Neuromodulated networks for lifelong learning and adaptation

Neuromodulated Networks for Lifelong Learning and Adaptation

by

Eseoghene Ben-Iwhiwhu

A thesis submitted to Loughborough University
in fulfilment of the requirements for the award of

Doctor of Philosophy

Department of Computer Science
Loughborough University, United Kingdom

15 March 2023

Abstract

The development of robust and adaptable intelligent system has been a long standing grand challenge. Recently, machine learning methods via neural networks have gained prominence. However, most systems are excellent at solving a single task. The field of lifelong (machine) learning seeks to alleviate this problem by endowing intelligent agents with the ability to learn multiple tasks. Interesting questions arise from the field about how to design agents that can learn many tasks over a lifetime without forgetting, rapidly adapt to task changes, reuse existing knowledge to foster rapid learning of new tasks, use knowledge from task being learned to improve knowledge of previous tasks. This thesis investigates several of the aforementioned challenges through the use of biologically inspired neuromodulatory mechanisms, that are incorporated into standard artificial neural networks. Experiments were conducted in simulated reinforcement learning environments in domains such as navigation, robotics, and autonomous driving simulations. A common theme from the findings showed that neuromodulation enabled the lifelong learning systems to solve problems of increasing complexity in comparison to systems without neuromodulation. Also, neuromodulation enabled the rapid switch of learned behaviour via the dynamic regulation of the agent's neural activity in fully and partially observable scenarios, and the efficient learning of new tasks in an online manner without forgetting. However, the use of neuromodulation incurs an extra cost in computational and memory requirements in neural networks.

Acknowledgements

This doctoral study and thesis was made possible through the help, support and encouragement of a great deal of people.

I would first like to thank my supervisors Dr Andrea Soltoggio and Prof Wen-Hua Chen whose valuable comments and insights helped shaped my research. I am sincerely grateful to Andrea for the many wonderful discussions, research adventures, patience in shaping my research understanding, freedom to explore different research direction and his immense support. I also want to express my appreciation to the Systems Engineering and Technical Assistance (SETA) teams of the Lifelong Learning Machines (L2M) and Shared Experience Lifelong Learning (ShELL) DARPA programmes, for their support and useful discussions that benefited my research.

I would also like to thank Dr Pawel Ladosz whose insights, collaborations, and presence at the early stages of my study helped in setting the tone of my research journey, and made it a wonderful experience. Many thanks to Dr Firat Batmaz and Dr Martin Sykora for giving me the opportunity to teach and interface with students, and work on projects. I am sincerely grateful to Firat for several advice and discussions about my PhD and various topics in general.

I want to thank all my research collaborators in universities and organisations across the UK, France, Denmark and the US. Thanks for the many insightful comments and discussions that helped shaped my research landscape. I am also thankful to all my research colleagues and friends in Loughborough, for their support and giving me the opportunity to learn from their experiences. Attempts to mention all names here would be incomplete, but I am sincerely thankful to every single person.

In my stay so far in Loughborough, I have been immensely supported by my local church and I am sincerely grateful to everyone from the Pastor to all friends and members. I express my deep gratitude to my friends and family, for their constant support and care through my doctoral journey and earlier studies. Special thanks to my siblings, parents and wife. Lastly, I am sincerely grateful to God for endowing me with the tenacity, strength and every blessing to complete my doctoral study. A billion thanks is not enough.

Contents

List of Figures

List of Tables

Nomenclature

AI	Artificial Intelligence
L2M	Lifelong Learning Machine
ML	Machine Learning
SL	Supervised Learning
RL	Reinforcement Learning
LL	Lifelong Learning
LSL	Lifelong Supervised Learning
LRL	Lifelong Reinforcement Learning
CL	Continual Learning
MTL	Multi-Task Learning
Meta-RL	Meta Reinforcement Learning
PPO	Proximal Policy Optimization
TRPO	Trust Region Policy Optimization
SAC	Soft Actor Critic
MDP	Markov Decision Process
POMDP	Partially Observable Markov Decision Process
ANN	Artificial Neural Network
MLP	Multi-Layer Perceptron
CNN	Convolutional Neural Network
SGD	Stochastic Gradient Descent
RMSProp	Root Mean Squared Propagation
GA	Genetic Algorithm
ES	Evolution Strategies
SoTA	State of The Art
SPN	Standard Policy Network
NPN	Neuromodulated Policy Network
DA	Dopamine
5-HT	Serotonin
ACh	Acetycholine
NA or NE	Noradrenaline or Norepinephrine

Chapter 1

Introduction

The engineering and development of intelligent systems that mimics biological intelligence has been a long standing challenge in artificial intelligence (AI) and engineering science in general. Recent innovation and advances in machine learning (ML) have encouraged and revived a widespread attention across academic and commercial entities in pursuit of this challenge. Machine learning is a sub-field of AI that is focused on the development of intelligent systems that are capable of exhibiting intelligence or intelligent behaviour from data and experience. ML systems, specifically artificial neural network (ANN or neural net) methods are the current state-of-the-art (SoTA) in AI.

Through learning from data using ML, AI systems are now capable of large scale dialogue text generation [206, 55], language translation [39, 16, 276], object classification/detection/segmentation [141, 99, 80, 213, 220], play video games [189], robotics [187, 235, 75, 89] and more. Beyond research, the innovation has lead to the deployment of such systems in commercial products across search engines/information retrieval [159, 109, 296], autonomous vehicle navigation [124, 97, 170, 168], health care [150, 135, 208, 300, 114, 62, 316], life sciences [283, 1, 10, 120], education [33, 34, 162], legal [227], and so on. The accounts of such deployment only give a foretaste of the benefits that are to come when true general purpose intelligent systems are deployed in the future. Nevertheless, it suffice to say that these systems are now making daily impact in society.

Despite the advances, the current SoTA intelligent systems suffer from several challenges that the field seeks to address. Some of these challenges include narrow intelligence (solves only one or a few specific tasks), the drastic drop in performance for out-of-distribution data, lack of the ability to reason in ML, black-box decisions (lack of explanation behind the system's decisions), inability to deal with adversarial attacks and so on. This thesis seeks to investigate some of these challenges with focus on the development of robust systems that are capable of learning and adapting to many tasks.

Traditional ML systems operate in two phases: learning and deployment. Dur-

ing learning, the system is trained to solve a specific task (e.g., play chess) using data collected, afterwards, it is deployed to automate the task it was trained on. Such a system possess narrow intelligence and it will drastically fail when a new task is encountered during deployment. If the system is allowed to continuously train, then the system will learn to solve new tasks if they are encountered in deployment, but loses knowledge and performance on the previous task as it will be overwritten. Such failures is especially evident in neural network methods where task knowledge is encoded in the synapses or weights of the network. Learning a new task will overwrite the weights, thus leading to a drastic drop in performance termed catastrophic forgetting [90, 128].

Ideally, we want intelligent systems that are capable of learning multiple tasks like humans and animals. For example, humans can play chess, compose music, farming, play tennis, swim, drive a car, speak multiple languages, walk, run, study nature and so on. While on average a single human may not learn all known tasks in the world, such a person would still be capable of solving and learning several tasks over a lifetime. Therefore, incorporating such capabilities into a machine would be useful to enable the development of flexible, general purpose systems for the automation of several tasks.

The desiderata of such a general system include: (i) the avoidance or elimination of catastrophic forgetting, (ii) the ability to quickly learn new task (adaptation), (iii) ability to re-use knowledge from previous task to learn new ones, and (iv) the ability to use knowledge acquired from task currently being learnt to improve performance on previously learnt task. The unifying framework that facilitates the investigation and development of such system is referred to as lifelong learning (LL) [203, 128, 20]. Lifelong learning is then defined as the ability of intelligent systems to continuously learn multiple tasks over the system's lifetime, while satisfying the aforementioned desiderata.

Also, the system should possess the ability to adapt to varying task conditions [255, 112, 197]. Adaptation can be viewed as a special case of learning (under desiderata iii) where the focus is on the learning speed. The system should be capable of adapting its behaviour to (rapidly learn) a task that is similar to the ones previously learnt. For example, a person that knows how to drive in the UK (right-hand drive) would likely learn to drive in France (left-hand drive) from a few trials in comparison to someone with no prior driving knowledge. Therefore, learning and adaptation to several tasks and task variations capture the essentials of a lifelong learning system.

1.1 Neural Learning Systems

The recent advances in intelligent systems development are powered by artificial neural networks (also referred to as ANNs or neural networks). Across task domains such as computer vision [141, 99, 80, 220], language processing [39, 16, 276, 206, 55] and agent-based systems [188, 187, 235, 103, 75, 89], neural network methods have gained prominence.

Drawing inspiration from biological brains, an ANN consist of a set of nodes (neurons) interconnected by a set of edges (synapses). The edges are referred to as weights or parameters of the network. To learn an intelligent behaviour from data using ANNs, the weights are adjusted to satisfy an objective as the data is processed by the network. After learning, the parameters of the network encode the acquired knowledge. Some of the popular neural architectures include multilayer perceptron (MLP), convolutional neural network (CNN) [76, 146, 147, 148], transformers [276, 56], and recurrent variants [106, 40].

When ANNs are employed in a lifelong learning setup, the parameters of the network are continuously updated as new tasks and corresponding objectives are encountered. Without a way to preserve knowledge, the knowledge encoded in the parameters of the network are constantly overwritten, thus leading to catastrophic forgetting. To avoid this issue and enable efficient re-use of knowledge for adaptation, modifications are required. Drawing inspiration from neuroscience and biology, several techniques [128, 273, 90] have been proposed to reduce or avoid the effect of forgetting, while retaining knowledge and performance of previously learnt tasks. The modifications come at the cost of extra storage or processing.

Loosely drawing inspiration from neuroscience, some techniques employed in neural lifelong learning systems include synaptogenesis and neurogenesis, synaptic memory consolidation, brain inspired replay, and neuromodulation. In synaptogenesis and neurogenesis [224, 297, 176], the emphasis is on growing the size of the network (neurons and synapses) as the new tasks are encountered. In synaptic memory consolidation [132, 303, 136], parameters encoding useful information for solving a task are identified and discouraged from any further updates. In replay methods [167, 37, 219], data from previous tasks are constantly stored, and the network is trained using a mix of both old data and current task data. Even still, neuromodulation techniques [251, 277, 182, 23, 287] employ the use of a neuromodulatory process to isolate a sub-region of the network (parameters or neurons) useful for solving specific tasks. It is on the last class of methods that the investigation in thesis reports on.

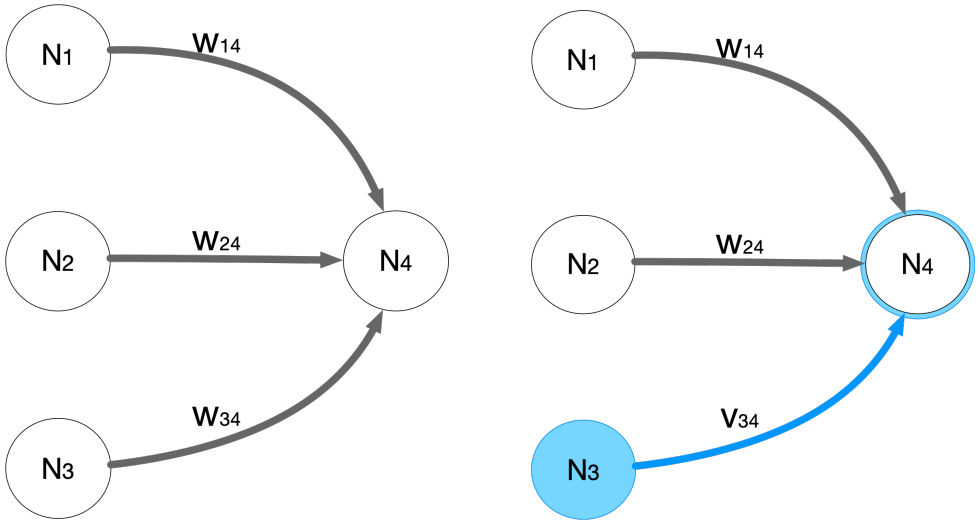


Figure 1.1: Snippet of a standard ANN (left) and neuromodulated network (right). Standard ANN consist of neurons and parameters W_{14}, W_{24}, and W_{34}. Neuromodulated network consist of standard neurons, with parameters W_{14}, W_{24}, and neuromodulator (highlighted in blue), with parameters V_{34}. The neuromodulatory activity on neuron N_4 (blue border) can either regulate its neural activation or the weights connected to it.

1.2 Neuromodulation

Evidence from neuroscience has shown the presence of neuromodulatory process (neurotransmitters) in biological brains. Biological neuromodulators facilitate the regulation of neuronal activities and the overall cognitive behaviour of humans and animals. For example, norepinephrine neuromodulator regulates the alertness and the body's fight-or-flight response in stress situations.

Loosely drawing inspiration from neuroscience, there exist studies [251, 277, 182, 291, 318] that incorporate neuromodulation with ANNs. We refer to this kind of network as a neuromodulated network which consist of both standard neurons (in ANNs) and neuromodulators (or modulatory neurons). Neuromodulators are modeled as special neurons that regulate either the weights or neuronal activities of the network. Figure 1.1 highlights the difference between a standard ANN and a neuromodulated network.

The focus of this thesis is the investigation into to the use design and use of neuromodulated networks in the lifelong learning setup. Taking advantage of the properties of neuromodulators, we seek to discover how they can facilitate efficient lifelong learning. By regulation, neuromodulators enable the selection of different sub-regions of the network relevant for solving different tasks.

Specifically, the focus is on agent-based lifelong learning (also called lifelong reinforcement learning), where the agent explores one or more environments, collect its own data, while receiving scalar rewards based on actions taken. In comparison to lifelong supervised learning, agent-based learning enables the study of the effect

of neuromodulators on the high level behaviour of the agent as tasks change. For example, in a visual navigation domain (e.g., maze or autonomous driving), an agent's behaviour as a result of neuromodulation can be observed directly from following its policy. This is loosely inspired by cognitive science literatures where biological neuromodulation and brain activity of rodents are studied in maze navigational problems.

1.3 Research Questions and Objectives

1.3.1 Research Questions

The thesis investigates the incorporation of neuromodulatory mechanisms into neural networks (referred to as neuromodulated networks) with the aim to facilitate the development of lifelong learning agents capable of rapid adaptation and learning in dynamic non-stationary environments. Such agents are capable to adapting to different tasks presented in an environment, and overcome the challenge of catastrophic forgetting. Therefore, the research from the thesis seek to answer the following questions;

1. Neuromodulators that regulate synaptic plasticity in Hebbian-based learning have been used to solve lifelong adaptation problems in environments with low dimensional input [251] in the past.
    (a) How can such networks be extended (directly or indirectly scaling them) to solve lifelong adaptation problems in partially observable environments with complex high dimensional visual observations?
    (b) What type of knowledge in such neuromodulated Hebbian-based networks is useful to facilitate rapid adaptation?
2. Neuromodulation can regulate neuronal or synaptic activities of a neural network.
    (a) How do we design and develop a simple yet modular computational framework that can support several training setup (i.e., gradient descent, neuroevolution, Hebbian-based plasticity and so on)?
3. In meta-reinforcement learning (meta-RL) framework, agents adapt their policy to change in task by fine-tuning their neural representations to the task. As the task/problem complexity increases and tasks becomes increasingly dissimilar standard meta-RL methods struggle.
    (a) What are the limitations of current methods as the task/problem complexity increases and tasks become increasingly dissimilar?

(b) What properties emerge from using neuromodulation to regulate neural activities in a meta-RL agent operating in complex task settings, and consequently does it lead to efficient adaptation in such settings?

4. In continual learning, parameter isolation methods via neuromodulatory masks have been devised to eliminate forgetting as the network parameters useful for solving different tasks are identified and kept fixed. When masks are directly optimised rather than derived from pruning, the method can learn many tasks without the limitations of the network capacity. Such approach have been largely devised for supervised learning and isolation methods generally lack the ability to re-use knowledge acquired from previous tasks (i.e., no forward transfer) since each mask is learned independently.

