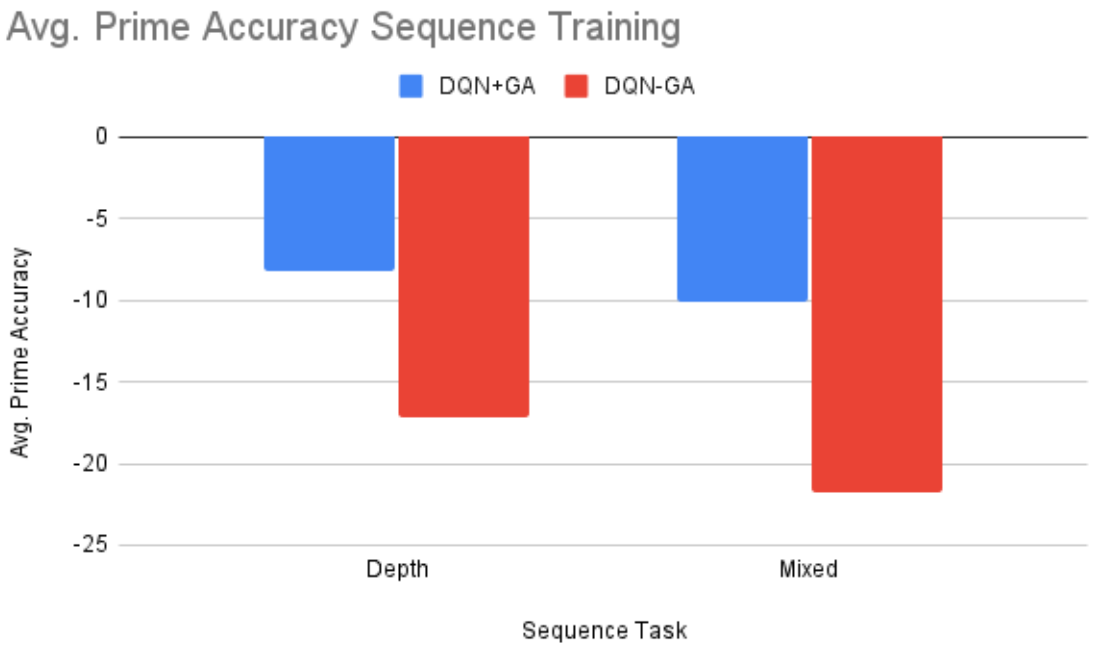


Figure 4.7: Average prime accuracy results across runs for DQN+GA and DQN-GA for depth sequence and mixed sequence training.

Test prime accuracies are compared for the DQN+GA and DQN-GA for the mixed sequence and depth sequence tasks during testing as shown in figure 4.8. The resultant value is an average of all tests per model for each environment for the mixed and depth sequences during testing. A t-test comparison between the DQN+GA and DQN-GA for the mixed sequence task across ten separate runs for the match environment resulted in a p-value of 0.8297, a p-value of 0.3831 for the raycast environment, a p-value of 0.8530 for the audio environment, a p-value of 0.0738 for the visual environment, and a p-value of 0.6168 for the position environment, none of which are considered statistically significant. A t-test comparison between the DQN+GA and DQN-GA for the depth sequence task across ten separate runs for the

match environment resulted in a p-value of 0.1753, a p-value of 0.8581 for the raycast environment, a p-value of 0.0842 for the audio environment, a p-value of 0.3907 for the visual environment, and a p-value of 0.9264 for the position environment, none of which are considered statistically significant. The test results for just the DQN-GA for each environment are recorded in Appendix section K. The table below describes the p-values per DQN+GA to DQN-GA model t-test comparison per environment 4.1.

mixed-match	0.8297
depth-match	0.1753
mixed-visual	0.0738
depth-visual	0.3907
mixed-position	0.6168
depth-position	0.9264
mixed-raycast	0.3831
depth-raycast	0.8581
mixed-audio	0.8530
depth-audio	0.0842

Table 4.1: P-values for Depth and Mixed Test DQN+GA and DQN-GA.

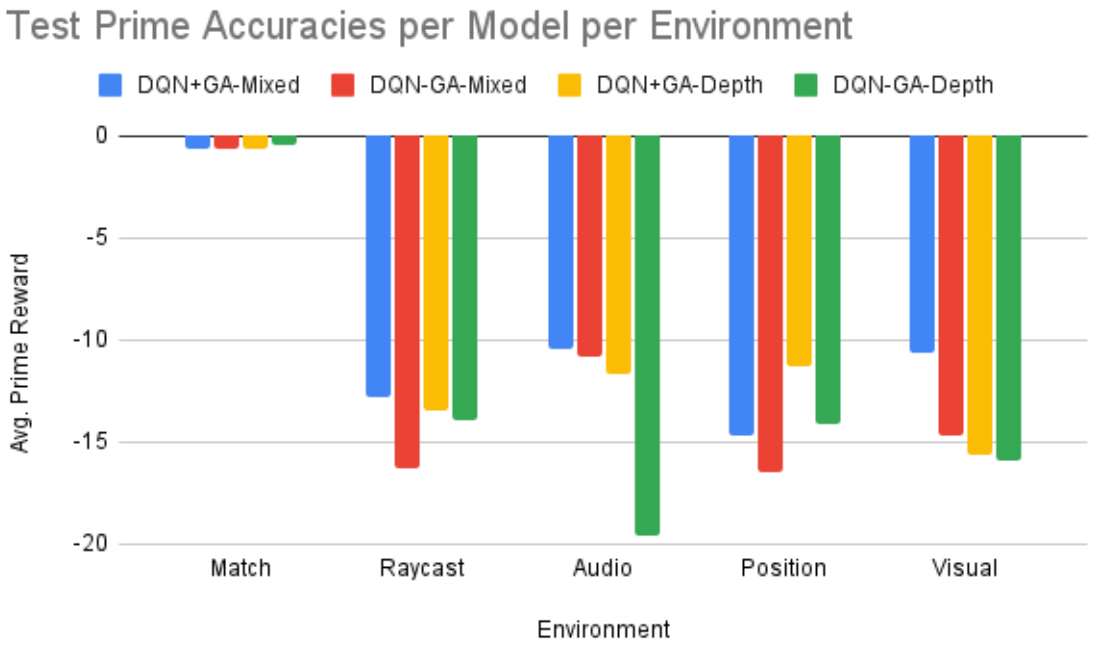


Figure 4.8: Average prime accuracy results across runs for DQN+GA and DQN-GA for depth sequence and mixed sequence test.

Research Question Two

How do evolutionary pressures of resource constraints affect architectures for learning and using high-level concepts?

Hypothesis 1 Architectures with neurons which are shared across different computational abilities display more multiplexing of learned information.

Figure 4.9 records results from the percentage of weight change between between task A and recurrent task A between evolutionary steps three and four for the DQN+GA per model layer. For the DQN+GA, layer one change resulted in 84.9318%, layer two in 1.944%, and layer three in %0.2626. This result is collected by calculating the absolute value of change between the two normalized weights and summing across each individual in the models population in relation to the total weights in each layer as a percentage of the total weight change.

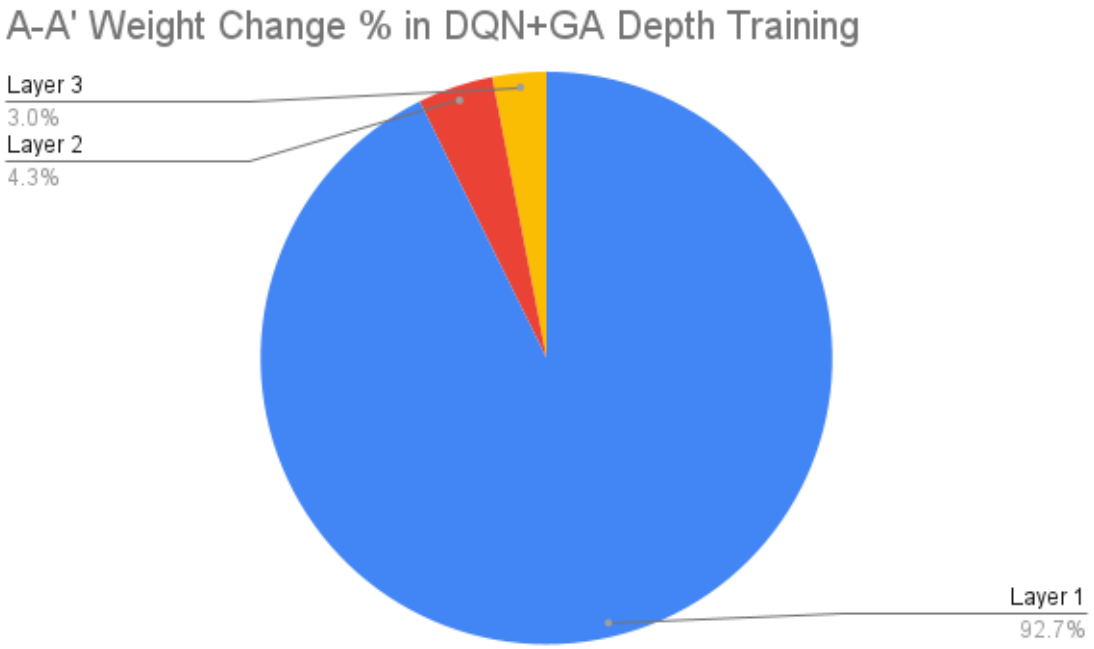


Figure 4.9: Percentage of change between model weights between tasks in evolutionary iterations per model.