   (a) How can the learned neuromodulatory masking method be extended to reinforcement learning domain for lifelong learning?

   (b) What are the limitation and challenges of such extension to RL?

   (c) What mechanisms are necessary for the incorporation of forward transfer property in neuromodulatory mask?

   (d) Are there any tangible benefits gained from the combination of neuromodulatory masks to enable the forward transfer property in the learning system?

1.3.2 Objectives

To answer the research questions, the objectives of study are listed below.

- To conduct an extensive review of biologically inspired and mathematically defined neuromodulators in artificial neural networks.
- To develop a Hebbian-based neuromodulated network learning system, capable of facilitating rapid adaptation in high dimensional input setting.
- To develop a simple yet extensible framework to easy incorporation of neuromodulators into standard artificial neural network implementations.
- To develop a novel neuromodulated meta-learning system for reinforcement learning domains.
- To extend neuromodulatory masking continual learning methods to reinforcement learning.
- To develop a novel method to enable forward transfer properties in neuromodulatory masking methods.

1.4 Knowledge Contribution

The research reported in this thesis contributes to the growing understanding of the development and use of neuromodulatory strategies in neural networks, specifically in the lifelong RL domain. The experimental results produced to accomplish the research objectives showed that neuromodulation does influence the learning and adaptation capability of a lifelong RL agent. The contribution to knowledge in the field of lifelong learning and neural information processing via neuromodulation is outlined hereafter.

Neuromodulation has been shown to be effective for simple adaptation problems by regulating Hebbian-based plasticity in neural networks [251]. However, the use of such approach for complex partially observable environments containing high-dimensional RGB input has been previously unexplored. The first contribution addressed this problem through the development of a novel system that connected a feature extractor autoencoder network with the neuromodulated Hebbian-based network. The system featured fast and slow learning timescales, and a mix of plasticity paradigms (i.e., neuroevolutionary, gradient-descent, and Hebbian learning) working towards the achievement of the rapid lifelong adaptation objective. The empirical investigations led to the discovery of "inborn knowledge" in the system that is acquired through the slow learning process, and is subsequently exploited by the fast learning to facilitate adaptation. Thus, the introduced system enabled the extension of neuromodulated Hebbian-based neural controllers to high dimensional problems.

Meta-RL has gained prominence as a promising class of approach to tackle the lifelong adaptation problem, with several algorithms devised. These algorithms work by directly or indirectly finetuning the neural representations from a few input samples to produce input/output mapping in the policy network to solve each task. One key assumption is that the neural representations of the optimal policy across tasks share similarity. The relevance of such similarity in the effectiveness of the algorithms have not been studied extensively. Using two SoTA meta-RL algorithms, the second contribution empirically demonstrated that adaptation performance degrades in scenarios where diverse neural representations are required by the optimal policy for the different tasks. Further to this, the problem was addressed by introducing a neuromodulatory approach into the policy network of the algorithms. A novel neuromodulator was introduced that regulates the neural representations per layer through a flip operation that inverts the numeric sign of the activations. Using the computational framework, the policy networks that contained neuromodulation enabled rapid lifelong adaptation where diverse neural representations were required.

Having shown the benefits of neuromodulation in lifelong adaptation, the next contribution investigated the use of neuromodulation in lifelong learning, where tasks are learned sequentially. Neuromodulation is usually implemented as a mask that regulates the activations or weights of a neural network, as a means to eliminate forgetting. The masks can be derived either from iterative magnitude pruning of weights after learning a task or from the direct optimisation of mask parameters while the network weights are kept fixed. While the approach to derive masks from direct optimisation has been largely explored in lifelong supervised learning (LSL), their effectiveness in the more challenging lifelong RL domain was an open question. Also, masking approaches lack the ability to reuse knowledge (i.e., forward transfer) since each mask is derived independently. To address these challenges, the learned mask approach was extended to the RL domain. A novel method to facilitate knowledge reuse was introduced by linearly combining masks weighted by a learnable set of co-efficients parameters. The introduction of knowledge reuse enabled the masking agent to exploit existing knowledge and bootstrap learning when faced with a new task. The research findings showed that the learned mask approach is effective in the lifelong RL domain, and the knowledge reuse mechanism enabled agents to rapidly learn new tasks and solve compositionally complex tasks that could not be solved otherwise if learned from scratch. By learning the co-efficients, the agent is able to adjust the combination process, increasing the weight of a previously learned mask that is useful for solving the current task and vice versa.

Overall, the inclusion of neuromodulation into neural networks for lifelong learning and adaptation showed that it helped improve the learning system as the problem complexity increases and where relationships exist between tasks (task similarity).

1.5 Thesis Structure

The following chapters in this thesis is described as follow. Chapter 2 reviews relevant literatures and background useful for understanding the rest of the thesis. It reviews literatures across neuromodulation, neural networks, flavours of lifelong learning with respective methods.

Chapter 3 presents the first contribution of the thesis, reporting an investigation into the augmentation of Hebbian-based neuromodulated networks, extending them to high-dimensional setting for lifelong adaptation.

Chapter 4 presents the second contribution that reported the development of a modular neuromodulation computational framework and the introduction of a novel neuromodulator that explored lifelong adaptation via meta-reinforcement learning setup.

Chapter 5 presents the introduction of neuromodulatory mask for lifelong learning and the novel incorporation of forward transfer feature into the masking method.

The thesis concludes with Chapter 6 with discussions on future research direction.

1.6 Publications List

The investigation carried out in this thesis resulted in the following publication output.

- Evolving inborn knowledge for fast adaptation in dynamic POMDP problems. E. Ben-Iwhiwhu, P. Ladosz, J. Dick, W.-H. Chen, P. Pilly, and A. Soltoggio. In Proceedings of the Genetic and Evolutionary Computation Conference, 2020.

- Context Meta-Reinforcement Learning via Neuromodulation. E. Ben-Iwhiwhu, J. Dick, N.A. Ketz, P.K. Pilly, and A. Soltoggio. Neural Networks, 2022.

- Lifelong Reinforcement Learning with Modulating Masks. E. Ben-Iwhiwhu, S. Nath, P.K. Pilly, S. Kolouri, A. Soltoggio. Transactions on Machine Learning Research (TMLR), 2023.

1.6.1 Other Related Publications

Other publications which are related but are not captured in this thesis are listed below. The work carried out in these publications supported some of the necessary preliminaries that underpinned the advances introduced in the thesis.

1. Deep reinforcement learning with modulated Hebbian plus Q-network architecture. P. Ladosz, **E. Ben-Iwhiwhu**, J. Dick, N. Ketz, S. Kolouri, J. L. Krichmar, P. K. Pilly, and A. Soltoggio. IEEE Transactions on Neural Networks and Learning Systems, 2021.

2. A domain-agnostic approach for characterization of lifelong learning systems. M. M. Baker, A. New, M. Aguilar-Simon, Z. Al-Halah, S. M. R. Arnold, **E. Ben-Iwhiwhu**, A. P. Brna, E. Brooks, R. C. Brown, Z. Daniels, A. Daram, F. Delattre, R. Dellana, E. Eaton, H. Fu, K. Grauman, J. Hostetler, S. Iqbal, C. Kent, N. Ketz, S. Kolouri, G. Konidaris, D. Kudithipudi, E. Learned-Miller, S. Lee, M. L. Littman, S. Madireddy, J. A. Mendez, E. Q. Nguyen, C. D. Piatko, P. K. Pilly, A. Raghavan, A. Rahman, S. K. Ramakrishnan, N. Ratzlaff, A. Soltoggio, P. Stone, I. Sur, Z. Tang, S. Tiwari, K. Vedder, F. Wang, Z. Xu, A. Yanguas-Gil, H. Yedidsion, S. Yu, G. K. Vallabha. Journal of Neural Networks, 2023.

3. The configurable tree graph (CT-graph): measurable problems in partially observable and distal reward environments for lifelong reinforcement learning. A. Soltoggio, **E. Ben-Iwhiwhu**, C. Peridis, P. Ladosz, J. Dick, P. K. Pilly, S. Kolouri. arXiv preprint, 2023.

The publications were borne out of the Lifelong Learning Machines (L2M) DARPA programme, an international collaboration across academia and industry.

In the first publication, the work was carried out with a small group of scientist from the L2M programme. Supporting activities such as experimentation of baseline deep RL algorithms and the provision of comments in the early drafts of the paper were carried out. The experience acquired from the activities enabled the understanding of deep RL algorithms and relevant literatures, which serve as a foundation for the lifelong RL algorithms discussed and introduced in this work.

The second publication involved a wider collaboration with scientist across the entire L2M programme. Through active participation in discussions, the thesis author gained exposure in collaborating with a big pool of established scientist with different backgrounds. The research output played a key role in the understanding of what set of metrics are useful for evaluating lifelong learning agents. The metrics informed how the agents were evaluated and analyzed in the contributions discussed in this thesis.

The third publication focused on the development of an RL simulation environment that is suitable for the assessment lifelong learning agents, with tasks that varies in complexity. The thesis author contributed to this work via code revision and maintenance, and the discussions of conceptual ideas for the paper. The outcome of the work provided a useful platform for the evaluation of the algorithms introduced in this thesis.

Chapter 2

Background: Lifelong Learning, Adaptation, Neuromodulation and Neural Models

This chapter covers relevant background in the field of lifelong learning, adaptation and neuromodulation. The field of neural intelligent systems and lifelong learning have been growing expansively with inspiration drawn from biological and neuroscience. The discussion and review presented in this chapter does not exhaustively cover the field but however provides enough contextual information relevant for the investigations carried out in this thesis.

Lifelong learning is a framework in machine learning that is focused on building intelligent systems capable of learning and adapting to several task and task conditions over its lifetime. The two main learning paradigms of lifelong learning are: (i) continual learning, where tasks are learned in sequence with focus on maintaining or improving performance on previously learned task, knowledge reuse, and avoidance of catastrophic forgetting, (ii) meta learning, with focus on rapid adaptation to tasks. Other related paradigm include multi-task learning, transfer learning, domain adaptation, out of distribution generalisation and so on.

The lifelong learning framework is applicable across supervised, self-supervised (unsupervised) and agent-based (reinforcement) learning domains. The scope of this thesis capture lifelong learning operations within agent-based learning domain. Therefore, a background in agent-based learning is first established in the section below before delving into lifelong learning background and theories.

2.1 Agent-based (Reinforcement) Learning

In this section, the reinforcement learning (RL) problem is first defined followed by theory and methods.

2.1.1 Reinforcement Learning: The Problem Definition

The standard reinforcement learning problem [267] is defined as markov decision process (MDP) containing a tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ the environment dynamics or transition

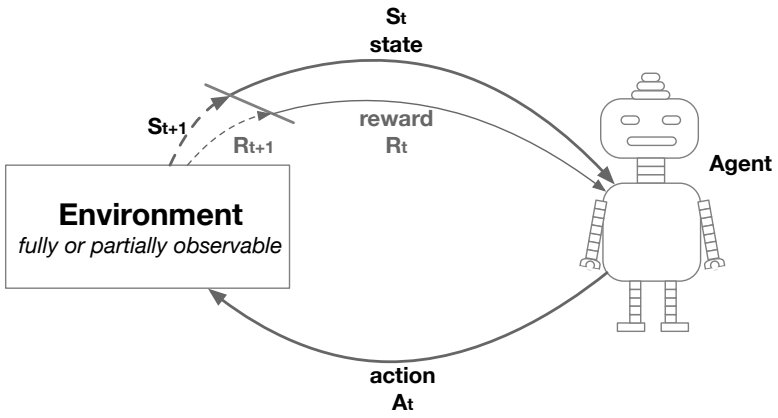


Figure 2.1: The reinforcement learning problem setup

probability distribution $p(s_{t+1}|s_t, a_t)$ of the next state given the current state and action taken at time t, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ the reward function $r(s_t, a_t)$ that produces a scalar value based on the state s_t and the action a_t taken at the current time step t, $\gamma \to [0, 1]$ is the discount factor that determines how importance future reward relative to the current time step t. An agent interacting in the MDP behaves based on a defined policy (either a stochastic π or a deterministic μ policy).

In a situation where the information from the environment is incomplete or partially observable, the MDP can be extended to a partially observable MDP (POMDP), with tuple defined as $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \gamma, \mathcal{O}, \Omega \rangle$. The added parameters to the tuple include $\mathcal{O}$ the observation set, and Ω the function that maps observations to states. The partially observability poses a greater to challenge to the learner as it needs to figure out its relative position in the environment, based on its history of actions taken till the current time.

As opposed to supervised learning that contains strong supervisory signal from every prediction of the learner during training, the supervisory signal for RL comes from the reward which is weak and may not contain useful information for every prediction. Therefore, the challenges of RL also include credit assignment, where the system learn to figure out which action either from a present or distant past led to good outcomes (i.e., higher reward).

Depending on the experimental setup, the agent either interacts with the environment for a finite number of time steps, after which a reset to the environment or an infinite horizon time steps, without a resets. The sum of rewards acquired by the agent for a batch of data collected is referred to as a return G. The return G can be discounted using the discount factor γ. When discounted return is employed, a trade-off prior between the focus on near term or long term reward is incorporated into the system, thus leading to instant versus delayed gratification behaviours respectively. In any given task, the goal or objective of an agent learner

is to maximize the amount of rewards accumulated. This is formalised as maximising the expected discounted returns $G = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t)]$ or undiscounted returns $G = \mathbb{E}[\sum_{t=0}^{\infty} r(s_t, a_t)]$. It achieves this by learning an optimal policy.

In a lifelong learning setup [128], a lifelong RL agent is exposed to a sequence of tasks $\mathcal{T}$. Given N tasks, the agent is expected to maximize the RL objective for each task in $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$. As the agent learns one task after another in the sequence, it is required to maintain (avoid forgetting) or improve (backward transfer of knowledge) performance on previously learned tasks. A desired property of such an agent is the ability to reuse knowledge from previous tasks to rapidly learn the current or future tasks.

2.1.2 Reinforcement Learning: Theory and Methods

To develop intelligent agents for the RL problem, several theories and methods have been developed. Recent literatures employ neural networks as a universal function approximator for methods being developed. Recent advancements include the development of agents to play video games [189, 187, 274, 245], complex strategy games [246, 247, 271, 30], for navigation [65, 88, 144], for robotics [233, 156, 235, 89, 75], electronics chip design [184, 309] and so on.

These method are focused on solving a single task in RL environment, maximising the expected accumulated rewards. They are trained using either a gradient-based [235, 156, 75] or gradient-free [264, 226, 218] methods as discussed later in Section 2.4. The algorithms are driven by a policy that produces the behaviour of the agent. A policy can either be deterministic (referred to as μ), with deterministic output or stochastic (referred to as π), with output sampled from a probability distribution. The choice of the type of policy is dependent on the algorithm.

RL algorithms can be categorized based on certain features. The main features which defines three categorisation include: (i) model-based or model-free, (ii) on-policy or off-policy, and (iii) direct policy (actor) optimization, value (critic) function optimization or hybrid (actor-critic) optimization.

For the model-based/model-free categorisation, an algorithm is considered model-based [74, 199, 12, 110, 88, 91, 319, 313, 94, 181] if it learns a model of the world (for example, the transition dynamics $\mathcal{P}$ of an environment). The learned dynamics then aids the agent in reasoning, planning and driving the agent's policy. Model-free algorithms [285, 189, 233, 187, 235, 156, 75, 89] solves the RL task by learning to directly map states to optimal actions, without using any model of the world. In a neural learner setup, the model-free algorithm is viewed as learning a function mapping input to output. Through reasoning and planning, model-based algorithms learns to solve the RL task faster than model-free algorithms.