Hypothesis 2 Emergent architectures from evolutionary pressures of resource constraints aid general learning and use of a high-level concept.

Results are recorded for test prime accuracy for the DQN+GA and DQN-GA depth sequence tasks. These results, shown in figure 4.10 are taken from research question one, hypothesis two for direct comparison here. A t-test comparison between the DQN+GA and DQN-GA for the depth sequence task across ten separate runs for the match environment resulted in a p-value of 0.1753, a p-value of 0.8581 for the raycast environment, a p-value of 0.0842 for the audio environment, a p-value of 0.3907 for the visual environment, and a p-value of 0.9264 for the position environment, none of which are considered statistically significant.

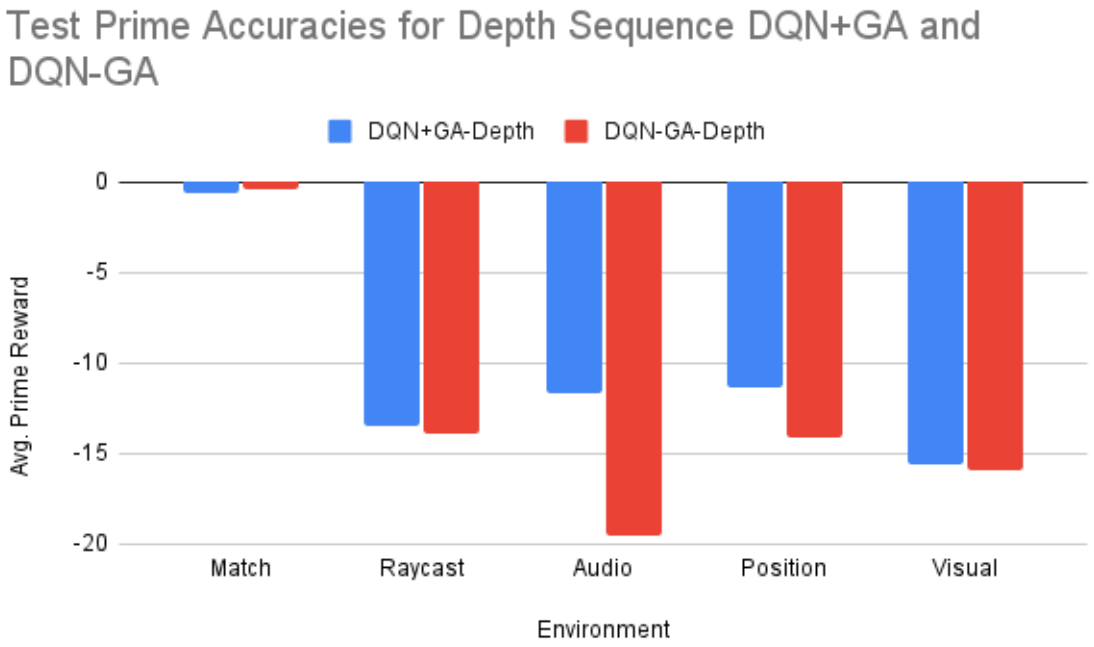


Figure 4.10: Average prime accuracy results for depth sequence task for DQN+GA and DQN-GA models.

Appendix section L shows the total results recorded for each research question and corresponding hypothesis.

Chapter 5

DISCUSSION

Summary

Evolutionary algorithms aid deep learning models in preserving knowledge during training on sequences of tasks. Some of this effect is transferable to testing on completely novel tasks. Resource constrained evolutionary algorithms outperform those without resource constraints, which supports the ideas laid out in the recommendation architecture theory of cognition by Andrew Coward [99].

Conclusions

Importance of Layer Learning

DQN Hyperparameter Tuning

Results form the DQN hyperparameter tuning suggests that increasing the number of episodes during training does not increase the overall average accuracy. One reason for this is that the model weights could be overfitting to each task as commonly observed in deep learning models [164]. For the epsilon, gamma, batch size, and learning rate related hyperparameters, the hyperameter search tunes the epsilon metrics to the overall audio task as expected. Interestingly, the hyperparameter search shows that the number of nodes in the fully connected layer two is more important than the number of nodes in the first fully connected layer. Both hyperparameters are anti-correlated. This result suggests that more nodes does not increase the average reward, which implies that conserved resources may increase learning across variances in a single-task objective. In addition, these results suggest that the nodes in layer

two, which are thought to encode higher-concept features from layer one, are more important; that the fewer nodes that layer one projects into creates better average learning, which expounds upon the current research of hierarchical networks encoding concepts of concepts from layer to layer [165]. This particular result suggests that higher degrees of multiplexing of shared lower-level features in layer one may lead to higher average rewards on the current task. Finally, the target update hyperaparameter results show that the more frequently the target policy in the DQN is updated, the higher average reward received for the task. This result is expected since updating the model results in advancing the stored learning protocol of the DQN. This further suggest that the task can be learned more effectively with incremental learning. This result is supported by the theory that biological brains require sleep as a sort of update feature during learning, as suggested in the recommendation architecture [99].

DQN+GA Hyperparameter Tuning

Results from the DQN+GA hyperaparameter tuning shows that the layer three mutation rate is more important than layer one, followed importance for layer two, in regards to the prime accuracy. Furthermore, layer three mutation rate is correlated, meaning the more mutation that occurs in layer three, the increase in prime accuracy. This suggests that layer three benefits from random changes; that variability in higher-order concepts can benefit the learning trajectory. Layer one is anti-correlated, suggesting that the more weights changed in layer one, the worse performance measured. This result is consistent with the DQN hyperparameter search, that layer one is encoding lower-level concepts that are then interpreted by layer two and then layer three for behavioral output. Layer two weights are again correlated, further suggesting that higher-order concepts make use of multiplexed information storage. This result is consistent with the recommendation architecture that some features are learned by the cortex which are later interpreted by the basal ganglia for behavioral

output [99]. Interestingly, number of nodes involved in crossover is less important than the number of layer three mutation rate, but more important then layer one and two mutation rate. Since the crossover hyperaprameter involves all layers, this suggests that layer three, with the most higher-level concepts is the most imperative to learning performance. Furthermore, since crossover is anti-correlated, this suggests that transfer learning has a limit of effectiveness; that some degree of transferring weight is helpful, but too much transfer can result in negative learning performance.

Environment and Model Viability

Single-task Accuracy DQN

The results from the DQN model single-task accuracy for each environment show that with the sensory organ for the fox environments, the model is statistically significantly more likely to learn the task. This result validates the experimental design in that, during training, the DQN benefits from sensory state inputs. The drawback of comparing the match environment is that the match environment does not involve a small negative reward per step as the fox environment does. Thus, it is not an accurate comparison, since the total negative reward is automatically smaller than the fox environments. It is important to take note of this distinction for further discussion; that the mixed environment cannot be directly compared to other environments due to the difference in reward curriculum.

Single-task Accuracy for Baseline Models

For each baseline model, the training accuracy per environment is provided by figure 4.3. These results show variability in the best model suited for the single task environment. Overall, the DQN preforms worse than the state-of-the-art models provided. This is likely due to the increased number of weight update iterations for PPO, SAC, and SAC+LSTM models compared to DQN. However, these results suggest that not only is each environment learnable, but also that the environments

can be ranked in difficulty. The match environment is the easiest, followed by visual, position, audio, raycast, and the none environments.

Training with Resource Constraints in Evolutionary Loops

In figure 4.6, the results show that the DQN with the evolutionary algorithm perform significantly better than the DQN without mutation and crossover, suggesting that these features help manage stored information in such a way that the model can learn sequences of tasks. At the very least, the comparison supports the validity of the experimental design in that the advantage of the DDQN+GA is due to the management of weights by mutation and crossover rather than more iterations, since the DQN-GA is still iterating through evolutionary steps. More interestingly is the result that the DQN with the evolutionary algorithm, but without resource constraints performs significantly worse, suggesting that resource constraints effect information learning in the DQN model on the sequence task. This supports the theory laid out in the recommendation architecture that evolutionary pressures of resource constraints can enforce information encodings that be used across a variety of contexts [23]. However, this finding is somewhat counterintuitive to the those in the deep learning community that subscribe to the ideas afformented in [73]; that larger networks capable of processing more data will lead to learning as the human brain does.

Mixed Sequences and Depth Sequences

Figure 4.7 further suggests that the DQN+GA is significantly better at learning recurrent sequences of tasks, even when the model learns a second task that is unrelated to features in the first (and recurrent) task. In moving from training to testing, figure 4.8 shows that on average, the DQN+GA performs better than the DQN-GA for mixed and depth sequence trained models across all test environments with the

exception of the match environment. This further supports the use of evolutionary algorithms with the DQN for continual learning.