On-policy algorithms optimize a policy using data directly acquired by the policy. Examples include Policy Gradient (PG) [285], Trust Region Policy optimization (TRPO) [233], Proximal Policy optimization (PPO) [235], and more. Off-policy algorithms on the other hand optimize its policy using data acquired by both the policy and other policies (i.e., data collected either from a different RL policy or an earlier versions of the same policy). Examples include Deep Q Network (DQN) [189], Deep Deterministic Policy Gradient (DDPG) [156], Soft Actor Critic (SAC) [89], and more. They are considered more sample efficient towards learning the given RL task in comparison to on-policy algorithms, but on-policy training tends to be more stable with respect to hyper-parameter selection and tuning.

Direct policy optimization algorithms [285, 233, 235, 226, 264] optimize the policy directly for the goal of maximising the expected returns. Value optimization algorithms [189, 274, 103] maximize the expected return indirectly by learning a value function that estimates or predicts the expected future return, which is used to define the policy. Actor-critic algorithms [187, 14, 75, 89] combine both the policy and the value function optimization, using the value prediction to correct or guide the actions taken by the actor.

2.2 Lifelong Agent-based Learning and Adaptation

Lifelong agent-based learning is a branch of lifelong learning where the emphasis is on the development of a lifelong learner (agent) for a reinforcement learning (RL) problem domain. It follows the RL problem setup [267] (see Figure 2.1). In the lifelong learning setup, the RL problem is posed as learning multiple environments or multiple tasks in a single environment. For each task presented, the agent explores the environment, collecting observations from the environment based on actions taken. The collection of data by the agent, rather than being fed hand crafted data makes the field different from it's supervised learning counterpart. Other differences are discussed in Section 2.3.

2.2.1 Continual Learning

The continual learning paradigm seeks to address the challenge of developing a lifelong learning system that learns tasks in sequence. Such a system under the learning paradigm needs to address several sub challenges such as ensuring that catastrophic forgetting is eliminated or reduced, performance in previously learned task is maintained or improved (backward transfer), knowledge from previous task is reused to accelerate learning (forward transfer) in new tasks, the system's model capacity is efficiently used to reduce intransigence effects, and interference in the knowledge acquired is reduced or avoided across tasks.

Several assumptions are considered when building the lifelong learner under this learning paradigm. For example, the data or input stream is assumed to be non-stationary as the agent is exposed to data from several tasks. Also, another assumption to consider is whether or not an task oracle is present to inform the lifelong learner about the task boundaries between tasks and identifiers (label) given to each task. Without a task oracle, the system is given the additional challenge to infer when a task change occur and give labels to tasks if necessary. Task detection is one of the sub-fields growing out of continual learning.

Several advances have recently been introduced in continual learning [273, 203, 90, 52, 128], addressing the aforementioned challenges. A large body of work focused on lifelong learning in the supervised learning domain and overcoming the challenge of forgetting [178]. Continual learning algorithms can be clustered into key categories such as synaptic consolidation approaches [132, 303, 4, 136], memory approaches [167, 302, 37, 219, 158], modular approaches [224, 174, 173, 287, 178] or a combination of the above. Synaptic consolidation approaches tackle lifelong learning by discouraging the update of parameters useful for solving previously learned tasks through a regularisation penalty. Memory approaches either store and replay samples of previous and current tasks (from a buffer or a generative model) during training [167, 37, 86, 279] or project the gradients of the current task being learned in an orthogonal direction to the gradients of previous tasks [302, 66, 225, 158]. Memory methods aim to keep the input-output mapping for previous tasks unchanged while learning a new task. Modular approaches either expand the network as new tasks are learned [224, 297] or select sub-regions of a fixed-sized network (via masking) to learn tasks [287].

In the RL domain, a lifelong learner under the continual learning paradigm is usually developed by combining a standard deep RL algorithm, either on-policy or off-policy (for example, DQN, PPO, or SAC), with a continual learning algorithm. Examples of such lifelong RL algorithms include: (i) CLEAR [219], a lifelong RL that combines IMPALA RL algorithm [65] with a replay method and behavioural cloning, and (ii) Progress & Compress [238] algorithm which demonstrated the combination of IMPALA with elastic weight consolidation (EWC, synaptic consolidation algorithm) [132] and policy distillation techniques. Other notable methods include the combination of a standard deep RL agent (e.g., SAC algorithm) with PackNet (parameter isolation algorithm) [174] as demonstrated in the Continual World robotics benchmark [286]. In addition, [178] developed an algorithm combining neural composition, offline RL [154], and PPO RL algorithm to facilitate knowledge reuse and rapid task learning via functional composition of knowledge. To tackle a lifelong learning scenario with interference among tasks, current methods [126, 127] employ a multi-head policy network (i.e., a network

with shared feature extractor layers connected to different output layers/heads per task) that combine a standard RL algorithm (e.g. DQN or SAC) with synaptic consolidation methods (e.g., EWC).

2.2.2 Meta-Learning

Meta-learning is a learning paradigm that is focused on automating the learning process and building intelligent systems that *learns how to learn*. Existing meta-learning literatures dates back to the 80s and 90s [232, 231, 270, 27, 255, 112]. Meta-learning has been employed to learn various learning components or building blocks, with the goal of developing better solutions in comparison to hand crafted ones. Studies include learning an optimizer [7], loss function [311, 24, 19], neural architecture [163, 301], task adaptation or few shots learning [69, 314, 209, 305, 304, 149] and so on. For the scope of this thesis, the review of meta-learning is focused on its application to adaptation.

Adaptation can be viewed as a specific form of learning where a learning system needs to rapidly learn the task given. The focus is on the speed of learning and it is sometimes referred to as few-shots learning. In the lifelong adaptation setting, the system would be required to adapt to both previously seen and unseen tasks. In this setup, there exist a task distribution from which tasks are continuously sampled for the system to rapidly learn. Over the course of the learning, the system aims to learn the underlying task distribution, which enables it to adapt to new tasks sampled from the distribution. Several studies exist that have investigated lifelong adaptation via meta-learning in domains such as supervised learning [134, 229, 248, 211, 193, 69, 151, 298, 314], unsupervised learning [278, 214, 63, 113, 179, 129] and reinforcement learning (RL) [69, 60, 87, 221, 314, 209]. Since the focus of this thesis is on agent-based (RL) lifelong adaptation, the review is further scoped to meta-learning in the reinforcement learning (RL) domain, also referred to as meta-reinforcement learning (meta-RL).

Meta-RL

Recent studies in meta-RL can be largely classified into optimization, context-based and hybrid methods.

Optimization methods seek to learn good initial parameters of a model that can be adapted with a few learning steps to a specific task. The neural network systems in this class are setup to contain two parameter update loops or levels, where the outer loop learns a good initial condition for the network's parameters and the inner loop quickly learns (adapt) the optimal parameters for the given task. The optimization can be gradient-based [69, 155, 257, 221] or gradient-free [251, 253, 256, 194] or a combination of both. The gradient-free approach embodies a setup where neuroevolution is employed in both learning loops [68, 256], or

neuroevolution employed in the outer loop and Hebbian-based learning employed in the inner [251, 194].

In contrast, context-based methods seek to adapt a model to a specific task based on few-shot experiences aggregated into context variables. The context can be derived via probabilistic methods [209, 161], recurrent memory [60, 282], recursive networks [185] or the combination of probabilistic and memory [315, 115].

Hybrid methods [314, 87] combine optimization and context-based methods whereby task specific context parameters are obtained via gradient updates. The adapted context parameters then serve as a useful signal to the learning system during adaptation.

2.2.3 Multi-Task Learning

Multi-task learning [35] is a learning paradigm loosely related to lifelong learning. The goal of a multi-task learner system is to learn a fixed number of multiple tasks jointly. After the tasks have been learned, the training stops and the agent is deployed. In a setup where the multi-task learner is a neural system, the training across multiple tasks reduces overfitting and enables the creation of a shared reusable representation across tasks [44]. Similar to meta-learning and continual learning, multi-task learners have been in deployed across supervised [48, 186, 171, 222, 78, 263, 44] and reinforcement learning [101, 54, 117] domains. In continual learning literatures where n tasks needs to be learned sequentially, the multi-task equivalent that learns the n tasks jointly is sometimes used as the upper bound for the continual learner system. However, a recent study [289] raised concern on the account of its suitability when the tasks to be jointly learned are interfering or adversarial to one another. Such a task curriculum would cause the multi-task learner to struggle during learning and may hurt the overall performance of the system.

A key theme of central importance to MTL is knowledge sharing across tasks, with varying degree of sharing. Acquired knowledge in MTL tends to be split into task specific and shareable (general) knowledge. The motivation behind sharing is that knowledge acquired in one task could be useful in other tasks to accelerate learning and subsequently improve the overall performance of the multi-task learner. In neural approaches, sharing can be defined at the neural representations level [262, 265, 293, 275], network parameters [306, 164, 186, 294] or modular level [101, 54, 180, 266]. Sharing via neural representations can be setup using a distillation framework [165, 293, 307, 275] or a masking approach [263, 265]. In shared network parameters methods, the shareable parameters are either manually defined in the architectural design [306, 48, 308, 171] or autonomously learned from data [186, 294, 78, 222].

2.3 Task Variations in Lifelong Agent-Based Learning

Due to the nature of the lifelong learning framework containing multiple tasks in sequence, it is important to describe the underlying variation across tasks. In lifelong supervised learning domain, changes between tasks are largely based on differences in the input distribution. For example, in the permuted MNIST problem [84], each task has an input distribution different from others. However, in the domain of lifelong reinforcement learning (our focus), variations in task can occur across one or a combination of the following factors: (i) input (state $\mathcal{S}$ or observation $\mathcal{O}$) distribution, (ii) reward function r, (iii) transition function p.

A task curriculum (i.e., order of tasks to be learned) can be designed to contain only one factor of variation across tasks (e.g., change only in reward function) or a combination of them. Given two consecutive tasks τ_k and τ_{k+1} in a curriculum, the effect of the factors of variation on the policy network are described below.

Variation only in the reward function: In this setting, a lifelong RL agent should be capable of capturing such variation, and changes to the policy network are likely to occur in the output layer. However, in this problem setting, such an agent may encounter interference [127] of knowledge across both tasks. Such interference arise from a scenario where the optimal action for a given state s may be different between τ_k and τ_{k+1} due to different rewarding schemes. Existing solutions to this problem include the use of multi-head output layer (one output head per task), the concatenation of unique task labels to input states (to different states of one task from another) or a combination of both. An example of this problem setting is the change in goal location of a Minigrid [38] environment, or change in the optimal position of the pendulum swing problem by setting orthogonal angles between tasks [126].

Variation only in the state distribution: In this setting, the agent is required new set of input-output mappings without affecting mappings learned for previous task. An example of such setting in lifelong RL is the curriculum of six Atari games employed in [238, 219, 205], where each task (or game) contains different state/observation distribution. Hence, concatenating a task label states is not necessary in this setting.

Variation only in the transition distribution: This is the less common of the three factors of task variation. In this setting, the state-action pairs required to reach a desired state would have changed. It is worth noting that when the input distribution changes, it also automatically changes the transition distribution across tasks.

In general, a good lifelong RL agent should be endowed with the capabilities of capturing these factor of variations in knowledge learned, understanding

similarities across learned tasks and efficiently re-using knowledge to learn future task.

2.4 Network optimization and Plasticity

In addition to the lifelong learning paradigms, it is important to consider how the neural lifelong learner is optimized and the timescales of plasticity (parameter or synaptic updates) in the system. Learning happens in the system by adjusting the parameters or architecture of the neural network to satisfy a defined objective. Several optimization and plasticity strategies exist in literatures. However, the focus is on the learning strategies employed in the methods introduced in this thesis. They are discussed in the Sections 2.4.1, 2.4.2, and 2.4.3 below.

2.4.1 Stochastic Gradient Descent and Backpropagation

Stochastic Gradient Descent (*SGD*) is a general purpose gradient-based optimization method for artificial neural networks. It is based on the use of the gradients of parameters, which are computed via backpropagation (*backprop*) [284, 223], an algorithm based on the application of calculus chain rule. *SGD/backprop* is most popular method for optimising modern deep ANNs, and widely applicable across supervised, semi-supervised, self-supervised and reinforcement learning domains. With a global objective defined, plasticity and learning is achieved through the recursive updates to the parameters of the network, by using the gradients of the parameters with respect to the objective. The gradients are scaled by a defined learning rate hyper-parameter before it is used for the update. By computing gradients, *SGD/backprop* offers the ability to administer updates specific to each parameter and pass error information from later layers back to earlier layers of the network [157].

The computation of the gradients in a high dimensional parameter search space leads to challenges, due to complex and sensitive nature of the search landscape, which can be non-convex depending on the problem domain. The major challenges include (i) the vanishing gradient problem, where the computed gradients are close or equal to zero, thus inhibiting parameter updates, and (ii) the exploding gradient problem, where the computed gradients become very large numbers, thus destabilising the learning process in the neural network. Additionally, the updates by *SGD* could lead the neural network to get stuck at a local minimal in the search space, and thus produce sub-optimal solutions. These challenges are addressed by introducing additional techniques and tricks that are combined with *SGD/backprop*. Examples of such additional techniques include the use of regularizers to penalize large gradients [105, 190, 142, 172, 195], better initialisation strategy [81, 230, 98], normalisation [116, 13, 133, 288], momentum [196, 160, 47],

adaptive learning rates [61, 104, 130, 317] and so on. Lastly, *SGD/backprop* requires the search landscape to be continuous and differentiable, otherwise learning would not be possible due to zero gradients.

In RL, the combination of *SGD/backprop* and deep neural networks has led to an explosive growth in the field, demonstrating super-human performance in video games [188, 189, 187], board games [246, 247] and robotic control [233, 156, 75, 89]. In the framework, *SGD/backprop* is used to directly optimize the parameters of the neural network.

2.4.2 Neuroevolution

Evolutionary computation as an optimization strategy for solving numerical problems date back to the 1950s and 60s [73, 32, 111]. It is a gradient-free approach where the optimal solution to the problem is derived by loosely simulating biological evolutionary processes within a computational framework. To solve a defined objective, a population of candidate solutions is initialized and iteratively optimized over several generations. It involves the repetition of one or more processes such as fitness evaluation of candidate solutions in the current population with respect to the specified objective, selection of the best candidates, cross-over of selected candidates, and the mutation of candidates in the newly produced population. These processes are repeated until an optimal solution is found. Several approaches such as genetic algorithm [111, 51, 64, 83], genetic programming [139, 145, 3], and evolution strategies [212, 240, 201, 95] have been popularized and applied across a wide range of problem settings [15].

In artificial neural networks, the optimization of a neural system via an evolutionary algorithm is referred to as *neuroevolution* [295, 70, 258]. Areas of optimization in the neural system via evolution include parameter optimization of a fixed neural architecture, architectural optimization [260, 77], network compression [50, 79, 259], behavioural novelty [253, 152, 153], robot skill diversity [192, 46] and so on. A major concern for evolutionary algorithms in neural network is their ability to scale as the number of parameters increase (i.e., more complex or deeper networks).

In the past, neuroevolution methods have been employed to solve RL tasks [260, 177, 251, 31] in environments with high level input features. In such setting, shallow networks containing a small number of parameters were employed. Recently, several approaches have been introduced that combine deep neural networks and neuroevolution to tackle high dimensional deep RL tasks [6, 204, 88, 226, 264]. For example, [216] showcased the evolution of a model-based RL system end-to-end, from perception to control. Also, [226] demonstrated the optimization of neural networks for complex continuous control robotics tasks.

The neuroevolution approaches in RL can be divided into two major categories. The first category uses neuroevolution to optimize the entire deep network end to end [226, 264, 216, 217]. The second category splits the network into parts (for example, a body and controller) where some part(s) (e.g. body) are optimized using gradient based methods and other part(s) (e.g. controller) are evolved using neuroevolution methods [6, 204, 88]. Current deep neuroevolution methods are usually evaluated in fully observable MDP environments, where the task is fixed. Furthermore, after the training phase is completed, the weights of a trained network are fixed (the same is true for standard deep RL). The recent attention to neuroevolution for deep RL aims to present such approaches as a competitive alternative to standard gradient based deep RL methods for fixed task problems.