For DQN+GA models, the depth sequence training provided better, on average, test results than the mixed sequence training for the match and position environments. This result is adversarial to the results found in training that suggest the evolutionary algorithm helps the DQN learn and conserve information in sequences of task. However, the results are not significant and are likely due to the difficulty of the task as well as random assignment. Here, match and position are the easiest tasks. During depth sequence training, the model is trained on two of the four fox environments and model weights are saved. The trained model is then used for the test procedure on one of the four fox environments at random. Thus, it is possible that an artifact of the test is that the models for the mixed and depth were trained on different fox environments. For example, if the depth training occurred with a more difficult, such as audio, while the mixed training occurred with an easier task, such as visual, the depth model could be more overfit to the model weights that afforded the nuances of the more difficult tasks rather than general features across tasks. In order to test this conjecture and find these results conclusive, more test with a larger sample size should be performed, further supported by the information in table 4.1. For the DQN-GA models, the results are similar that only three environments, match, raycast, and position perform better with depth sequence training. This further supports that the test results shown here could be influenced by an artifacts since the evolutionary algorithm has been removed, which results in the models being re-trained after each task.

Model Weight Change

In figure 4.9 the DQN+GA model, from evolutionary iteration three to four, the weights in the model for each layer change similar for the results provided by the

hyperparameter tuning, figure 4.4. Although layer two resulted in being least important for prime accuracy during training, layer one has the majority of weight changes, even though layer two and layer three have higher mutation rates. This suggests that the core feature learning, before hierarchical feature encoding, is done by layer one; which is consistent with current literature. Layer three is most important in relation to accuracy, and has the fewest percent of change, even though it has the highest mutation rate. This suggest that resources here are more likely to be conserved for encoding higher-order concepts, which is further supported by the small degree of change in layer two.

Evolutionary Approach to Learning

Finally, the test prime accuracies recorded by figure 4.10 shows that the DQN+GA are on average more accurate than the DQN-GA, which further suggests that evolutionary iterations during training on reccurent sequential tasks perserve knowledge in such a way that the model is better at adapting to new environments, even not trained on, than models without the evolutionary algorithm. Increasing sample size would likely result in increased significance.

Hypothesis Analysis

The results of the work proposed here research question one of how does architecture preserve knowledge, hypothesis one in that architectures that self-modify neuron weights in a way that manages the trade-off between existing information and new information in relation to sequential specific goals preserve knowledge. This hypothesis is supported by results from figure 4.6 in that the DQN with mutation, crossover, and resource constraints outperforms models without those weight managing features on a recurrent tasks after learning a new task. Hypothesis two states that architectures that self-modify by rewriting existing neuron weights are able to adapt to

changing modalities by optimizing for consecutive tasks is not supported in that figure 4.7 significantly shows that the DQN-GA, which is prone to more rewriting and less protection of the evolutionary crossover feature of the DQN+GA, performs worse on returning to a task after learning a prior task during training. Thus, resource question one can be partially answered in the work proposed here that evolutionary learning can preserve knowledge in continual learning environments for deep-learning based models and that resource constraints enforce hierarchical learning to be more effective across tasks learning.

In regards to research question two of how do evolutionary pressures of resource constraints affect architectures for learning and using high-level concepts, hypothesis one, which states architectures with neurons which are shared across different computational abilities display more multiplexing of learned information, is supported by figure 4.9 in that lower level layers in the evolutionary resource constrained models are more susceptible to change while higher order concepts are more likely to be preserved in higher layers. Conserved weights suggest sharing across environments since the higher layers have higher mutation rates. While hypothesis two, which states emergent architectures from evolutionary pressures of resource constraints aid general learning and use of a high-level concept, is partially supported by figure 4.10 which suggests that the emergent architectures during training resultant from the evolutionary weight management features preserve knowledge better since they are able to, on average, perform completely novel tasks or return to previously learned tasks after learning new tasks better than models with static architectures. Here, the DQN-GA has a static architecture because new connections between weights can't be turned on and although they can be turned off by zeroing, they can't be turned on again afterwards, all of which the evolutionary algorithm provides. Thus the DQN+GA results in more dynamic emergent weight architectures that are no longer fully connected. Overall, the research question of how do evolutionary pressures of resource constraints

affect architectures for learning and using high-level concepts was investigated in this work by comparing the DQN+GA and DQN-GA and can be partially answered in that it is likely resource constraints are encoding higher order concepts that can be used for multiplexing across novel tasks or returning to prior tasks.

Limitations

There are several challenges in constructing experiments to test the deliberation presented here. First, it is difficult to design fitness functions which escape goal optimization. With limited time to train models, a limited set of systems can be used. It is not yet clear how many cycles of novelty and evaluation are needed to endow a system with generally adaptive properties that could be applied to a plethora of goal-directed behavior. For example, if a handful of tasks are given to the system to perform in recurrent sequences, it would be difficult to draw conclusions that superseded the correlation of those results into a causal manner. However, similar reasoning can be applied to any reductionist experiment. Many different configurations of the system would need to be tested to tease out the relationship between the optimization goal and underlying mechanisms.

Secondly, the coupling of learning architecture, tasks/data, and embodiment is more demanding than most DNNs in computation and time. Although, it should be noted that the overall system may require more computational resources, the resulting encoded architectures may be extractable; that once trained, many of the existing environments and embodiments could be ablated since the model could employ its preserved knowledge in future tasks. Furthermore, the particular pairing of embodiment, architecture, and tasks is not trivial. The design must consider the feasibility of each component working in conjunction with the other.

If one designed an environment that relied upon a particular behavior for which the model had no means of producing, the system would instantaneously become in-

tractable (e.g. expecting a sphere to climb a hill when the applied forces to the sphere can't exceed gravitational forces). Furthermore, the system is limited in requiring the same input and output vector sizes. This approach, indicative of all deep learning, severely limits any deep learning based approach for tackling general learning. More specifically, comparing accuracies is only somewhat fruitful since environments with more negative rewards can incur average accuracy results that are skewed when compared to those that, by chance, have higher average accuracy rewards.

Final considerations revolve around initializing components and minimizing confounding variables. Designing an appropriate starting point for each component is difficult in that the design implementation is biased by the operator. Each hyperparameter has to be in part chosen by the operator, even with hyperparameter search. Since there is too much history in natural evolution, it is difficult to pinpoint and construct a computational model that can minimize bias whilst curating a similar evolutionary path to biological life. The resultant model will most likely have many hyperparameters, each affecting the overall dynamics of the system. The author suspects early work will involve more simple approaches that try to capture relevant foundational characteristics to inform later work.

Thus, early models that show even small degrees of knowledge preservation would be beneficial in this space. The goal of using GA characteristics in designing knowledge preservation algorithms is to produce systems which can lead to broad intelligence. Adaptive stability-plasticity in GA models affords a mechanism of introducing novelty while preserving prior learning. A balance between the top-down success metric (sequence training optimization) which evaluates the novelty producing bottom-up mechanism (mutation, crossover, and weight optimization) must be taken into account. Fitness functions which enforce adaptability in recurrent sequences, through resource constraint metrics, of tasks are likely to conserve resources given the organization of local components has characteristics that lead to generalization.

Recommendations for Future Research

Direct future research from this study should expound upon the impact of resource constraints in relation to the mutation and crossover rates. Furthermore, studies which investigate how shared concepts across different input and output sizes would support to the results found here. For example, completely different task unbound by reinforcement learning environments could provide a modality of multi-task learning that gives relevant insight into general learning emerging from deep learning approaches. In addition, studies could investigate how much continual learning can be preserved across task sequence lengths.

The work provided here demonstrates that evolutionary algorithms in conjunction with traditional deep learning approaches can overcome some of the limitations of catastrophic forgetting. More importantly, resource constraints can be used to guide models into encoding higher-level concepts. Thus, it would be pertinent to investigate alternative models which employ variations of the resource constraint metrics. In addition, evolutionary algorithms in conjunction with deep learning models are and emerging technique. Amongst these approaches, investigations of how resource conservation plays a role in the overall learning with varying mutation and crossover rates will likely result in important discoveries. Approaches such as these would support the theory in the recommendation architecture that evolutionary pressures of resource constraints result in learning architectures that can encode information for later multiplexing, ultimately leading to general intelligence.

APPENDIX

A. Default configurations for proximal policy optimization hyperparameters.

PPO HYPERPARAMETERS
trainer_type: ppo
hyperparameters:
batch_size: 512
buffer_size: 10240
learning_rate: 0.0003
beta: 5e-05
epsilon: 0.2
lambd: 0.9
num_epoch: 10
learning_rate_schedule: linear
network_settings:
normalize: True
hidden_units: 128
num_layers: 2
vis_encode_type: simple
memory: None
goal_conditioning_type: hyper
reward_signals:
extrinsic:
gamma: 0.9
strength: 1.0
network_settings:
normalize: False
hidden_units: 128
num_layers: 2
vis_encode_type: simple
memory: None
goal_conditioning_type: hyper
init_path: None
keep_checkpoints: 5
checkpoint_interval: 500000
max_steps: 500000
time_horizon: 1000
summary_freq: 12000
threaded: True
self_play: None
behavioral_cloning: None

B. Default configurations for soft actor-critic hyperparameters.

SAC HYPERPARAMETERS
trainer_type: sac
hyperparameters:
learning_rate: 0.0003
learning_rate_schedule: constant
batch_size: 32
buffer_size: 50000
buffer_init_steps: 0
tau: 0.005
steps_per_update: 1.0
save_replay_buffer: False
init_entcoef: 1.0
reward_signal_steps_per_update: 1.0
network_settings:
normalize: True
hidden_units: 128
num_layers: 2
vis_encode_type: simple
memory: None
goal_conditioning_type: hyper
reward_signals:
extrinsic:
gamma: 0.99
strength: 1.0
network_settings:
normalize: False
hidden_units: 128
num_layers: 2
vis_encode_type: simple
memory: None
goal_conditioning_type: hyper
init_path: None
keep_checkpoints: 5
checkpoint_interval: 500000
max_steps: 500000
time_horizon: 1000
summary_freq: 12000
threaded: True
self_play: None
behavioral_cloning: None

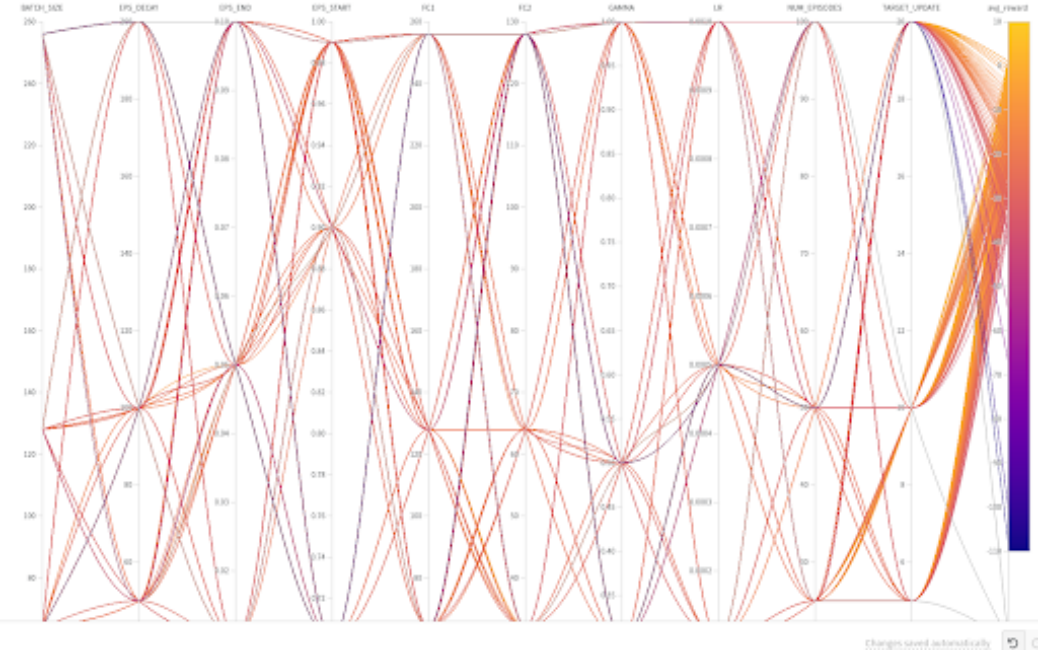
C. Default configurations for soft actor-critic with LSTM hyperparameters.

SAC + LSTM HYPERPARAMETERS
trainer_type: sac
hyperparameters:
learning_rate: 0.0003
learning_rate_schedule: constant
batch_size: 32
buffer_size: 50000
buffer_init_steps: 0
tau: 0.005
steps_per_update: 1.0
save_replay_buffer: False
init_entcoef: 1.0
reward_signal_steps_per_update: 1.0
network_settings:
normalize: True
hidden_units: 128
num_layers: 2
vis_encode_type: simple
memory:
sequence_length: 32
memory_size: 128
goal_conditioning_type: hyper
reward_signals:
extrinsic:
gamma: 0.99
strength: 1.0
network_settings:
normalize: False
hidden_units: 128
num_layers: 2
vis_encode_type: simple
memory: None
goal_conditioning_type: hyper
init_path: None
keep_checkpoints: 5
checkpoint_interval: 500000
max_steps: 500000
time_horizon: 1000
summary_freq: 12000
threaded: True
self_play: None
behavioral_cloning: None

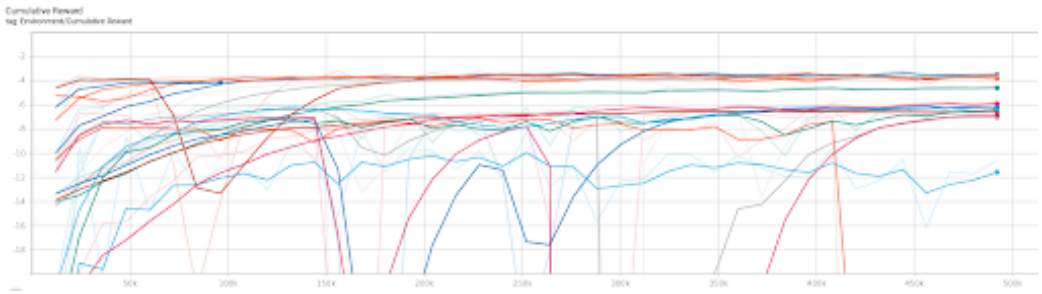
D. DQN configuration for hyperparameters.

DQN HYPERPARAMETERS	
BATCH_SIZE	256
EPS_DECAY	200
EPS_END	0.05
EPS_START	0.7
FC1	64
FC2	32
GAMMA	0.3
LR	0.001
NUM_EPISODES	25
TARGET_UPDATE	5

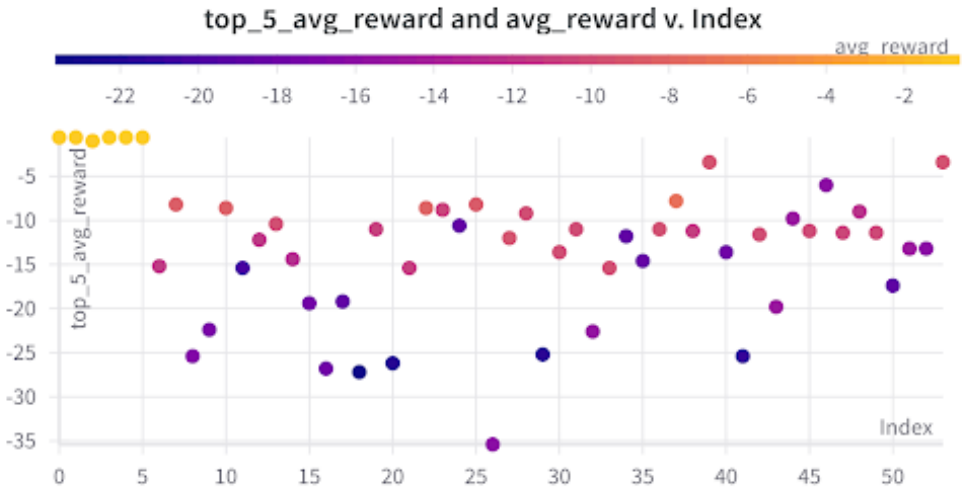
E. DQN hyperparameter tuning for the audio task.