2.4.3 Local Synaptic Plasticity

Another approach to parameter/synapse updates in a neural system is the use of local synaptic plasticity methods. The focus is on the use of local information to update each parameter, rather than rely on a global objective. For each parameter, the local information is defined by the activities of the neurons it connects. The neurons are referred to as pre-synaptic and post-synaptic neurons.

The inspiration for this category of synaptic plasticity method stems from the Hebb's rule [100] in neuroscience, which loosely translates as *neurons that fire together wire together*. In ANNs, the update is computationally defined as the product (multiplication) of neural activations of the pre and post-synaptic neurons of each parameter. This means that the value of a parameter (synaptic strength) should increase during updates if its pre and post-synaptic neurons possess output that activates in the same (numerical) sign direction and vice versa. This formulation leads to unstable parameters, since they can that grow infinitely large (either positively or negatively) due to the repeated multiplicative delta (update) term applied to update them. Therefore, several extensions have been developed to combat this issue. One class of approach [200, 228] performs L2 normalization on the delta term, scaling down the values before they are used to update the network's parameters.

Local synaptic learning has been applied across supervised learning [290, 17, 18, 53] and reinforcement learning domains [251, 277, 182, 194]. In the lifelong adaptation settings, where plasticity can occur at multiple time scales, the local synaptic methods have been employed [251, 255, 194]. In such setting, neuroevolution or SGD is used as the high level optimizer to learn an initial parameter condition for the neural system, while a local synaptic method is used at the task specific level to quickly finetune the network parameters to foster rapid task adaptation.

2.5 Neuromodulation

2.5.1 Biological Neuromodulators

Neural network in biological brains is composed of neurons intricately connected and extended across the nervous system in the body. A connections between two neurons occur at the synapse, a space between where excitatory or inhibitory signals (neurotransmitters) are released from one neuron to the other. The release of such signal occur when an action potential (i.e., polarisation of the electro-chemical charges in the neuronal state) is reached in the pre-synaptic neuron. Apart from the excitatory and inhibitory signals, existing evidence [49, 125, 21, 175, 169] suggests that other types of signals are also communicated via neurotransmitters. These special signals, referred to as modulatory signals modulates or regulates the neuronal behaviour and function of a post-synaptic neuron. Neurons that release modulatory neurotransmitters are called modulatory neurons or neuromodulators.

Generally, neuromodulation in biological brains is a process whereby a neuron alters or regulates the properties of other neurons [175]. The altered properties can either be in the cellular activities or synaptic weights of the neurons. Some examples of neuromodulators identified in mammalian brains include dopamine (DA), serotonin (5-HT), acetycholine (ACh), and noradrenaline or norepinephrine (NA) [11, 269, 22]. These neuromodulators impact the high level cognitive behaviours and functions, and several theories have been developed about the role they play. Dopamine is particularly linked to reward error prediction and economic utility (representation of subjective value derived from choices under risk) [236, 261, 237], acetycholine drives top-down goal driven attention mechanisms [11, 318] and memory retrieval/consolidation [96, 198], serotonin particularly influences harm aversion, impulsivity and anxiety behaviours [241, 268, 45, 244, 9], and noradrenaline influences the detection of salient inputs and network reconfiguration or short term memory resets [11, 269, 318].

Despite the existing theories, a thorough understanding of neuromodulatory systems has not been fully achieved. Studies into the cross-functional role they play across the brain and how the neuromodulators complement or oppose each other are necessary to develop more robust theories. Nonetheless, inspiration can loosely derived from biological neuromodulators with the goal of applying them in artificial neural networks either to regulates neuronal activation patterns or regulate synaptic activities [175].

2.5.2 Neuromodulators in Artificial Neural Networks

Neuromodulatory processes [2, 11, 22] enable the dynamic alteration of neuronal behaviour by affecting synapses or the neurons connected to synapses. Neur-

omodulation in artificial neural networks [236, 67, 58] draws inspiration from modulatory dynamics in biological brains. In general, neuromodulation has been applied across supervised learning[182, 318, 23, 287] and reinforcement learning [58, 251, 182, 291, 292] domains. Existing studies use neuromodulation to regulate plasticity [251, 277, 182, 287] for adaptation, and regulate neuronal activation for continual classification [23], with varying designs of modulatory functions. Other examples include the use of neuromodulation to alter high-level behaviour (policy) of agents to control patience level [291] or task detection [292, 318].

The seminal work by [58] presents a description about how the biological neuromodulators (discussed in Section 2.5.1) maps to an agent in the RL computational framework. Dopamine loosely mapped to the reward signal error derived using the reward prediction output from an agent's value function. Serotonin represented the discount factor which helps the agent decide whether to act impulsively and collect immediate reward or take actions based on a long term strategy to amass more rewards in the future. Acetycholine represented the learning rate of the agent, highlighting how fast or slow knowledge is encoded into the synapse or long-term memory. Noradrenaline represented the randomness in the agent's policy action distribution. Drawing inspiration from this framework, several studies have applied the different neuromodulators to gradient-based RL [182, 291, 292] and neuroevolutionary RL [252, 251, 277] for adaptation settings.

Chapter 3

Evolving Inborn Knowledge For Fast Adaptation in Dynamic POMDP Problems

This chapter presents the first original contribution of the research reported in the thesis. A lifelong adaptation system was developed that exploited the highly adaptive nature of neuromodulated plasticity in networks, to evolve a neural controller that uses the latent space of an autoencoder in partially observable environments. The analysis of the evolved networks reveals the ability of the proposed algorithm to acquire inborn knowledge (i.e., knowledge encoded into a network structure that is invariant to change in task conditions and facilitates learning) in a variety of aspects such as the detection of cues that reveal implicit rewards, and the ability to evolve location neurons that help with navigation. The integration of inborn knowledge and online plasticity enabled fast adaptation and better performance in comparison to some non-evolutionary meta-reinforcement learning algorithms. The algorithm proved also to succeed in the 3D gaming environment Malmo Minecraft.

3.1 Introduction

The field of deep reinforcement learning (RL) has showcased amazing results in recent time, solving tasks in robotic control [59, 156], games [189] and other complex environments. Despite such successes, deep RL algorithms are sample inefficient and sometimes unstable. Furthermore, they usually perform sub-optimally when dealing with sparse reward and partially observable environments. One further limitation of deep RL is when rapid adaptation to changing tasks (dynamic goals) is required. Established methods only work well in fixed task environments. In an attempt to solve this problem, deep meta-reinforcement learning (meta-RL) methods [69, 221, 314, 60, 282] were specifically devised. However, these methods (especially optimization approaches [69, 314]) are largely evaluated on dense reward, fully observable MDP environments, and perform sub-optimally in sparse reward, partially observable environments.

One key aspect in achieving fast adaptation in dynamic partially observable environments is the presence of appropriate learning structures and memory units that fits the specific class of learning problems. Therefore, standard model-free RL algorithms do not perform well in dynamic environments because they are tabula-rasa systems. They hold no knowledge in their architectures to allow a fast and targeted learning when a change in the environment occurs. Upon a task change, these algorithms will try to randomly explore the action space to relearn from scratch a different, new policy. On the other hand, model-based RL, holds knowledge of the structure of the environment, which in turn allows for rapid adaptation to changes in the environment, but such a knowledge needs to be built manually into the system or learned [92, 118, 93]. When learned, such models are brittle and require many sample frames to capture the environment dynamics.

In this chapter, we investigate the use of neuroevolution to autonomously evolve inborn knowledge [255] in the form of neural structures and plasticity rules with a specific focus on dynamic POMDPs that have posed challenges to current RL approaches. The neuroevolutionary approach that we propose is designed to solve rapid adaptation to changing tasks [255] in complex high dimensional partially observable environments. The idea is to test the ability of evolution to build an unconstrained neuromodulated network architecture with problem-specific learning skills that can exploit the latent space provided by an autoencoder. Thus, in the proposed system, an autoencoder serves as a feature extractor that produces low dimensional latent features from high dimensional environment observations. A neuromodulated network [251] receives the low dimensional latent features as input and produces the output of the system, effectively acting as high level controller. Evolved neuromodulated networks have shown computational advantages in various dynamic task scenarios [251, 255].

The proposed approach is similar to that proposed in [6]. One key novelty is that our approach seeks to evolve selective plasticity with the use of modulatory neurons, and therefore, to evolve problem-specific neuromodulated adaptive systems. The relationships among image-pixel inputs and control actions in POMDPs is highly nonlinear and history dependent, therefore, an open question is whether neuroevolution can exploit latent features to evolve learning systems with inborn knowledge. Thus, we test the hypothesis that a neuromodulated evolved network can discover neural structures and their related plasticity rules to encode required memory and fast adaptation mechanisms to compete with current deep meta-RL approaches.

We call the proposed system a Plastic Evolved Neuromodulated Network with Autoencoder (PENN-A), denoting the combination of the two neural components. We evaluate our proposed method in a POMDP environment where we show bet-

ter performance in comparison to some non-evolutionary deep meta-reinforcement learning methods. Also, we evaluated the proposed method in the Malmo Minecraft environment to test its general applicability.

Two interesting findings from our experiments are that (i) the networks acquire through evolution the ability to recognise reward cues (i.e. environment cues that are associated with survival even when reward signals are not given) and (ii) the networks can evolve *location* neurons that help solving the problem by detecting, and becoming active at, specific location of the partially observable MDP. The evolved network topology allows for richer dynamics in comparison to fixed architectures such as hand-designed feed-forward or recurrent networks.

The next section reviews the related work. Following that, a formal task definition is presented. Next is the description of the proposed method employed in this work, followed by the evaluation of results. The PENN-A source code is made available at: https://github.com/dlpbc/penn-a.

3.2 Background

In reinforcement learning (RL) literature, meta-RL methods seek to develop agents that adapt to changing tasks in an environment or a set of related environments. Meta-RL [231, 242] is based on the general idea of meta-learning [27, 270, 107] applied to the RL domain.

Recently, deep meta-RL has been used to tackle the problem of rapid adaptation in dynamic environments. Methods such as [69, 60, 282, 314, 185, 221, 209] use deep RL methods to train a meta-learner agent that adapts to changing tasks. These methods are mostly evaluated in dense reward, fully observable MDP environments. Furthermore, most methods are either memory based [60, 282, 185] or optimization based [69, 314]. Optimization based methods seek to find an optimal initial set of parameters (e.g. for an agent network) across tasks, which can be fine-tuned with a few gradient steps for each specific task presented to it. Therefore, a small amount of re-training is required to enable adaptation to every change in task. Memory based methods (implemented using a recurrent network or temporal convolution attention network) do not necessarily require fine tuning after initial training to enable adaptation. This is because memory-based agents learn to build a memory of past sequence of tasks and interactions, thus enabling them to identify change in task and adapt accordingly.

In the past, neuroevolution methods have been employed to solve RL tasks [260, 177], including adapting to changing tasks [251, 31] in partially observable environments. These methods were evaluated in environments with high level feature observations. Recently, several approaches have been introduced that combine deep neural networks and neuroevolution to tackle high dimensional deep RL tasks

[6, 204, 88, 226, 264]. These approaches can be divided into two major categories. The first category uses neuroevolution to optimize the entire deep network end to end [226, 264, 216, 217]. The second category splits the network into parts (for example, a body and controller) where some part(s) (e.g. body) are optimized using gradient based methods and other part(s) (e.g. controller) are evolved using neuroevolution methods [6, 204, 88]. Current deep neuroevolution methods are usually evaluated in fully observable MDP environments, where the task is fixed. Furthermore, after the training phase is completed, the weights of a trained network are fixed (the same is true for standard deep RL). The recent attention to neuroevolution for deep RL aims to present such approaches as a competitive alternative to standard gradient based deep RL methods for fixed task problems.

In the past, neural network based agents employing Hebbian-based local synaptic plasticity have been used to achieve behavioural adaptation with changing tasks [71, 31, 251]. Such methods use a neuroevolution algorithm to optimize the parameters of the network when producing a new generation of agents. As an agent interacts with an environment during its lifetime in training or testing, the weights are adjusted in an online fashion (via a local plasticity rule), enabling adaptation to changing tasks. In [71, 31] this technique was employed, and further extended to include a mechanism of gating plasticity via neuromodulation in [251]. These methods were evaluated in environments with low dimensional observations (with high level features) and not compared with deep (meta-)RL algorithms.

3.3 Task Definition

A POMDP environment E, defined by a sextuple $(\mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R}, \mathcal{O}, \Omega)$ is employed in this work. $\mathcal{S}$ defines the state set, $\mathcal{A}$ the action set, $\mathcal{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \rightarrow [0, 1]$ the environment dynamics, $\mathcal{R} : \mathcal{S} \times \mathcal{A} \rightarrow \mathbb{R}$ the reward function, $\mathcal{O}$ the observation set, and Ω the function that maps observations to states.

The environment E contains a number of related tasks. A task $\mathcal{T}_i$ is sampled from a distribution of tasks $\mathcal{T}$. The task distribution $\mathcal{T}$ can either be discrete or continuous. A sampled task is an instance of the partially observable environment E. The configuration of the environment (for example, the goal or reward function) varies across each task instance. An optimal agent is required to adapt its behaviour to task changes in the environment (and maximize accumulated reward), only from few interactions in the environment. When presented with a task $\mathcal{T}_i$, an optimal agent should initially explore, and subsequently exploit when the task is understood. When the task is changed (a new task $\mathcal{T}_j$ sampled from $\mathcal{T}$), the agent needs to re-explore the environment in few-shots, and then to start exploiting again when the new task has been understood.

In each task, an episode is defined as the trajectory τ of an agent interactions

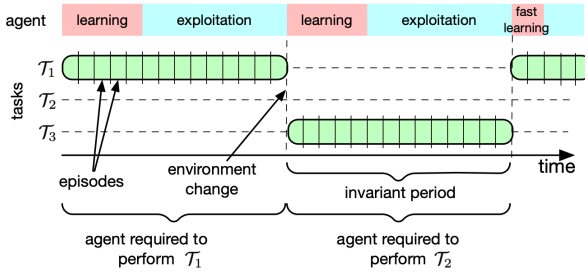


Figure 3.1: Illustration of a dynamic environment and required behaviour of a learning agent. An agent is required to learn to perform optimally and then exploit the learned policy until a change in the environment occurs, at which point the agent needs to learn again before exploiting.

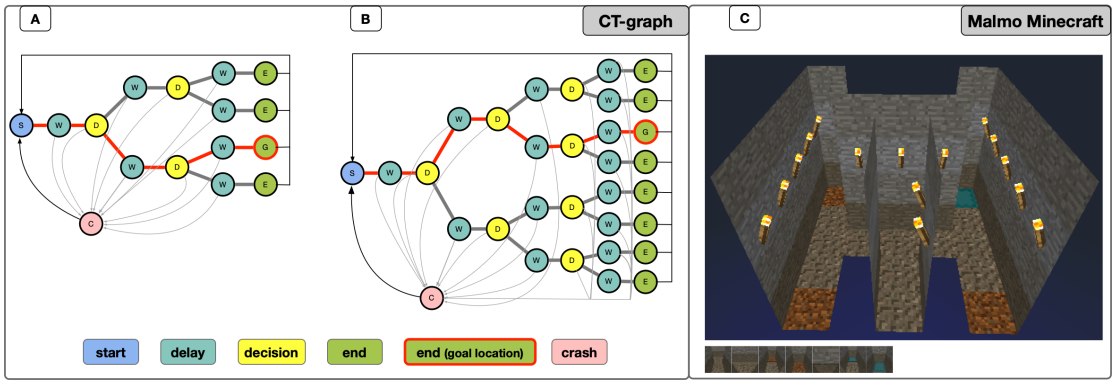


Figure 3.2: Environments (note, during execution, goal location is dynamic across episodes). (A) CT-graph instance, $b = 2$ and $d = 2$. (B) CT-graph instance, $b = 2$ and $d = 3$. (C) Malmo Minecraft instance (a double T-Maze), bird's eye view on top, with some sample observations at the bottom. The maze-end with the teal colour is the goal location.

in the environment, terminating at a terminal state. A trial consist of two or more tasks sampled from $\mathcal{T}$. The total number of episodes in a trial is kept fixed. A trial starts with an initial task $\mathcal{T}_i$ that runs for a number of episodes, and then the task is changed to other tasks (one after another) at different points within the trial (see Figure 3.1). The points at which a task change occurs are stochastically generated, and the task is changed before the start of the next episode. For example, when the number of tasks is set as 2 (i.e. $\mathcal{T}_i$ and $\mathcal{T}_j$), the trial starts with task $\mathcal{T}_i$ which runs for a number of episodes, and it is replaced by task $\mathcal{T}_j$ for the remaining episodes in the trial. An agent is iteratively trained, with each iteration consisting of a fixed number of trials. The subsections below describes two environments where the proposed system is evaluated.