F. Base models reward training results for all fox environments.



G. DQN reward training trajectory for each task.



H. DQN training average reward for last five episodes results used in t-test.

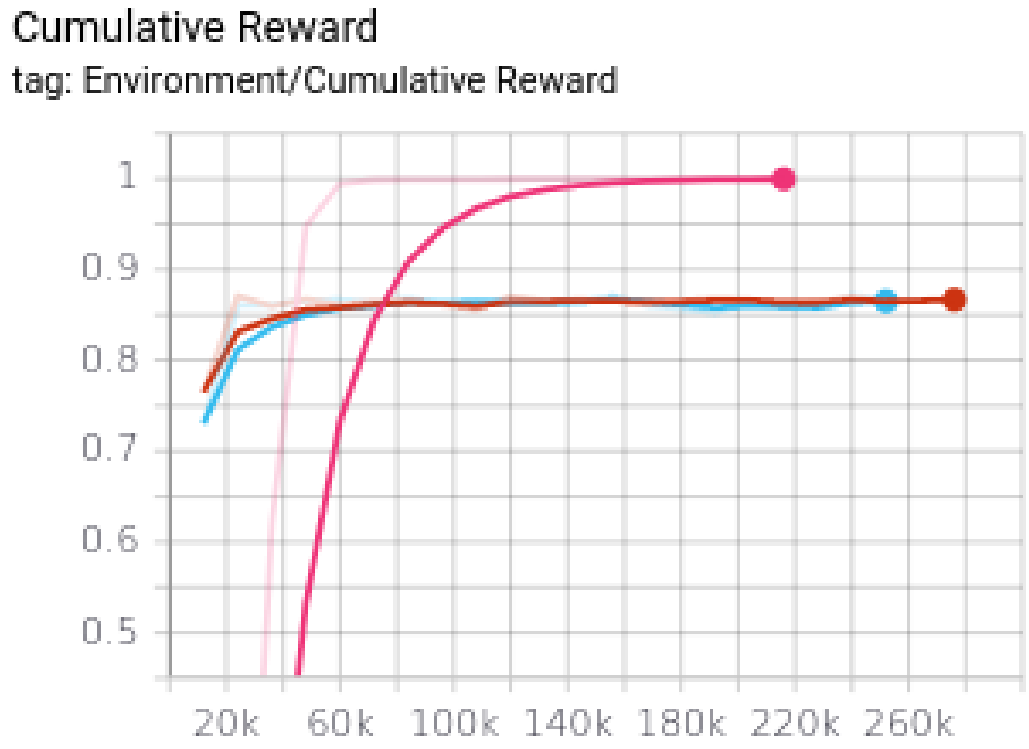
DQN TRAINING					
Name	avg_reward	episode_rewards	top_5_avg_reward	AVG. TOP 5 PER ENV	
none	-11.88000134	-9.000000954	**-15.20000169**		P value and statistical significance:
none	-8.720000982	-10.00000107	**-8.200000906**		The two-tailed P value equals 0.0411
none	-16.92000196	-58.00000679	**-25.40000296**		By conventional criteria, this difference is considered to be statistically significant.
none	-17.44000203	-6.000000596	**-22.4000026**		Confidence interval:
none	-8.680000982	-14.00000155	**-8.600001001**		The mean of none minus organ equals -4.02500046221
none	-20.52000239	-27.0000031	**-15.40000181**		95% confidence interval of this difference: From -7.88032033078 to -0.16968059364
none	-12.04000135	1	**-12.20000138**		Intermediate values used in calculations:
none	-10.0000011	-10.00000107	**-10.40000112**		t = 2.1015
none	-12.96000153	-33.00000381	**-14.40000169**		df = 46
none	-17.96000209	-11.00000119	**-19.40000229**		standard error of difference = 1.915
none	-18.04000208	-40.00000465	**-26.80000312**		
none	-19.04000223	-12.00000155	**-19.20000226**		
none	-23.64000276	-14.00000155	**-27.20000317**		
none	-11.20000126	-5.000000715	**-11.00000129**		
none	-22.3600026	-7.000000715	**-26.200003**		
none	-11.32000129	-15.00000191	**-15.40000176**		
none	-8.360000954	-10.00000107	**-8.600001001**		
none	-10.92000122	-9.000000954	**-8.800000978**		
none	-18.84000219	-12.00000131	**-10.60000124**		
none	-9.000001059	-16.00000179	**-8.200000954**		
none	-16.32000189	-55.00000644	**-35.40000415**		
none	-9.560001078	-1.000000238	**-12.00000136**		
none	-10.16000113	1	**-9.200001025**		
none	-22.08000256	-19.00000215	**-25.20000298**	-16.47500191	
raycast	-10.08000113	-14.00000155	-13.6000015		
raycast	-10.28000117	-5.000000477	-11.00000124		
raycast	-15.72000187	-47.00000548	-22.60000272		
raycast	-9.920001135	-8.000000954	-15.40000174		
raycast	-19.04000225	-17.00000179	-11.80000136		
raycast	-18.76000221	-20.00000226	-14.60000172	-14.88000171	

audio	-9.960001097	-8.000000715	-11.00000117		
audio	-7.240000854	-8.000001192	-7.800000978		
audio	-11.48000135	-8.000000834	-11.20000126		
audio	-9.640001106	-3.000000477	-3.400000501		
audio	-18.04000208	-21.00000238	-13.6000016		
audio	-21.52000253	-12.00000131	-25.40000298	-12.06666808	
position	-10.12000111	-10.00000107	-11.60000126		
position	-15.28000179	-23.00000286	-19.80000238		
position	-15.36000179	1	-9.800001168		
position	-10.36000115	-14.00000143	-11.20000119		
position	-16.24000189	-6.000000596	-6.000000691		
position	-10.72000123	-1.19E-07	-11.40000134	-11.63333467	
visual	-12.60000141	-10.00000107	-9.000001025		
visual	-10.5200012	-1.70E+01	-11.40000129		
visual	-19.24000223	-6.000000596	-17.400002		
visual	-15.16000177	-9.000001192	-13.2000016		
visual	-16.36000189	-10.00000107	-13.2000015		
visual	-9.760001097	1	-3.400000429	-11.26666797	
match	-0.76	-1	-0.6		
match	-0.76	-1	-0.6		
match	-0.84	-1	-1		
match	-0.68	-1	-0.6		
match	-0.6	1	-0.6		
match	-0.76	-1	-0.6	-0.6666666667	

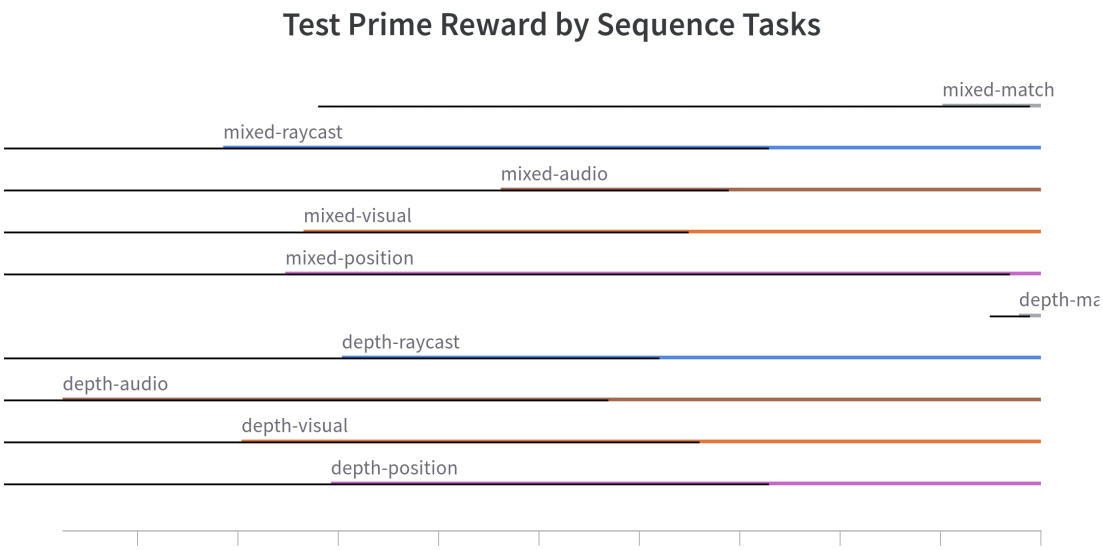
I. Training accuracies for all base models and environments.

ENVIRONMENT	MODEL	TRAINING ACCURACY
AUDIO	DQN	-12.067
AUDIO	PPO	-5.88
AUDIO	SAC	-6.39
AUDIO	SAC + LSTM	-6.342
VISUAL	DQN	-11.267
VISUAL	PPO	-4.58
VISUAL	SAC	-3.79
VISUAL	SAC + LSTM	-3.49
POSITION	DQN	-11.633
POSITION	PPO	-6.17
POSITION	SAC	-7.38
POSITION	SAC + LSTM	-7.06
RAYCAST	DQN	-14.88
RAYCAST	PPO	-3.58
RAYCAST	SAC	-3.52
RAYCAST	SAC + LSTM	-3.73
NONE	DQN	-16.475
NONE	PPO	-6.53
NONE	SAC	-6.81
NONE	SAC + LSTM	-11.56
MATCH	DQN	0.66
MATCH	PPO	0.99
MATCH	SAC	0.86
MATCH	SAC + LSTM	0.86

J. Match training for PPO (pink), SAC (blue), and SAC+LSTM (red).



K. DQN-GA Prime Accuracy Test Results.



L. Research Question and Hypothesis Results.

R1H1	**TRAINING**		ACCURACIES				
	DEPTH SEQ	DQN + GA	**-8.266667652**				
	DEPTH SEQ	DQN + GA -RC	**-14.73333502**				
	DEPTH SEQ	DQN - GA	**-17.13333533**				
R1H2	**TRAINING**		ACCURACIES				
	DEPTH SEQ	DQN + GA	**-8.266667652**				
	DEPTH SEQ	DQN - GA	**-17.13333533**				
	TESTING		ACCURACIES				
	DEPTH SEQ	DQN + GA	**match = -0.64**	**raycast = -13.42000155**	**audio = -11.62000135**	**position = -11.32000131**	**visual = -15.60000181**
	DEPTH SEQ	DQN - GA	**match = -0.44**	**raycast = -13.94000165**	**audio = -19.58000224**	**position = -14.14000164**	**visual = -15.92000183**
	TRAINING		ACCURACIES				
	MIXED SEQ	DQN + GA	**-10.0666678**				
	MIXED SEQ	DQN - GA	**-21.80000254**				
	TESTING		ACCURACIES				
	MIXED SEQ	DQN + GA	**match = -0.64**	**raycast = -12.84000148**	**audio = -10.44000121**	**position = -14.66000169**	**visual = -10.60000121**
	MIXED SEQ	DQN - GA	**match = -0.6**	**raycast = -16.28000191**	**audio = -10.78000124**	**position = -16.44000194**	**visual = -14.70**
R1H1	**TRAINING**		Layer 1	Layer 2	Layer 3		
	DEPTH SEQ	DQN + GA	**1.52**	**0.07**	**0.05**		
R2H2	**TEST**		ACCURACIES				
	DEPTH SEQ	DQN + GA	**match = -0.64**	**raycast = -13.42000155**	**audio = -11.62000135**	**position = -11.32000131**	**visual = -15.60000181**
	DEPTH SEQ	DQN - GA	**match = -0.44**	**raycast = -13.94**	**audio = -19.58**	**position = -14.14**	**visual = -14.70000172**

REFERENCES

[1] Phil Torres. "The possibility and risks of artificial general intelligence". In: *Bulletin of the Atomic Scientists* 75.3 (2019), pp. 105–108.