3.3.1 The Configurable Tree Graph Environment

The configurable tree graph (CT-graph) environment is a graph abstraction of a decision making process. The complexity of the environment is specified via configuration parameters; branching factor b and depth d, controlling the width

and height of the graph. Additionally, it can be configured to be fully or partially observable. It contains the following types of state; start, wait, decision, end (leaf node of graph) and crash. Each observation $o \in \mathcal{O}$ is a 12×12 grey-scale image. The total number of end states grows exponentially as the depth d of the graph increases (see Figure 3.2A and B).

In the experiments in this study, partial observability is configured by mapping all wait states to the same observation, and all decision states to the same observation. Therefore, an agent in such partially observable environments would not be able to infer the state for its current position using the current observation alone. Also, b is set to 2. Therefore, each decision state has two choices, splitting into two sub-graphs. The discrete action space is defined as; *choice 1, choice 2, wait action*, thus discrete. The *wait action* is the correct action in a wait state. In a decision state, *choice 1* or *choice 2* is the correct subset from which to select. All incorrect actions lead to the crash state and episode termination.

An agent starts an episode in the start state, and the episode is completed when the agent traverses the graph to an end state or takes a wrong action in a state. Once an agent transitions from one state to the next, it cannot go back. In a task instance, one of the end states is set as the goal location. An agent receives a positive reward when it traverses to the goal location, and reward of 0 at other non-goal states. The agent may receive a negative reward in a crash state.

3.3.2 Malmo Minecraft Environment

Malmo [119] is an AI research platform built on top of Minecraft video game. The platform is configurable, and it enables the construction of various worlds in which AI agents can be evaluated. In this work, a double T-maze was constructed, with discrete action space *left turn, right turn and forward action.* A task is defined based on the maze ends, requiring the agent to navigate to a specific maze end (goal location). The maze end that is set as the goal location varies across tasks. The agent only receives a positive reward when they navigate to the maze end that is the goal location. It receives reward of 0 in every other time step. If the agent runs into a wall, the episode is terminated and it receives a negative reward. The agent receives a visual observation of its current view at each time step (hence it does not fully observe the entire environment). Each observation is a $32x32$ RGB image based on a first-person view of the agent at each time step.

3.4 Methods

We seek to develop an agent that is capable of continual adaptation through its life time (across episodes) - exploring, exploiting, re-exploring when the task changes and exploiting again. The system (specifically the controller or decision maker) is

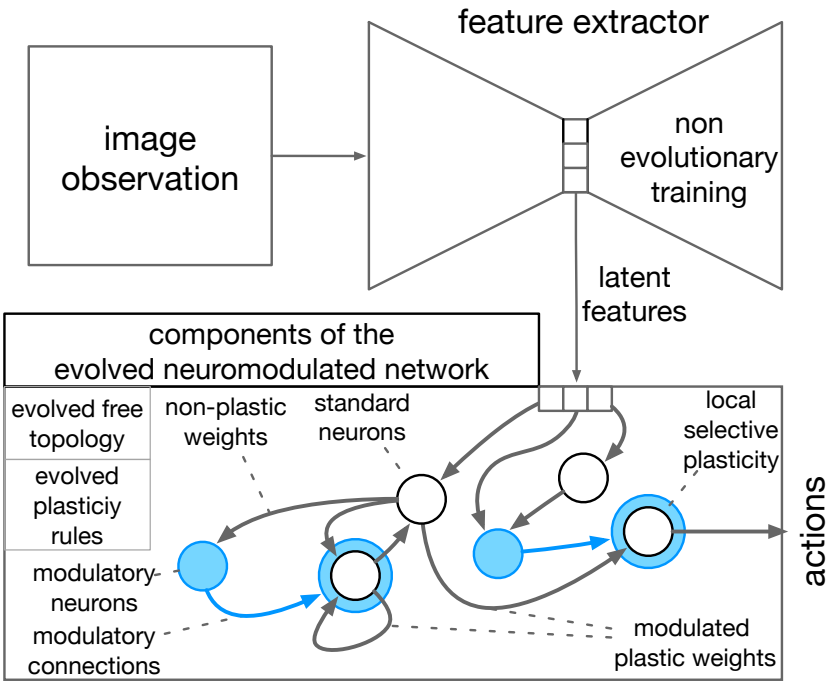


Figure 3.3: System overview, showcasing the feature extractor and controller components. In the controller, white and blue nodes are standard and modulatory neurons respectively. Modulatory connections facilitates selective plasticity in the network.

evolved to acquire knowledge about both the invariant and variant aspects of an environment (e.g. changing tasks).

The agent is modelled using two neural components with separate parameters and objectives; a deep network F_θ (used as a feature extractor and parameterized by θ) and a neuromodulated network G_ϕ (serving as a controller and parameterized by ϕ). Both components make up the overall system model $\mathcal{M}_{\theta,\phi}$. See Figure 3.3 for a general system overview. The presented architectural style is similar to a standard deep RL setup. However, it differs on two fronts; (i) the controller is a neuromodulated network (described in Section 3.4.2) rather than a standard neural network, (ii) the training setup combines gradient based optimization method [284, 223]), gradient free optimization method (neuroevolution [295, 258]), and Hebbian-based synaptic plasticity to train the system. Using this setup, each neural component therefore contains its own objective function. An autoencoder network was employed as the feature extractor, thus enabling the use of Mean Squared Error (MSE) or Binary Cross Entropy (BCE) objective function:

$$\arg\min_{\theta} \frac{1}{n} \sum_{i=1}^{n} (F_\theta(o_i) - o_i)^2$$

$$\arg\min_{\theta} -\frac{1}{n} \sum_{i=1}^{n} o_i \cdot \log(F_\theta(o_i)) + (1 - o_i) \cdot \log\left(1 - F_\theta(o_i)\right)$$

where n is the number of training observations and $F_\theta(o_i)$ is the output of the autoencoder for observation i (reconstructed observation). Each agent in the population uses the same feature extractor. The fitness function of the evolutionary algorithm is given by:

$$\arg\max_{\phi} \sum_{\mathcal{T}_i \sim \mathcal{T}} \sum_{ep=1}^{z} R(\tau_{ep})$$

$\mathcal{T}_i$ represents a task sampled from the task distribution $\mathcal{T}$, and a single trial consist of two tasks as defined in Section 3.3. Also, z is the number of episodes in which a task is kept fixed within a trial. It is stochastically generated and may differ between tasks in a trial within an interval. $R(\tau_{ep})$ is the accumulated reward of a trajectory of an episode ep, defined as:

$$R(\tau_{ep}) = \sum_{t=0}^{k} \mathcal{R}(s_t, G_\phi(F_\theta^{enc}(o_t))) \tag{3.1}$$

where $\mathcal{R}(s, a)$ is the reward function that takes state and action as arguments and produces a scalar reward value. F_θ^{enc} is the same autoencoder feature extractor network earlier described, but denoting that we only want the output from the encoder (the latent features). Also, t represents discrete time steps and k is the length of the trajectory of an episode.

3.4.1 Feature Extractor

This neural component of the system is tasked with learning a good latent representation of the observations from the environment, which can be fed to the controller as input. In the CT-graph experiments, a fully connected autoencoder was employed (two layers encoder and decoder respectively). In the Malmo Minecraft experiments, a convolutional autoencoder was employed (four layers encoder and decoder respectively).

3.4.2 Control Network (Decision Maker)

This neural component takes the latent features of the feature extractor as its input, and produces an output which serves as the final output of the system (the action or behaviour of the system). It is a neuromodulated network (described in the *Neuromodulated Network Dynamics* section below), that reproduces the model introduced in [251]. The network can evolve two neuron types - a standard and a modulatory neuron. The output neuron(s) always belong to the standard neuron type.

The control network is parameterized by ϕ. Unlike θ (which represents only the weights of the feature extractor network), ϕ consists of the weights, architecture and the co-efficients of Hebbian-based plasticity rule (described in the *Neur-*

omodulated Hebbian Plasticity section below) of the network, and it is evolved. Therefore, evolution is tasked with finding the architecture and plasticity rules, including selective plasticity enabled by modulatory neurons to target neurons. The large search space that is granted to evolution allows for rich dynamics that include memory in the form of both recurrent connections and temporary values of rapidly changing modulated weights.

The agent is never fed the reward signal explicitly. The reward signal is given only to the evolutionary process for the fitness evaluation (during an evolutionary step), which in turn drives the selection process. Therefore, the network is tasked to learn the discovery of reward cues implicitly from the visual observations in the environment.

Neuromodulated Network Dynamics

Though processing is distributed across neurons, a standard neural network usually contains one type of neuron - where the dynamics of each neuron is homogeneous across the network. In a neuromodulated network, there can be two types of neurons, each type having different dynamics - thus heterogeneous. The two types of neurons are standard neurons and modulatory neurons [251]. The standard neurons have the same dynamics as the ones in standard neural network. The modulatory neurons are used to dynamically regulate plasticity in the network.

Each neuron i has one standard and one modulatory activation value that represent the weighted amount of standard and modulatory activity they receive from other neurons (see Equations 3.2 and 3.3). $a_{std,i}$ is the output signal of neuron i that is propagated to other neurons in its outgoing connections (this is true for both standard and modulatory neurons). $a_{mod,i}$ is used internally by the neuron itself to regulate the Hebbian-based plasticity of the incoming connections from other standard neurons, as described in the next section below. The framework allows for selective plasticity in the network, as parts of the network may become plastic or not plastic depending on the change of the modulatory activation signals over time. In turn, the final action of the network is affected in the current and future time steps - thus enabling adaptation.

$$a_{\text{std},i} = \tanh \frac{\sum_{j \in \text{std}} w_{ji} a_{\text{std},j}}{2} \tag{3.2}$$

$$a_{\text{mod},i} = \tanh \frac{\sum_{j \in \text{mod}} w_{ji} a_{\text{std},j}}{2} \tag{3.3}$$

Neuromodulated Hebbian Plasticity

The Hebbian synaptic plasticity of the control network (which occurs per environment state-transition) is governed by the Equations 3.4, 3.5 and 3.6. A, B, C, D, α

are the coefficients of the plasticity rule. The update of a weight is dependent pre-synaptic and post-synaptic standard activations, the plasticity co-efficients, and the post-synaptic modulatory activation. This is true for all weights in the neuromodulated network.

$$w_{ij} = w_{ij} + \Delta w_{ij} \tag{3.4}$$

$$\Delta w_{ij} = a_{\text{mod},j} \cdot \delta w_{ij} \tag{3.5}$$

$$\delta w_{ij} = \alpha \cdot (A \cdot a_{\text{std},i} \cdot a_{\text{std},j} + B \cdot a_{\text{std},i} + C \cdot a_{\text{std},j} + D) \tag{3.6}$$

3.5 Results and Analysis

Figures 3.4 and 3.6 show the results of the experiments in the CT-graph environment. Figure 3.9 shows the results of the experiment in the Malmo Minecraft environment. In addition, we present results obtained in the Malmo Minecraft environment (Figure 3.9), evaluating the general applicability of PENN-A.

3.5.1 Performance in CT-graph Environments

The proposed method (PENN-A) was evaluated on depth 2 and 3 CT-graph environments, with branching factor of 2. The controller was evolved for 200 generations, with population of 600 and 800 for depth 2 and 3 experiments respectively. Tournament selection with segment size of 5 was employed. For each generation, each controller was evaluated for 4 trials, with 100 episodes and 2 tasks per trial. The initial task is changed between episodes 35 and 65, determined stochastically for each trial. The Hebbian weights of the control network were reset at the end of each trial. The depth 2 CT-graph experiment was employed as a baseline, and we compared PENN-A against some recent deep meta-RL methods (each with its own experimental setup). The depth 3 CT-graph experiment was employed to evaluate the PENN-A in a more complex configuration of the environment.

In order to ensure compatibility in the result presented across all methods, the number of evaluations (horizontal axis) were scaled to the approximate number of episodes equivalent. Additionally, the vertical axis is the average accumulated reward across all trials and episodes. In the depth 2 CT-graph result (Figure 3.4), we see that PENN-A performs optimally when compared to deep meta-RL methods; optimization-based (MAML [69] and CAVIA [314]) and memory-based (RL^2 [60] without extra input). Only the observations were fed as input to the neural network for all methods including PENN-A. We hypothesize the deep meta-RL methods perform sub-optimally due to the partial observability of the environment. When extra input (the reward, previous time step action and done state) are concatenated to the observation and fed to the RL^2 method (which is vanilla setup), then it is able to perform optimally (see Figure 3.5). We

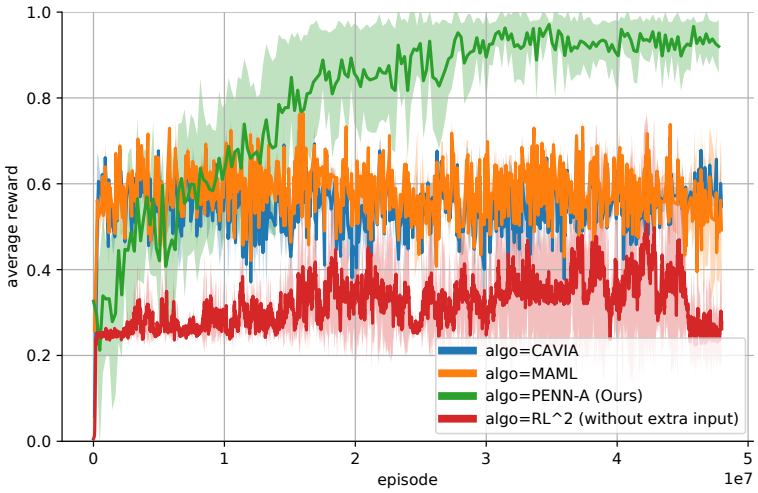


Figure 3.4: Results for a CT-graph with depth 2. PENN-A is compared against non-evolutionary meta-RL methods.

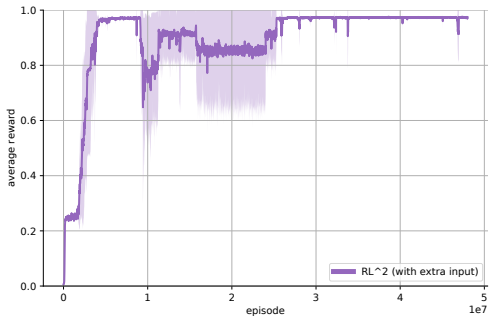


Figure 3.5: RL^2 in the CT-graph with depth 2. The method is run with extra input to the network (reward, done state, and previous time step action concatenated with current observation to form input).

hypothesize that RL^2 exploits the actions fed as input to the network, ignoring the observations and other parts of the input. This reduces the problem complexity in comparison to conditions where only the observations are fed as input.