[2] Ben Goertzel. "Artificial general intelligence: concept, state of the art, and future prospects". In: *Journal of Artificial General Intelligence* 5.1 (2014), p. 1.

[3] Alejandro Barredo Arrieta et al. "Explainable Artificial Intelligence (XAI): Concepts, taxonomies, opportunities and challenges toward responsible AI". In: *Information fusion* 58 (2020), pp. 82–115.

[4] Harry Jerison. *Evolution of the brain and intelligence*. Elsevier, 2012.

[5] L Andrew Coward. "Modelling memory and learning consistently from psychology to physiology". In: *Perception-Action Cycle*. Springer, 2011, pp. 63–133.

[6] L Andrew Coward. *A system architecture approach to the brain: From neurons to consciousness*. Nova Publishers, 2005.

[7] L Andrew Coward. *Towards a theoretical neuroscience: from cell chemistry to cognition*. Vol. 8. Springer Science & Business Media, 2013.

[8] Gerhard Fischer. "Lifelong learning—more than training". In: *Journal of Interactive Learning Research* 11.3 (2000), pp. 265–294.

[9] Jerry A Fodor and Zenon W Pylyshyn. "Connectionism and cognitive architecture: A critical analysis". In: *Cognition* 28.1-2 (1988), pp. 3–71.

[10] Timothy P Lillicrap et al. "Backpropagation and the brain". In: *Nature Reviews Neuroscience* 21.6 (2020), pp. 335–346.

[11] Sebastian Seung. *Connectome: How the brain's wiring makes us who we are*. HMH, 2012.

[12] Geoffrey E Hinton. "Connectionist learning procedures". In: *Machine learning*. Elsevier, 1990, pp. 555–610.

[13] John R Riesenberg. "Catastrophic Forgetting in Neural Networks". PhD thesis. University of Cincinnati, 2000.

[14] Margaret Wilson. "Six views of embodied cognition". In: *Psychonomic bulletin & review* 9.4 (2002), pp. 625–636.

[15] Michael L Anderson. "Embodied cognition: A field guide". In: *Artificial intelligence* 149.1 (2003), pp. 91–130.

[16] Kenji Doya. "Reinforcement learning: Computational theory and biological mechanisms". In: *HFSP journal* 1.1 (2007), p. 30.

[17] Leslie Pack Kaelbling, Michael L Littman, and Andrew W Moore. "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* 4 (1996), pp. 237–285.

[18] Stephen Jay Gould. "Darwinism and the expansion of evolutionary theory". In: *Science* 216.4544 (1982), pp. 380–387.

[19] Michael D Vose. *The simple genetic algorithm: foundations and theory*. MIT press, 1999.

[20] Tiantian Zhang et al. "Catastrophic Interference in Reinforcement Learning: A Solution Based on Context Division and Knowledge Distillation". In: *arXiv preprint arXiv:2109.00525* (2021).

[21] David M Bossens and Adam J Sobey. "Lifetime policy reuse and the importance of task capacity". In: *arXiv preprint arXiv:2106.01741* (2021).

[22] Abhiit Banerjee, Dipendranath Ghosh, and Suvrojit Das. "Hyper-parameter tuned deep q network for area estimation of oil spills: a meta-heuristic approach". In: *Evolutionary Intelligence* 14.1 (2021), pp. 175–190.

[23] L Andrew Coward and Tamas O Gedeon. "Implications of resource limitations for a conscious machine". In: *Neurocomputing* 72.4-6 (2009), pp. 767–788.

[24] Brian Carnell. *The Etymology of the Word "Computer"*. 2015. URL: `https://brian.carnell.com/articles/2015/the-etymology-of-the-word-computer/` (visited on 03/05/2022).

[25] Wikipedia. *Antikythera mechanism*. 2022. URL: `https://en.wikipedia.org/wiki/Antikythera_mechanism` (visited on 03/05/2022).

[26] Wikipedia. *Antikythera mechanism*. 2022. URL: `https://en.wikipedia.org/wiki/Abacus` (visited on 03/05/2022).

[27] Jim O'Reilly. *Jacquard Looms at Lang Pioneer Village Museum*. 2015. URL: `http://thechawkersfoundation.org/jacquard-looms-at-lang-pioneer-village/` (visited on 03/05/2022).

[28] DRS Education. *James Burke Connections, Ep. 4 "Faith in Numbers"*. 2019. URL: `https://www.youtube.com/watch?v=z6yL0_sDnX0&ab_channel=DRS_Education` (visited on 03/05/2022).

[29] S Barry Cooper and Jan Van Leeuwen. *Alan Turing: His work and impact*. Elsevier, 2013.

[30] Atomic Heritage Foundation. *John von Neumann*. 2019. URL: `https://www.atomicheritage.org/profile/john-von-neumann` (visited on 03/05/2022).

[31] Alan M Turing. "Computing machinery and intelligence". In: *Parsing the turing test*. Springer, 2009, pp. 23–65.

[32] The History of Artificial Intelligence. *Rockwell Anyoha*. 2017. URL: `https://sitn.hms.harvard.edu/flash/2017/history-artificial-intelligence/` (visited on 03/05/2022).

[33] Donald Olding Hebb. *The organization of behavior: A neuropsychological theory*. Psychology Press, 2005.

[34] WS McCulloh and W Pitts. "A logical calculus of the ideas immanent in neural nets". In: *Bull Math. Biophys* 5 (1943), pp. 133–137.

[35] Frank Rosenblatt. *The perceptron, a perceiving and recognizing automaton Project Para*. Cornell Aeronautical Laboratory, 1957.

[36] Alan Mathison Turing. *Intelligent machinery*. 1948.

[37] John J Hopfield. "Neural networks and physical systems with emergent collective computational abilities". In: *Feynman and computation*. CRC Press, 2018, pp. 7–19.

[38] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. *Learning internal representations by error propagation*. Tech. rep. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.

[39] Edward A Feigenbaum. "Expert systems in the 1980s". In: *State of the art report on machine intelligence. Maidenhead: Pergamon-Infotech* (1981).

[40] Arthur J Bernstein. "Analysis of programs for parallel processing". In: *IEEE transactions on electronic computers* 5 (1966), pp. 757–763.

[41] Atilim Gunes Baydin et al. "Automatic differentiation in machine learning: a survey". In: *Journal of Marchine Learning Research* 18 (2018), pp. 1–43.

[42] Volodymyr Mnih et al. "Playing atari with deep reinforcement learning". In: *arXiv preprint arXiv:1312.5602* (2013).

[43] Jianqing Fan et al. "A theoretical analysis of deep Q-learning". In: *Learning for Dynamics and Control*. PMLR. 2020, pp. 486–489.

[44] Blake A Richards et al. "A deep learning framework for neuroscience". In: *Nature neuroscience* 22.11 (2019), pp. 1761–1770.

[45] Sebastian Ruder. "An overview of multi-task learning in deep neural networks". In: *arXiv preprint arXiv:1706.05098* (2017).

[46] Kevin Lu et al. "Pretrained transformers as universal computation engines". In: *arXiv preprint arXiv:2103.05247* (2021).

[47] Hao Li et al. "Pruning filters for efficient convnets". In: *arXiv preprint arXiv:1608.08710* (2016).

[48] Russell Reed. "Pruning algorithms-a survey". In: *IEEE transactions on Neural Networks* 4.5 (1993), pp. 740–747.

[49] Joseph Early. "Reducing catastrophic forgetting when evolving neural networks". In: *arXiv preprint arXiv:1904.03178* (2019).

[50] George FR Ellis. "Top-down causation and emergence: some comments on mechanisms". In: *Interface Focus* 2.1 (2012), pp. 126–140.

[51] Kevin Frans and Olaf Witkowski. "Population-Based Evolution Optimizes a Meta-Learning Objective". In: *arXiv preprint arXiv:2103.06435* (2021).

[52] Samuel Butler and Denis Foa. *Erewhon*. Royal Victorian Institute for the Blind Tertiary Resource Service., 1988.

[53] Lawrence Krauss and Lawrence M Krauss. *The physics of star trek*. Basic Books (AZ), 2007.

[54] Stanley Kubrick. "2001: a space odyssey". In: (1968).

[55] Ben Goertzel and Cassio Pennachin. *Artificial general intelligence*. Vol. 2. Springer, 2007.

[56] Ben Goertzel. "Reflective Metagraph Rewriting as a Foundation for an AGI" Language of Thought"". In: *arXiv preprint arXiv:2112.08272* (2021).

[57] How many words do you need to speak a language? *Beth Sagar-Fenton Lizzy McNeill*. 2018. URL: `https://www.bbc.com/news/world-44569277` (visited on 03/05/2022).

[58] Cameron Buckner and James Garson. "Connectionism". In: (1997).

[59] Ernest Jones. "The theory of symbolism". In: *British Journal of Psychology* 9.2 (1918), p. 181.

[60] Tzvetan Todorov. *Symbolism and interpretation*. Cornell University Press, 1986.

[61] Charles Kay Ogden and Ivor Armstrong Richards. *The Meaning of Meaning: A Study of the Influence of Language upon Thought and of the Science of Symbolism*. Vol. 29. Harcourt, Brace, 1925.

[62] Sybille Kramer. "Mind, symbolism, formalism: Is Leibniz a precursor of artificial intelligence?" In: *KO KNOWLEDGE ORGANIZATION* 23.2 (1996), pp. 83–87.

[63] Ben Goertzel. "The general theory of general intelligence: a pragmatic patternist perspective". In: *arXiv preprint arXiv:2103.15100* (2021).

[64] Seth Baum. "A survey of artificial general intelligence projects for ethics, risk, and policy". In: *Global Catastrophic Risk Institute Working Paper* (2017), pp. 17–1.

[65] Michael Genesereth Vinay K. Chaudhri Naren Chittar. *CS 520 Knowledge Graphs*. 2021. URL: `https://web.stanford.edu/class/cs520/` (visited on 03/05/2022).

[66] Antonio Lieto. *Cognitive design for artificial minds*. Routledge, 2021.

[67] Md Kamruzzaman Sarker et al. "Neuro-symbolic artificial intelligence: Current trends". In: *arXiv preprint arXiv:2105.05330* (2021).

[68] Zonghan Wu et al. "A comprehensive survey on graph neural networks". In: *IEEE transactions on neural networks and learning systems* 32.1 (2020), pp. 4–24.

[69] James CR Whittington and Rafal Bogacz. "Theories of error back-propagation in the brain". In: *Trends in cognitive sciences* 23.3 (2019), pp. 235–250.

[70] Rachel A StClair, William Edward Hahn, and Elan Barenholtz. "The Role of Bio-Inspired Modularity in General Learning". In: *International Conference on Artificial General Intelligence*. Springer. 2021, pp. 261–268.

[71] Alexander Ororbia et al. "Lifelong neural predictive coding: Learning cumulatively online without forgetting". In: *arXiv preprint arXiv:1905.10696* (2019).

[72] Gal Chechik, Isaac Meilijson, and Eytan Ruppin. "Neuronal regulation: A mechanism for synaptic pruning during brain maturation". In: *Neural computation* 11.8 (1999), pp. 2061–2080.

[73] Uri Hasson, Samuel A Nastase, and Ariel Goldstein. "Direct fit to nature: an evolutionary perspective on biological and artificial neural networks". In: *Neuron* 105.3 (2020), pp. 416–434.

[74] Luciano Floridi and Massimo Chiriatti. "GPT-3: Its nature, scope, limits, and consequences". In: *Minds and Machines* 30.4 (2020), pp. 681–694.

[75] Robert Dale. "GPT-3: What's it good for?" In: *Natural Language Engineering* 27.1 (2021), pp. 113–118.

[76] Zihao Zhao et al. "Calibrate before use: Improving few-shot performance of language models". In: *International Conference on Machine Learning*. PMLR. 2021, pp. 12697–12706.

[77] Suzana Herculano-Houzel and Roberto Lent. "Isotropic fractionator: a simple, rapid method for the quantification of total cell and neuron numbers in the brain". In: *Journal of Neuroscience* 25.10 (2005), pp. 2518–2521.

[78] Matthew Hutson et al. "Robo-writers: The rise and risks of language-generating AI". In: *Nature* 591.7848 (2021), pp. 22–25.

[79] Bernard J Baars. *A cognitive theory of consciousness*. Cambridge University Press, 1993.

[80] Antti Revonsuo. *Consciousness: The science of subjectivity*. Psychology Press, 2009.

[81] Sean Kugele and Stan Franklin. "Learning in LIDA". In: *Cognitive Systems Research* 66 (2021), pp. 176–200.

[82] Changeux Dehaene Stanislas and Jean-Pierre. "A simple model of prefrontal cortex function in delayed-response tasks". In: *Journal of Cognitive Neuroscience* 1.3 (1989), pp. 244–261.

[83] Anirudh Goyal et al. "Coordination among neural modules through a shared global workspace". In: *arXiv preprint arXiv:2103.01197* (2021).

[84] Giulio Tononi. "An information integration theory of consciousness". In: *BMC neuroscience* 5.1 (2004), pp. 1–22.

[85] Scott Aaronson. *Why I Am Not An Integrated Information Theorist (or, The Unconscious Expander)*. 2014. URL: https://scottaaronson.blog/?p=1799 (visited on 03/05/2022).

[86] Adrien Doerig et al. "The unfolding argument: Why IIT and other causal structure theories cannot explain consciousness". In: *Consciousness and cognition* 72 (2019), pp. 49–59.

[87] William GP Mayner et al. "PyPhi: A toolbox for integrated information theory". In: *PLoS computational biology* 14.7 (2018), e1006343.

[88] Erik P Hoel et al. "Can the macro beat the micro? Integrated information across spatiotemporal scales". In: *Neuroscience of Consciousness* 2016.1 (2016).

[89] Adam B Barrett and Anil K Seth. "Practical measures of integrated information for time-series data". In: *PLoS computational biology* 7.1 (2011), e1001052.

[90] Pedro AM Mediano, Anil K Seth, and Adam B Barrett. "Measuring integrated information: Comparison of candidate measures in theory and simulation". In: *Entropy* 21.1 (2019), p. 17.

[91] Rajesh PN Rao and Dana H Ballard. "Predictive coding in the visual cortex: a functional interpretation of some extra-classical receptive-field effects". In: *Nature neuroscience* 2.1 (1999), pp. 79–87.

[92] Andy Clark. *Surfing uncertainty: Prediction, action, and the embodied mind*. Oxford University Press, 2015.

[93] Andy Clark. "Whatever next? Predictive brains, situated agents, and the future of cognitive science". In: *Behavioral and brain sciences* 36.3 (2013), pp. 181–204.

[94] Karl Friston. "Does predictive coding have a future?" In: *Nature neuroscience* 21.8 (2018), pp. 1019–1021.

[95] Philipp Sterzer et al. "The predictive coding account of psychosis". In: *Biological psychiatry* 84.9 (2018), pp. 634–643.

[96] Tengda Han, Weidi Xie, and Andrew Zisserman. "Video representation learning by dense predictive coding". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision Workshops*. 2019, pp. 0–0.

[97] Tengda Han, Weidi Xie, and Andrew Zisserman. "Memory-augmented dense predictive coding for video representation learning". In: *European conference on computer vision*. Springer. 2020, pp. 312–329.

[98] Michael W Spratling. "A review of predictive coding algorithms". In: *Brain and cognition* 112 (2017), pp. 92–97.

[99] L Andrew Coward. "The recommendation architecture: lessons from large-scale electronic systems applied to cognition". In: *Cognitive Systems Research* 2.2 (2001), pp. 111–156.

[100] L Andrew Coward. *Pattern thinking*. Greenwood Publishing Group, 1990.

[101] L Andrew Coward and Tamas D Gedeon. "Using the change manager model for the hippocampal system to predict connectivity and neurophysiological parameters in the perirhinal cortex". In: *Computational Intelligence and Neuroscience* 2016 (2016).

[102] L Andrew Coward. "The pattern extraction architecture: A connectionist alternative to the von Neumann architecture". In: *International Work-Conference on Artificial Neural Networks*. Springer. 1997, pp. 634–643.

[103] Ian J Goodfellow et al. "An empirical investigation of catastrophic forgetting in gradient-based neural networks". In: *arXiv preprint arXiv:1312.6211* (2013).

[104] Ronald Kemker et al. "Measuring catastrophic forgetting in neural networks". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 32. 1. 2018.

[105] James Kirkpatrick et al. "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the national academy of sciences* 114.13 (2017), pp. 3521–3526.

[106] Tyler L Hayes et al. "Remind your neural network to prevent catastrophic forgetting". In: *European Conference on Computer Vision*. Springer. 2020, pp. 466–483.

[107] Joan Serra et al. "Overcoming catastrophic forgetting with hard attention to the task". In: *International Conference on Machine Learning*. PMLR. 2018, pp. 4548–4557.

[108] Xilai Li et al. "Learn to grow: A continual structure learning framework for overcoming catastrophic forgetting". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 3925–3934.

[109] Junfeng Wen, Yanshuai Cao, and Ruitong Huang. "Few-shot self reminder to overcome catastrophic forgetting". In: *arXiv preprint arXiv:1812.00543* (2018).

[110] Craig Atkinson et al. "Pseudo-rehearsal: Achieving deep reinforcement learning without catastrophic forgetting". In: *Neurocomputing* 428 (2021), pp. 291–307.

[111] Dongbo Liu et al. "An improved dual-channel network to eliminate catastrophic forgetting". In: *IEEE Transactions on Systems, Man, and Cybernetics: Systems* (2020).

[112] Joao Ribeiro, Francisco S Melo, and Joao Dias. "Multi-task learning and catastrophic forgetting in continual reinforcement learning". In: *arXiv preprint arXiv:1909.10008* (2019).

[113] Anthony Robins and SIMON McCALLUM. "Catastrophic forgetting and the pseudorehearsal solution in Hopfield-type networks". In: *Connection Science* 10.2 (1998), pp. 121–135.

[114] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[115] Monika Schak and Alexander Gepperth. “A study on catastrophic forgetting in deep LSTM networks”. In: *International Conference on Artificial Neural Networks*. Springer. 2019, pp. 714–728.