Figure 3.6 presents result for a depth 3 CT-graph. We present result for only PENN-A in depth 3 CT-graph (a more difficult problem than depth 2 CT-graph) since the other methods performed sub-optimally in depth 2 CT-graph. We again observe PENN-A performing optimally in the more difficult CT-graph setting.

Network Analysis

To better understand the evolved solution and how the network implements policies, we analyzed the best performing networks after evolution in a depth 2 CT-graph environment. While different evolutionary runs produced highly different networks, we observed interesting patterns in the neural activations. For one network of 11 neurons (consisting of standard and modulatory neurons, including the output neuron), the absolute activation value distribution per episode time step

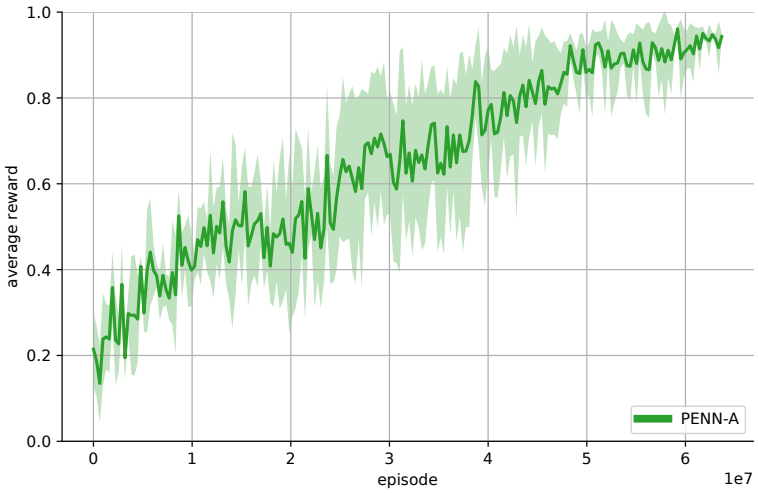


Figure 3.6: PENN-A performance in CT-graph instance of depth 3.

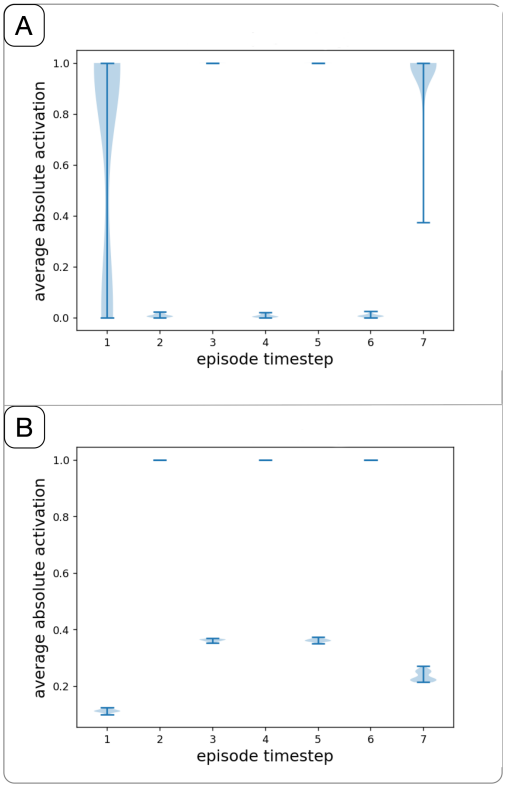


Figure 3.7: Absolute activation values distribution (across trials and episodes) per time step of a sample evolved controller. (A) This neuron is active specifically at decision states (steps 3 and 5), while it remains low at wait states. (B) This neuron clearly identifies wait states (steps 2, 4 and 6) and remains inactive otherwise.

(across all episodes) is plotted for each neuron in Figure 3.7. We see that the absolute activation distribution of some neurons are high at specific time steps, i.e., at specific points within the graph environment (see Figure 3.7A and B) — and therefore function as *location neurons*. Such kind of location neurons had been previously discovered in an evolutionary setting in [70]. In the current ex-

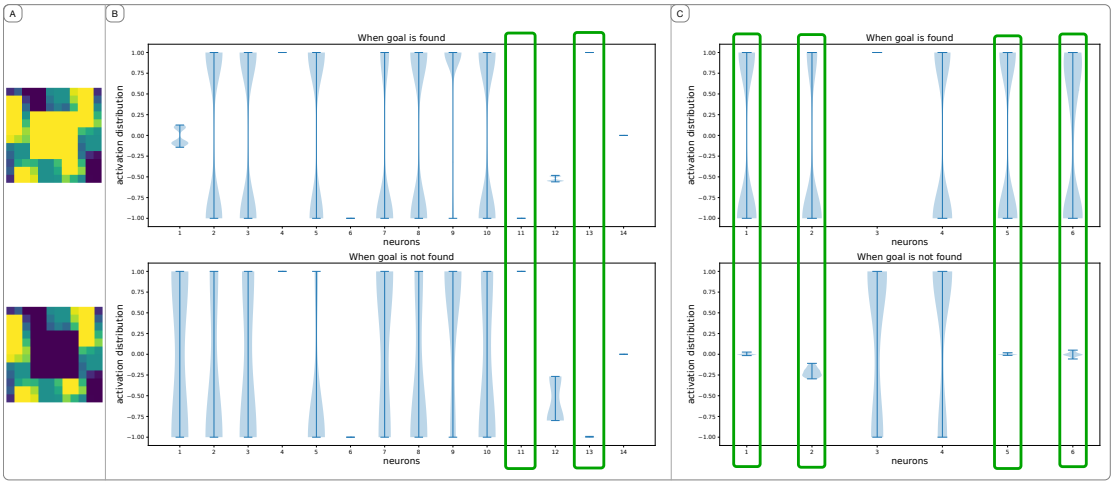


Figure 3.8: Distributions of the activation values for each neuron in a sample network when the goal location (reward) is found and vice versa. The neurons highlighted in green bounding box react differently to the presence or absence of reward cues from observation. (A) heat maps of grey-scale CT-graph observations. The top image is the observation presented when the goal location is found, with a bright square reward cue. The bottom image is the observation when the goal location is not found, with the reward cue absent. (B) Neurons 11 and 13 show complementary firing patterns based on reward cues. (C) Neurons 1, 2, 5 and 6 are active when a reward cue is observed, and have little or no activity when the reward cue is not observed.

periments, it is worth noting that location neurons are designed by evolution to exploit latent features and possibly help action-selection in a high-dimensional dynamic POMDP. In particular, the neuron in Figure 3.7A is active at decision states, while the neuron in Figure 3.7B is active at wait states.

One aspect of our experimental setting is that the reward signal is not fed to the network, but the environment provides reward cues embedded in the observations as it is shown in Figure 3.8A where a bright square represents a reward. The actual reward value is only accumulated in the fitness function, and is therefore not explicitly visible to the network. The surprising results that networks evolved to explore the environment and find the reward even if no reward signal was given suggests that the reward cue was recognised. In fact, in the example shown in Figure 3.8B, some neurons fire positively when a reward cue is observed and negatively when not observed or vice versa. Other neurons fire when a reward cue is observed and have little or no firing when not observed (see Figure 3.8C). Not all evolved networks appeared to have *reward neurons*. Nevertheless, the examples that evolved such reward cues detectors demonstrate that evolution is able to incorporate invariant knowledge of the environment to optimize the policy, in this case, reward seeking behaviour and fast adaptation speed to changing task. The deviation from the normal firing pattern of a reward neuron, due to task change, acts as an internal mechanism for the network to detect the task change. Thus,

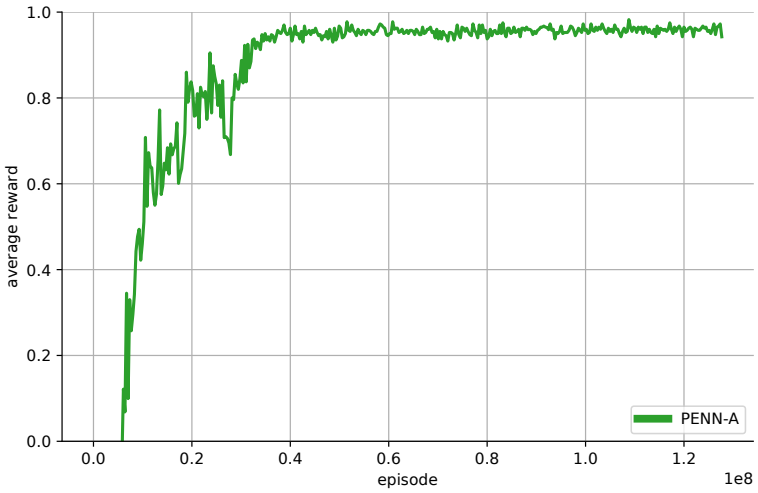


Figure 3.9: PENN-A performance in Malmo Minecraft.

the network rapidly reconfigures weights (via local synaptic plasticity) and adapt to the new task.

3.5.2 Performance in Malmo Minecraft

To further assess the validity of our method, it is important to use a different benchmark environment with a larger input and RGB observations that offered a different feature space, hence the Malmo Minecraft environment. The controller was evolved with population size of 800, in 400 generations. The same selection strategy as used in the CT-graph was employed. Each controller was evaluated for 8 trials, with 50 episodes and 3 tasks per trial. The task is changed at two stochastically generated points within the trial. The result is presented in Figure 3.9, keeping the same axes format as with the results presented for the CT-graph environment. Again, the proposed method was able to perform optimally with a high average reward score, demonstrating its capability to scale to other high dimensional, less abstract environments.

3.6 Conclusion

This chapter introduced an evolutionary design method for fast adaptation in POMDP environments. The system combines a feature extractor network and an evolved neuromodulated network with the aim of acquiring specific inborn knowledge and structure via evolution. While the suitability of evolved neuromodulated networks to solve environments with changing task was known [251, 255], we demonstrated that such advantages are scalable to high dimensional input spaces, and can be used in combination with an autoenconder. The results showed performance that compare or surpass some deep meta-RL algorithms. Interestingly, the evolved networks were capable of learning to recognise implicit reward cues,

and therefore could explore the environment in search for the goal location without an explicit reward signal. This ability that was acquired by the networks through evolution is an example of inborn knowledge that allow networks to be born with the knowledge of what are reward cues. Subsequently, this information can be used to direct fast adaptation when the optimal policy changes (e.g. the task change). The networks also evolved location neurons to help the deployment of a policy by distinguishing different states in the underlying MDP. We speculate that this approach might be promising when a combination of inborn knowledge and online learning are required to perform optimally in rapidly changing environments.

Acknowledgement

This research contribution is based upon work supported by the United States Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-18-C-0103. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA).

3.7 Supplementary Materials

The PENN-A source code containing the experimental setup is made available at: https://github.com/dlpbc/penn-a.

3.7.1 Feature Extractor

Mean Squared Error (MSE) was employed as the loss function across all CT-graph experiments, with a vanilla SGD (learning rate of 0.001) optimizer. Likewise, Binary Cross Entropy (BCE) was employed as the loss function in the Malmo Minecraft experiments, using RMSprop (learning rate of 0.0005) optimizer. The network architectures for the CT-graph and Malmo Minecraft experiments are presented in Table 3.1 and 3.2. The double horizontal line in both tables highlight the split between the encoder and decoder (i.e. specifications on top of the double line are for the encoder and likewise bottom for the decoder) of each autoencoder. In the CT-graph experiments where a Fully Connected (FC) autoencoder was employed, each input observation is flattened into a vector before feeding it to the network.

Layer	Activation	Units
Input	N/A	144
FC	ReLU	64
FC	ReLU	16
FC	ReLU	64
FC	ReLU	144

Table 3.1: Network architecture for CT-graph experiments

Layer	Activation	Kernel	Stride	Channels
Input	N/A	N/A	N/A	N/A
Conv2D	ReLU	3x3	2	16
Conv2D	ReLU	3x3	2	32
Conv2D	ReLU	3x3	2	32
Conv2D	ReLU	3x3	2	8
ConvTranspose2D	ReLU	3x3	2	32
ConvTranspose2D	ReLU	3x3	2	32
ConvTranspose2D	ReLU	3x3	2	16
ConvTranspose2D	Sigmoid	4x4	2	3

Table 3.2: Network architecture for Malmo Minecraft experiments

3.7.2 Control Network

Excluding population size and number of generations, the evolutionary parameters from [251] were followed.

The latent features from the feature extractor network were scaled between 0 and 1. To further restrict the latent features, a transformation operation was applied to the scaled latent features v before it was fed to the control network as shown below.

$$w = \begin{cases} 1 & if \sigma'(v) > 1 \\ 0 & if \sigma'(v) < 0 \\ \sigma'(v) & otherwise \end{cases}$$

$$\sigma'(v) = \log \frac{v}{1-v}$$

where $\sigma'(v)$ is an inverse sigmoid operation on v, and w is the transformed feature space. The scaling and transformation operations were performed independently of the feature extractor optimization (i.e. the operations were applied on copies of the latent features), and were applied across all experiments.

In this work, both evaluation environments were designed to work with discrete action space (3 actions each). Therefore, for each agent, a single output neuron (in the agent's pool of neurons) was employed across all experiments. The tanh activation value of the neuron was discretized to produce the actions of an agent. An activation value within the interval [-1.0, -0.33) mapped to one action, the interval [-0.33, +0.33] mapped to another action, and the interval (+0.33, 1.0] mapped to the last action.

Chapter 4

Context Meta-Reinforcement Learning via Neuromodulation

This chapter introduces and describes the second novel contribution in this thesis, which focused on the application of neuromodulation in meta-reinforcement learning (meta-RL) domain. Meta-RL algorithms enable agents to adapt quickly to tasks from few samples in dynamic environments. Such a feat is achieved through dynamic representations in an agent's policy network (obtained via reasoning about task context, model parameter updates, or both). However, obtaining rich dynamic representations for fast adaptation beyond simple benchmark problems is challenging due to the burden placed on the policy network to accommodate different policies. This contribution addresses the challenge by introducing neuromodulation as a modular component to augment a standard policy network that regulates neuronal activities in order to produce efficient dynamic representations for task adaptation. The proposed extension to the policy network was evaluated across multiple discrete and continuous control environments of increasing complexity. To prove the generality and benefits of the extension in meta-RL, the neuromodulated network was applied to two state-of-the-art meta-RL algorithms (CAVIA and PEARL). The result demonstrates that meta-RL augmented with neuromodulation produces significantly better result and richer dynamic representations in comparison to the baselines.

4.1 Introduction

Human intelligence, though specialized in some sense, is able to generally adapt to new tasks and solve problems from limited experience or few interactions. The field of meta-reinforcement learning (meta-RL) seeks to replicate such a flexible intelligence by designing agents that are capable of rapidly adapting to tasks from few interactions in an environment. The recent progress in the field such as [209, 69, 60, 282, 314, 87] have showcased start-of-the-art results. Studies with agents endowed with such adaptation capabilities are a promising venue for

developing much desired and needed artificial intelligence systems and robots with lifelong learning dynamics.

When an agent's policy for a meta-RL problem is encoded by a neural network, neural representations are adjusted from a base pre-trained point to a configuration that is optimal to solve a specific task. Such dynamic representations are a key feature to enable an agent to rapidly adapt to different tasks. These representations can be derived from gradient-based approaches [69], context-based approaches such as memory [185, 282, 60] and probabilistic [209], or hybrid approaches (i.e., combination of gradient and context methods) [314]. The hybrid approach obtains a task context via gradient updates and thus dynamically alters the representations of the network. Context approaches such as CAVIA [314] and PEARL [209] are more interpretable as they disentangle task context from the policy network, thus the task context is used to achieve optimal policies for different tasks.

One limitation of such approaches is that they do not scale well as the problem complexity increases because of the demand to store many diverse policies to be reached within a single network. In particular, it is possible that, as tasks grow in complexity, the tasks similarities reduce and thus the network's representations required to solve each task optimally becomes dissimilar. We hypothesize that standard policy networks are not likely to produce diverse policies from a trained base representation because all neurons have a homogeneous role or function: thus, significant changes in the policy require widespread changes across the network. From this observation, we speculate that a network endowed with modulatory neurons (neuromodulators) has a significantly higher ability to modify its policy.