[116] Claudio Greco et al. “Measuring Catastrophic Forgetting in Visual Question Answering”. In: *Increasing Naturalness and Flexibility in Spoken Dialogue Interaction*. Springer, 2021, pp. 381–387.

[117] Nicolas Y Masse, Gregory D Grant, and David J Freedman. “Alleviating catastrophic forgetting using context-dependent gating and synaptic stabilization”. In: *Proceedings of the National Academy of Sciences* 115.44 (2018), E10467–E10475.

[118] Dylan R Ashley, Sina Ghiassian, and Richard S Sutton. “Does the Adam Optimizer Exacerbate Catastrophic Forgetting?” In: *arXiv preprint arXiv:2102.07686* (2021).

[119] Thang Doan et al. “A theoretical analysis of catastrophic forgetting through the ntk overlap matrix”. In: *International Conference on Artificial Intelligence and Statistics*. PMLR. 2021, pp. 1072–1080.

[120] Vinay V Ramasesh, Ethan Dyer, and Maithra Raghu. “Anatomy of catastrophic forgetting: Hidden representations and task semantics”. In: *arXiv preprint arXiv:2007.07400* (2020).

[121] Jiahao Huo and Terence L van Zyl. “Comparative Analysis of Catastrophic Forgetting in Metric Learning”. In: *2020 7th International Conference on Soft Computing & Machine Intelligence (ISCMI)*. IEEE. 2020, pp. 68–72.

[122] Karl Weiss, Taghi M Khoshgoftaar, and DingDing Wang. “A survey of transfer learning”. In: *Journal of Big data* 3.1 (2016), pp. 1–40.

[123] Shunsuke Imai, Shin Kawai, and Hajime Nobuhara. “Stepwise pathnet: a layer-by-layer knowledge-selection-based transfer learning algorithm”. In: *Scientific Reports* 10.1 (2020), pp. 1–14.

[124] Steven Williams and Larry Yaeger. “Evolution of neural dynamics in an ecological model”. In: *Geosciences* 7.3 (2017), p. 49.

[125] Deok-Sun Lee. “Evolution of regulatory networks towards adaptability and stability in a changing environment”. In: *Physical Review E* 90.5 (2014), p. 052822.

[126] Joaquin Vanschoren. “Meta-learning: A survey”. In: *arXiv preprint arXiv:1810.03548* (2018).

[127] Alireza Goudarzi et al. “Emergent criticality through adaptive information processing in Boolean networks”. In: *Physical review letters* 108.12 (2012), p. 128702.

[128] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. "Neural architecture search: A survey". In: *The Journal of Machine Learning Research* 20.1 (2019), pp. 1997–2017.

[129] Mantas Lukoševičius, Herbert Jaeger, and Benjamin Schrauwen. "Reservoir computing trends". In: *KI-Künstliche Intelligenz* 26.4 (2012), pp. 365–371.

[130] Fabrice Normandin et al. "Sequoia: A Software Framework to Unify Continual Learning Research". In: *arXiv preprint arXiv:2108.01005* (2021).

[131] Quanziang Wang et al. "Revisiting Experience Replay: Continual Learning by Adaptively Tuning Task-wise Relationship". In: *arXiv preprint arXiv:2112.15402* (2021).

[132] Andy Cahill. "Catastrophic forgetting in reinforcement-learning environments". PhD thesis. University of Otago, 2011.

[133] Sean Mondesire and R Paul Wiegand. "Forgetting Beneficial Knowledge in Decomposition-Based Reinforcement Learning Using Evolutionary Computation". In: *Proceedings of the International Conference on Genetic and Evolutionary Methods (GEM)*. The Steering Committee of The World Congress in Computer Science, Computer ... 2014, p. 1.

[134] Claes Strannegård et al. "The animat path to artificial general intelligence". In: *Proceedings of IJCAI-17 Workshop on Architectures for Generality & Autonomy*. 2017.

[135] Ricardo Vilalta and Youssef Drissi. "A perspective view and survey of meta-learning". In: *Artificial intelligence review* 18.2 (2002), pp. 77–95.

[136] A Juliani et al. "Unity: A general platform for intelligent agents. arXiv 2018". In: *arXiv preprint arXiv:1809.02627* ().

[137] .

[138] John Schulman et al. "Proximal policy optimization algorithms". In: *arXiv preprint arXiv:1707.06347* (2017).

[139] Tuomas Haarnoja et al. "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905* (2018).

[140] Abien Fred Agarap. "Deep learning using rectified linear units (relu)". In: *arXiv preprint arXiv:1803.08375* (2018).

[141] Ross Girshick. "Fast r-cnn". In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[142] Alex Graves. "Generating sequences with recurrent neural networks". In: *arXiv preprint arXiv:1308.0850* (2013).

[143] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Curran Associates, Inc., 2019, pp. 8024–8035. URL: `http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf`.

[144] NVIDIA, Péter Vingelmann, and Frank H.P. Fitzek. *CUDA, release: 10.2.89*. 2020. URL: https://developer.nvidia.com/cuda-toolkit.

[145] Adam Paszke. *Reinforcement Learning (DQN) Tutorial*.

[146] Stefan Droste, Thomas Jansen, and Ingo Wegener. "On the analysis of the (1+ 1) evolutionary algorithm". In: *Theoretical Computer Science* 276.1-2 (2002), pp. 51–81.

[147] Lukas Biewald. *Experiment Tracking with Weights and Biases*. Software available from wandb.com. 2020. URL: https://www.wandb.com/.

[148] Yi-Hong Liang, Sin-Jin Kang, and Sung Hyun Cho. "A Study about the Usefulness of Reinforcement Learning in Business Simulation Games using PPO Algorithm". In: *Journal of Korea Game Society* 19.6 (2019), pp. 61–70.

[149] Pontus Andersson. *Future-proofing Video Game Agents with Reinforced Learning and Unity ML-Agents*. 2021.

[150] Abu Jafar Md Muzahid, Syafiq Fauzi Kamarulzaman, and Md Arafatur Rahman. "Comparison of ppo and sac algorithms towards decision making strategies for collision avoidance among multiple autonomous vehicles". In: *2021 International Conference on Software Engineering & Computer Systems and 4th International Conference on Computational Science and Information Management (ICSECS-ICOCSIM)*. IEEE. 2021, pp. 200–205.

[151] Jun Lai, Xi-liang Chen, and Xue-zhen Zhang. "Training an Agent for Third-person Shooter Game Using Unity ML-Agents". In: *International Conference on Artificial Intelligence and Computing Science. Hangzhou*. 2019, pp. 317–332.

[152] Abhishek Nandy and Manisha Biswas. "Unity ml-agents". In: *Neural Networks in Unity*. Springer, 2018, pp. 27–67.

[153] LA Rybak et al. "Development of an algorithm for managing a multi-robot system for cargo transportation based on reinforcement learning in a virtual environment". In: *IOP Conference Series: Materials Science and Engineering*. Vol. 945. 1. IOP Publishing. 2020, p. 012083.

[154] Xiao wenwen. "Application Research of end to end behavior decision based on deep reinforcement learning". In: *Proceedings of the 2021 5th International Conference on Electronic Information Technology and Computer Engineering*. 2021, pp. 889–894.

[155] Kevin Tan and Andy L Khuu. "Deep Reinforcement Learning Dodgeball". In: ().

[156] CPA Awoga and Oluwaseyi Tony PRM. "Using Deep Q-Networks to Train an Agent to Navigate the Unity ML-Agents Banana Environment". In: *Available at SSRN 3881878* (2021).

[157] Kyushik Min, Hayoung Kim, and Kunsoo Huh. "Deep distributional reinforcement learning based high-level driving policy determination". In: *IEEE Transactions on Intelligent Vehicles* 4.3 (2019), pp. 416–424.

[158] Qi Zhang, Tao Du, and Changzheng Tian. "Self-driving scale car trained by deep reinforcement learning". In: *arXiv preprint arXiv:1909.03467* (2019).

[159] Arina Afanasyeva and Maxim Buzdalov. "Optimization with auxiliary criteria using evolutionary algorithms and reinforcement learning". In: *Proceedings of 18th International Conference on Soft Computing MENDEL 2012*. Vol. 2012. 2012, pp. 58–63.

[160] Dimitris E Koulouriotis and A Xanthopoulos. "Reinforcement learning and evolutionary algorithms for non-stationary multi-armed bandit problems". In: *Applied Mathematics and Computation* 196.2 (2008), pp. 913–922.

[161] Yoshitaka Sakurai et al. "A method to control parameters of evolutionary algorithms by using reinforcement learning". In: *2010 sixth international conference on signal-image technology and internet based systems*. IEEE. 2010, pp. 74–79.

[162] P.A. Borisovsky and A.V. Eremeev. "Comparing evolutionary algorithms to the (1+1) -EA". In: *Theoretical Computer Science* 403.1 (2008), pp. 33–41. ISSN: 0304-3975. DOI: https://doi.org/10.1016/j.tcs.2008.03.008. URL: https://www.sciencedirect.com/science/article/pii/S0304397508002028.

[163] Martín Abadi et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[164] Douglas M Hawkins. "The problem of overfitting". In: *Journal of chemical information and computer sciences* 44.1 (2004), pp. 1–12.

[165] Micheal L Mavrovouniotis and S Chang. "Hierarchical neural networks". In: *Computers & chemical engineering* 16.4 (1992), pp. 347–369.