Our approach to overcome this limiting design factor in current meta-RL neural approaches is to introduce a neuromodulated policy network to increase its ability to encode rich and flexible dynamic representations. The rich representations are measured based on the dissimilarity of the representations across various tasks, and are useful when the optimal policy of an agent (input-to-action mapping) is less similar across tasks. When combined with the CAVIA and PEARL meta-learning frameworks, the proposed approach produced better dynamic representations for fast adaptation as the neuromodulators in each layer serve as a means of directly altering the representations of the layer in addition to the task context.

Several designs exist for neuromodulation [58], either to gate plasticity [251, 182], gate neural activations [23] or alter high level behaviour [291]. The proposed mechanism in this work focuses on just one simple principle: modulatory signals alter the representations in each layer by gating the weighted sum of input of the standard neural component.

The primary contribution of this work is a neuromodulated policy network for

meta-reinforcement learning for solving increasingly difficult problems. The modular approach of the design allows for the proposed layer to be used with other existing layers (such as standard fully connected layer, convolutional layer and so on) when stacking them to form a deep network. The experimental evidence in this work demonstrates that neuromodulation is beneficial to adapt network representations with more flexibility in comparison to standard networks. Experimental evaluations were conducted across high dimensional discrete and continuous control environments of increasing complexity using CAVIA and PEARL meta-RL algorithms. The results indicate that the neuromodulated networks show an increasing advantage as the problem complexity increases, while they perform comparably on simpler problems. The increased diversity of the representations from the neuromodulated policy network are examined and discussed. The open source implementation of the code can be found at: https://github.com/dlpbc/nm-metarl

4.2 Related Concepts

Meta-reinforcement learning. This work builds on the existing meta learning frameworks [27, 231, 270, 242] in the domain of reinforcement learning. Recent studies in meta-reinforcement learning (meta-RL) can be largely classified into optimization and context-based methods. Optimization methods [69, 155, 257, 221] seek to learn good initial parameters of a model that can be adapted with a few gradient steps to a specific task. In contrast, context-based methods seek to adapt a model to a specific task based on few-shot experiences aggregated into context variables. The context can be derived via probabilistic methods [209, 161], recurrent memory [60, 282], recursive networks [185] or the combination of probabilistic and memory [315, 115]. Hybrid methods [314, 87] combine optimization and context-based methods whereby task specific context parameters are obtained via gradient updates.

Neuromodulation. Neuromodulation in biological brains is a process whereby a neuron alters or regulates the properties of other neurons in the brain [175]. The altered properties can either be in the cellular activities or synaptic weights of the neurons. Well known biological neuromodulators include dopamine (DA), serotonin (5-HT), acetycholine (ACh), and noradrenaline (NA) [22, 11]. Such neuromodulators were described in [58] within the reinforcement learning computation framework, with dopamine loosely mapped to the reward signal error (like TD error), serotonin representing discount factor, acetycholine representing learning rate and noradrenaline representing randomness in a policy's action distribution. Several studies have drawn inspiration from neuromodulation and applied it to gradient-based RL [291, 182] and neuroevolutionary RL [252, 251, 277] for dynamic task settings. In broader machine learning, neuromodulation has been

applied to goal-driven perception [318], and also in continual learning setting [23] where it was combined with meta-learning to sequentially learn a number of classification tasks without catastrophic forgetting. The neuromodulators used in these studies have different designs or functions: plasticity gating [251, 182], activation gating [23], direct action modification in a policy [291].

4.3 Background

4.3.1 Problem Formulation

In a meta-RL setting, tasks are sampled from a task distribution $p(\mathcal{T})$. Each task $\mathcal{T}_i$ is a Markov Decision Process (MDP), which is a tuple M_i=$\{\mathcal{S}, \mathcal{A}, q, r, q_0\}$ consisting of a state space $\mathcal{S}$, an action space $\mathcal{A}$, a state transition distribution $q(s_{t+1}|s_t, a_t)$, a reward function $r(s_t, a_t, s_{t+1})$, and an initial state distribution $q_0(s_0)$. When presented with a task $\mathcal{T}_i$, an agent (with a policy π) is required to quickly adapt to the task from few interactions. Therefore, the goal of the agent for each task is to maximize the expected reward in the shortest time possible:

$$\mathcal{J}(\pi) = \mathbf{E}_{q_0,q,\pi}\left[\sum_{t=0}^{H-1} \gamma^t r(s_t, a_t, s_{t+1})\right], \tag{4.1}$$

where H is a finite horizon and $\gamma \in [0, 1]$ is the discount factor.

4.3.2 Context Adaptation via Meta-Learning (CAVIA)

The CAVIA meta-learning framework [314] is an extension of the *model-agnostic meta-learning algorithm (MAML)* [69] that is interpretable and less prone to meta-overfitting. The key idea in CAVIA is the introduction of context parameters in a policy network. Therefore, the policy $\pi_{\theta,\phi}$ contains the standard network parameters θ and the context parameters ϕ. During the adaptation phase for each task (the gradient updates in the inner loop), only the context parameters are updated, while the network parameters are updated during the outer loop gradient updates. When fine-tuned in the inner loop, the context parameters provide a mechanism for the meta-RL agent to learn embeddings that map to specific tasks. There are different ways to provide or condition the policy network with the context parameters. In [314], the parameters were concatenated to the input.

In the meta-RL framework, an agent is trained for a number of iterations. For each iteration, N tasks represented as $\mathbf{T}$ are sampled from the task distribution $\mathcal{T}$. For each task i, a batch of trajectories τ_i^{train} is obtained using the policy $\pi_{\theta,\phi}$ with the context parameters set to an initial condition ϕ_0. The obtained trajectories for task i are used to perform a one step inner loop gradient update of the context parameters to new values ϕ_i, shown in the equation below:

$$\phi_i = \phi_0 - \alpha \nabla_\phi \mathcal{J}_{\mathcal{T}_i}(\tau_i^{train}, \pi_{\theta,\phi_0}), \tag{4.2}$$

where $\mathcal{J}_{\mathcal{T}_i}(\tau_i, \pi_{\theta,\phi})$ is the objective function for task i. After the one step gradient update of the policy, another batch of trajectories τ_i^{test} is collected using the updated task specific policy π_{θ,ϕ_i}.

After completing the above procedure for all tasks sampled from $\mathcal{T}$, a meta gradient step (also referred to as the outer loop update) is performed, updating θ to maximize the average performance of the policy across the task batch.

$$\theta = \theta - \beta \nabla_\theta \frac{1}{N} \sum_{\tau_i \in \mathbf{T}} \mathcal{J}_{\mathcal{T}_i}(\tau_i^{test}, \pi_{\theta,\phi_i}). \tag{4.3}$$

As earlier mentioned, the context parameter is set to an initial condition ϕ_0 when learning a new task. The initial condition is set using one of two main strategies: either by meta-learning the initial parameter or set a fixed value. The authors in [314] argued that setting a fixed value (i.e. zero initialization employed) for the context initialization is sufficient. The algorithm for the training an agent using CAVIA is presented in Algorithm 1.

Algorithm 1 CAVIA Meta-Training (source: [314])

Require: Distribution over tasks $p(\mathcal{T})$
Require: Step sizes α and β
Require: Initial policy $\pi_{\phi_0,\theta}$ with θ initialised randomly and $\phi_0 = 0$
while not done **do**
  Sample batch of tasks $\mathbf{T} = \{\mathcal{T}_i\}_{i=1}^{N}$ where $\mathcal{T}_i \sim p$
  for all $\mathcal{T}_i \in \mathbf{T}$ **do**
    Collect rollout τ_i^{train} using $\pi_{\phi_0,\theta}$
    $\phi_i = \phi_0 + \alpha \nabla_\phi \tilde{\mathcal{J}}_{\mathcal{T}_i}(\tau_i^{\text{train}}, \pi_{\phi_0,\theta})$
    Collect rollout τ_i^{test} using $\pi_{\phi_i,\theta}$
  end for
  $\theta \leftarrow \theta + \beta \nabla_\theta \frac{1}{N} \sum_{\mathcal{T}_i \in \mathbf{T}} \tilde{\mathcal{J}}_{\mathcal{T}_i}(\tau_i^{\text{test}}, \pi_{\phi_i,\theta})$
end while

4.3.3 Probabilistic Embeddings for Actor-Critic Meta-RL (PEARL)

PEARL [209] is an off-policy meta-RL algorithm that is based on the soft actor-critic architecture [89]. The algorithm derives the context of the task to which an agent is exposed through probabilistic sampling. Given a task, the agent maintains a prior belief of the task, and as the agent interacts with the environment, it updates the posterior distribution with the goal of identifying the specific task context. The context variables $\mathbf{z}$ are concatenated to the input of the actor and critic neural components of the setup. To estimate this posterior $p(\mathbf{z}|\mathbf{c})$, an addi-

tional neural component called an inference network $q_\phi(\mathbf{z}|\mathbf{c})$ is trained using the trajectories $\mathbf{c}$ collected for tasks sampled from the task distribution $\mathcal{T}$. The objective function for the actor, critic and inference neural components are described below,

$$\mathcal{L}_{actor} = \mathbb{E}_{\substack{\mathbf{s}\sim\mathcal{B},\mathbf{a}\sim\pi_\theta \\ \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c})}} \left[D_{\mathrm{KL}} \left(\pi_\theta(\mathbf{a}|\mathbf{s},\bar{\mathbf{z}}) \middle\| \frac{\exp(Q_\theta(\mathbf{s},\mathbf{a},\bar{\mathbf{z}}))}{\mathcal{Z}_\theta(\mathbf{s})} \right) \right] \tag{4.4}$$

$$\mathcal{L}_{critic} = \mathbb{E}_{\substack{(\mathbf{s},\mathbf{a},r,\mathbf{s}')\sim\mathcal{B} \\ \mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c})}} [Q_\theta(\mathbf{s},\mathbf{a},\mathbf{z}) - (r + \bar{V}(\mathbf{s}',\bar{\mathbf{z}}))]^2 \tag{4.5}$$

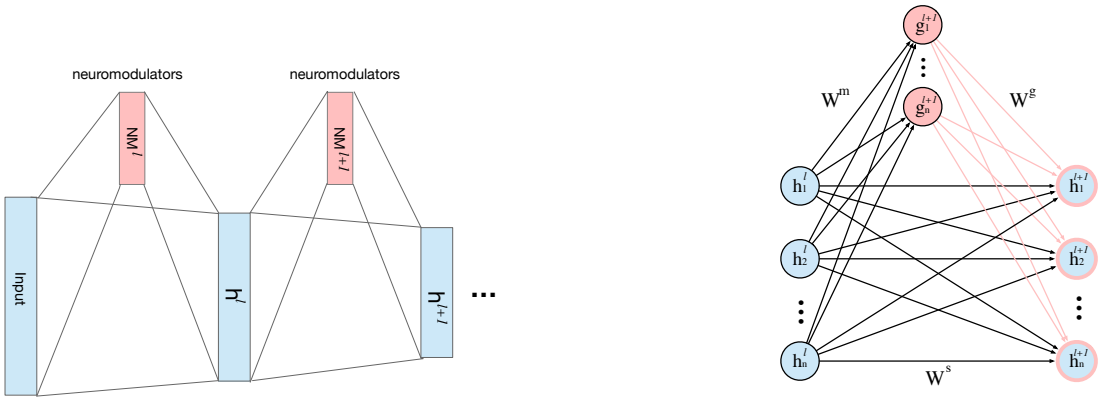
$$\mathcal{L}_{inference} = \mathbb{E}_{\mathcal{T}}[\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{c}^{\mathcal{T}})}[\mathcal{L}_{critic} + \beta D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{c}^{\mathcal{T}})||p(\mathbf{z}))]] \tag{4.6}$$

where $\bar{V}$ is a target network and $\bar{\mathbf{z}}$ means that gradients are not being computed through it, $p(\mathbf{z})$ is a unit Gaussian prior over Z, $\mathcal{B}$ is the replay buffer and β is a weighting hyper-parameter. The complete algorithm for meta-training in PEARL is reported in Algorithm 2.

Algorithm 2 PEARL Meta-training (source: [209])

Require: Batch of training tasks $\{\mathcal{T}_i\}_{i=1...T}$ from $p(\mathcal{T})$, learning rates $\alpha_1, \alpha_2, \alpha_3$

Initialize replay buffers $\mathcal{B}^i$ for each training task
while not done **do**
  for each $\mathcal{T}_i$ **do**
    Initialize context $\mathbf{c}^i = \{\}$
    for $k = 1, \ldots, K$ **do**
      Sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$
      Gather data from $\pi_\theta(\mathbf{a}|\mathbf{s},\mathbf{z})$ and add to $\mathcal{B}^i$
      Update $\mathbf{c}^i = \{(\mathbf{s}_j, \mathbf{a}_j, \mathbf{s}'_j, r_j)\}_{j:1...N} \sim \mathcal{B}^i$
    end for
  end for
  for step in training steps **do**
    for each $\mathcal{T}_i$ **do**
      Sample context $\mathbf{c}^i \sim \mathcal{S}_c(\mathcal{B}^i)$ and RL batch $b^i \sim \mathcal{B}^i$
      Sample $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{c}^i)$
      $\mathcal{L}^i_{actor} = \mathcal{L}_{actor}(b^i, \mathbf{z})$
      $\mathcal{L}^i_{critic} = \mathcal{L}_{critic}(b^i, \mathbf{z})$
      $\mathcal{L}^i_{KL} = \beta D_{\mathrm{KL}}(q(\mathbf{z}|\mathbf{c}^i)||r(\mathbf{z}))$
    end for
    $\phi \leftarrow \phi - \alpha_1 \nabla_\phi \sum_i (\mathcal{L}^i_{critic} + \mathcal{L}^i_{KL})$
    $\theta_\pi \leftarrow \theta_\pi - \alpha_2 \nabla_\theta \sum_i \mathcal{L}^i_{actor}$
    $\theta_Q \leftarrow \theta_Q - \alpha_3 \nabla_\theta \sum_i \mathcal{L}^i_{critic}$
  end for
end while



(a) Modular format of the proposed architecture

(b) A single neuromodulated fully connected layer

Figure 4.1: Overview of the proposed computational framework. (a) Light blue boxers indicate layers of standard neurons and pink boxes are layers of modulatory neurons. (b) Illustration of a single layer of the proposed architecture.

4.4 Neuromodulated Network

This section introduces the extension of the policy network with neuromodulation. A graphical representation of the network is shown in Figure 4.1a. The neuromodulated policy network is a stack of neuromodulated fully connected layers.

4.4.1 Computational Framework

A neuromodulated fully connected layer contains two neural components: standard neurons and neuromodulators (see Figure 4.1b). The standard neurons serve as the output of the layer (i.e., the layer's representations) and they are connected to the preceding layer via standard fully connected weights W^s. The neuromodulators serve as a means to alter the output of the standard neurons. They receive input via standard fully connected weights W^g from the preceding layer in order to generate their neural activity, which is then projected to the standard neurons via another set of fully connected weights W^m. The function of the projected neuromodulatory activity defines the representation altering mechanism. For example, it could gate the plasticity of W^s, gate neural activation of $\mathbf{h}$ or do something else based on the designer's specification. While different types of neuromodulators can be used [58], in this particular work, we employ an activity-gating neuromodulator. Such neuromodulator multiplies the activity of the target (standard) neurons before a non-linearity is applied to the layer. Formally, the structure can be described with three parameter matrices: W^s defines weights connecting the input to the standard neurons, W^g defines weights connecting the input to the neuromodulators and W^m defines weights connecting the neuromodulators to the standard neurons. The step-wise computation of a forward pass through the neuromodulatory structure is given below:

$$\mathbf{h}^s = W^s \cdot \mathbf{x} \tag{4.7}$$

$$\mathbf{g} = ReLU(W^g \cdot \mathbf{x}) \tag{4.8}$$

$$\mathbf{h}^m = tanh(W^m \cdot \mathbf{g}) \tag{4.9}$$

$$\mathbf{h} = ReLU(\mathbf{h}^s \otimes \mathbf{h}^m) \tag{4.10}$$

where $\mathbf{x}$ is the layer's input, $\mathbf{h}^s$ is the weighted sum of input of the standard neurons, $\mathbf{g}$ is activity of the neuromodulators derived from the weighted sum of input, $\mathbf{h}^m$ is the neuromodulatory activity projected onto the standard neurons, and $\mathbf{h}$ is the output of the layer. The key modulating process takes place in the element-wise multiplication of the $\mathbf{h}^s$ and $\mathbf{h}^m$.

The *tanh* non-linearity is employed to enable positive and negative neuromodulatory signals, and thus gives the network the ability to affect both the magnitude and the sign of target activation values. When *ReLU* is used as the non-linearity for the layer's output h, $\mathbf{h}^m$ has the intrinsic ability to dynamically turn on or off certain output in $\mathbf{h}$.

A simpler version of the proposed model can be achieved by only considering the sign, and not the magnitude, of the neuromodulatory signal, using the following variation of Equation 4.10:

$$\mathbf{h} = ReLU(\mathbf{h}^s \otimes \mathbf{sign}(\mathbf{h}^m)) \tag{4.11}$$

This variation is shown to be suited for discrete control problems.

4.5 Experimental Configurations

All experiments were conducted using machines containing Tesla K80 and GeForce RTX 2080 GPUs. Also note that across all experiments, the output layer in the neuromodulated policy network (in CAVIA and in PEARL) employed a regular fully connected linear layer while the preceding layers were neuromodulated fully connected layers.

4.5.1 CAVIA

Following the experimental setup of the original CAVIA paper [314], the context variables were concatenated to the input of the policy network and were reset to zero at the beginning of each task across all experiments. Also, during each training iteration, the policy was adapted using one gradient update in the inner loop as employed in [314, 69]. After training, the iteration with the best policy performance or the final policy at the end of training was used to conduct meta-

testing evaluations to produce the final result. During meta-testing, the policy was evaluated using a number of tasks sampled from the task distribution and it was adapted (fine-tuned) for each task using 4 inner loop gradient updates. All policy networks employed ReLU non-linearity across all experiments.

The CAVIA experimental configurations across all environments are presented in Table 4.1, with *2D Nav* denoting the 2D navigation benchmark, *Ch Dir and Ch Vel* denoting the Half-Cheetah direction and Half-Cheetah Velocity benchmarks, *ML 1 and ML45* denoting the meta-world ML1 and ML45 benchmarks, *CT d2, d3, d4* denoting the CT-graph depth2, 3 and 4 benchmarks respectively.

	2D Nav	**Ch Dir**	**Ch Vel**	**ML 1**	**ML 45**	**CT d2**	**CT d3**	**CT d4**
Number of iterations	500	500	500	500	500	500	700	1500
Number tasks per iteration (meta-batch size)	20	40	40	40	45	20	25	20
Number inner loop grad steps (for meta-training)	1	1	1	1	1	1	1	1
Number trajectories per task (for meta-training)	20	20	20	20	10	20	25	60
Number inner loop grad steps (for meta-testing)	4	4	4	4	4	4	4	4
Number trajectories per task (for meta-testing)	20	40	40	40	20	20	40	100
Policy network specification								
Number of context parameters	5	50	50	50	100	5	10	20
Number of hidden layers	2	2	2	2	2	2	2	2
Hidden layer size	100	200	200	200	200	200	300	600
Neuromodulator size (*for neuromodulated policy only*)	4	32	32	32	32	8	16	32

Table 4.1: CAVIA experimental configurations.

Across all experiments in CAVIA, Trust Region Policy Optimization (TRPO) [233] was employed as the outer loop update algorithm. Vanilla policy gradient

[285] with generalized advantage estimation (GAE) [234] was employed as the inner loop update algorithm with learning rate of 0.5 for the 2D navigation and the CT-graph experiments, and 10.0 for half-cheetah and meta-world experiments. Both the inner and outer loop training employed a linear feature baseline introduced in [59]. The hyperparameters for TRPO are presented in Table 4.2. Furthermore, finite-differences was employed to compute the Hessian-vector product for TRPO in order to avoid computing third-order derivatives as highlighted in [69]. During sampling of data for each task in environments, multiprocessing was employed using 4 workers.

Name	**Value**
maximum KL-divergence	1×10^{-2}
number of conjugate gradient iterations	10
conjugate gradient damping	1×10^{-5}
maximum number of line search iterations	15
backtrack ratio for line search iterations	0.8

Table 4.2: TRPO hyperparameters

4.5.2 PEARL

Similar to CAVIA, the original experimental configurations in PEARL were followed for the half-cheetah benchmarks. Also, most of the configurations of PEARL in the original meta-world experiments were followed.

Across all PEARL experiments in this work, the learning rate across all neural components (policy, Q, value and context networks) was set to 3e-4, with KL penalty (KL lambda) set to 0.1. Furthermore, for experiments that involved the use of neuromodulation, the neuromodulator was employed only in the policy (actor) neural component. Table 4.3 highlights some of the PEARL configurations across the evaluation environments.

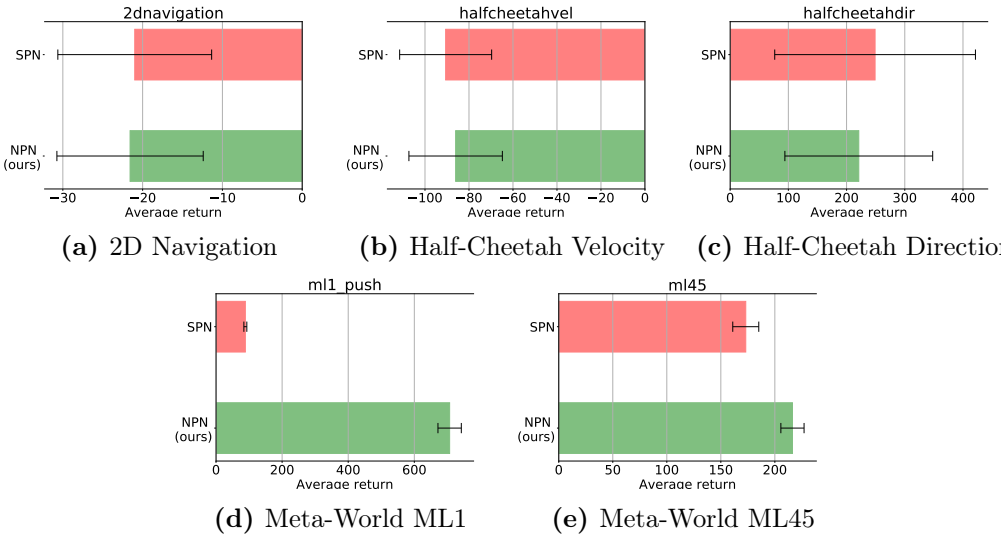
4.6 Results and Analysis

In this section, the results of the neuromodulated policy network evaluations across high dimensional discrete and continuous control environments with varying levels of complexity are presented. The continuous control environments are the simple 2D navigation, the half-cheetah direction [69] and velocity [69] Mujoco [272] based environments and the meta-world ML1 and ML45 environments [299]. The discrete action environment is a graph navigation environment that supports configurable levels of complexity called the CT-graph [254, 144, 26]. The experimental setup focused on investigating the beneficial effect of the proposed neuromodulatory mechanism when augmenting existing meta-RL frameworks (i.e., neuromodulation as complementary tool to meta-RL rather than competing). To this

	2D Nav	**Ch Dir**	**Ch Vel**	**ML 1**	**ML 45**
Number of iterations	500	500	500	1000	1000
Number of train task	40	2	100	50	225
Number of test task	40	2	30	50	25
Number of initial steps	1000	2000	2000	4000	4000
Number of steps prior	400	1000	400	750	750
Number of steps posterior	0	0	0	0	0
Number of extra posterior steps	600	1000	600	750	750
Reward scale	5	5	5	10	5
Policy network specification					
Context vector size	5	5	5	7	7
Network size (*policy, Q and value networks*)	300	300	300	300	300
Inference (context) network size	200	200	200	200	200
Number of hidden layers (*policy, Q, value and context networks*)	3	3	3	3	3
Neuromodulator size (*for neuromodulated policy only*)	4	32	32	32	32

Table 4.3: PEARL experimental configurations.

end, using CAVIA meta-RL method [314], a standard policy network (SPN) is compared against the neuromodulated policy network (NPN) across the aforementioned environments. Similarly, SPN is compared against NPN using PEARL [209] method only in the continuous control environments because the soft actor-critic architecture employed by PEARL is designed for continuous control. We present the analysis of the learned dynamic representations from a standard and a neuromodulated network in Section 4.6.2. Finally, the policy networks were evaluated in a RGB autonomous vehicle navigation domain in the CARLA driving simulator using CAVIA and the results and discussions are presented in Section 4.6.1.



(a) 2D Navigation **(b)** Half-Cheetah Velocity **(c)** Half-Cheetah Direction

(d) Meta-World ML1 **(e)** Meta-World ML45

Figure 4.2: Adaptation performance across tasks of the standard policy network (SPN) and the neuromodulated policy network (NPN) in continuous control environment using CAVIA meta-RL framework. Across three seed runs, the performance was measured based on average return from the rewards acquired during evaluation.

4.6.1 Performance

The experimental setup for CAVIA and PEARL as in [314] and [209] were followed. For PEARL, neuromodulation was applied only to the actor neural component. The performance reported are the meta-testing results of the agents in the evaluation environments after meta-training has been completed (Figures 4.2, 4.3, 4.4 and 4.6). During meta-testing in CAVIA, the policy networks were fine-tuned for 4 inner loop gradient steps. Lastly, depending on the evaluation environment, the metric used to judge evaluation performance was either return[1] or success rate[2].

2D Navigation Environment

The first simulations are in the 2D point navigation experiment introduced in [69]. An agent is tasked with navigating to a randomly sampled goal position from a start position. A goal position is sampled from the interval [-0.5, 0.5]. The reward function is the negative squared distance between the current agent position and the goal. An observation is the agent's current 2D position while the actions are velocity commands clipped at [-0.1, 0.1]. The result of the meta-testing performance evaluation comparing both the standard policy network and neuromodulated policy network is presented in Figure 4.2a for CAVIA and Figure 4.3a for PEARL. The result shows that both policy networks had a relative good

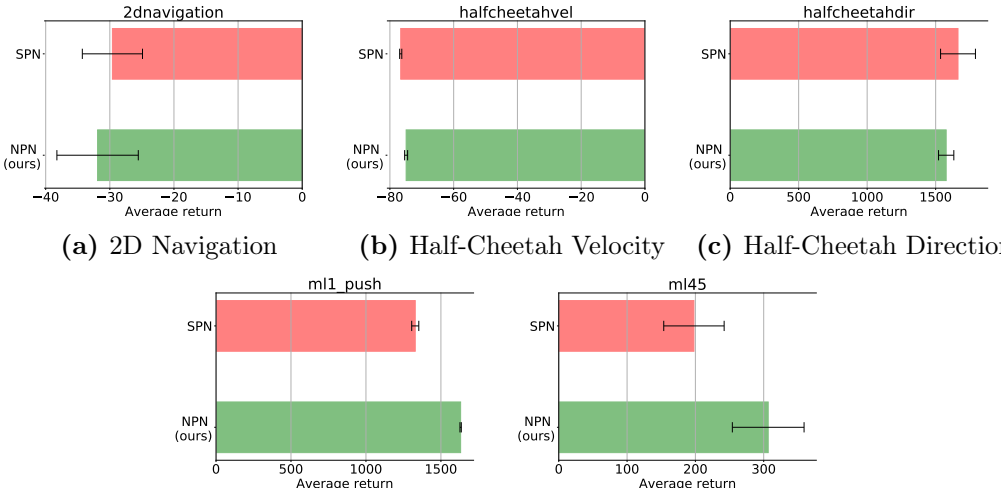
[1]return is a standard metric in RL that is computed as the sum of cumulative reward acquired by the agent.

[2]success rate is a metric introduced in Meta-World, having a value of 1 if the agent has solved or is close to solving the task (i.e., if the distance between the current position of the task relevant object and goal position is smaller than some ϵ value, otherwise, it is set to 0.

performance. Such optimal performance is expected from both policies as the environment is simple and the dynamic representations required for each task are not very distinct.

Half-Cheetah

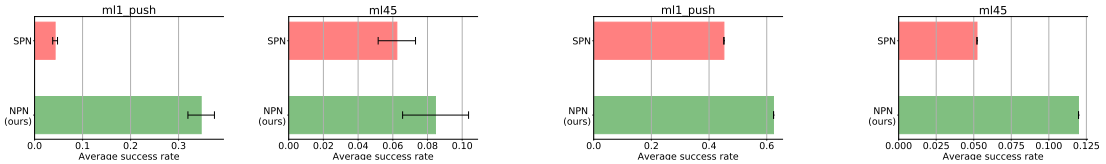
The half-cheetah is an environment based on the MuJoCo simulator [272] that requires an agent to learn continuous control locomotion. We employ two standard meta-RL benchmarks using the environment as proposed in [69]; (i) the direction task that requires the cheetah agent to run either forward or backward and (ii) the velocity task that requires the agent to run at a certain velocity sampled from a distribution of velocities. Although challenging (due to their high dimensional nature) in comparison to the 2D navigation task, these benchmark are still simplistic as the direction benchmark contains only two unique tasks and the velocity benchmark samples small range of velocities ($[0, 2.0)$ or $[0, 3.0)$). Therefore, the optimal policies across tasks in these benchmarks possess similar representations. The results of the experiments for both benchmarks are presented in Figures 4.2c and 4.2b for CAVIA, and Figures 4.3c and 4.3b for PEARL. Unsurprisingly, the results show comparable level of performance between the standard policy network and the neuromodulated policy network across CAVIA and PEARL. These benchmarks are of medium complexity and the optimal policy for each task is similar to others.



(a) 2D Navigation **(b)** Half-Cheetah Velocity **(c)** Half-Cheetah Direction

(d) Meta-World ML1 **(e)** Meta-World ML45

Figure 4.3: Adaptation performance across tasks of the standard policy network (SPN) and the neuromodulated policy network (NPN) in continuous control environment using PEARL meta-RL framework. Across three seed runs, the performance was measured based on average return from the rewards acquired during evaluation.



(a) CAVIA, ML1 **(b)** CAVIA, ML45 **(c)** PEARL, ML1 **(d)** PEARL, ML45

Figure 4.4: Adaptation performance (across tasks, based on success rate metric) of the standard policy network (SPN) and the neuromodulated policy network (NPN) in CAVIA and PEARL. Across three seed runs, the performance was measured based on the success rate metric from the evaluation.

Meta-World

The neuromodulated policy network was evaluated in a complex high-dimensional continuous control environment called meta-world [299]. In meta-world, an agent is required to manipulate a robotic arm to solve a wide range of tasks (e.g. pushing an object, pick and place objects, opening a door and more). Two instances of the benchmark ML1 and ML45 were employed. In ML1 instance, the robot is required to solve a single task that contains several parametric variations (e.g. push an object to different goal locations). The parametric variations of the selected task are used as the meta-train and meta-test tasks. ML45 is a more complex instance that contains a wide variety of tasks (each task with parametric variations). It consists of 45 distinct meta-train tasks and 5 distinct meta-test tasks. The standard policy network and neuromodulated policy network were evaluated in ML1 and ML45 instances using CAVIA and PEARL. The results[3] are presented in Figures 4.2d and 4.2e for CAVIA, and Figures 4.3d and 4.3e for PEARL. In these complex benchmarks, the results show that the neuromodulated policy network outperforms the standard policy network in both CAVIA and PEARL, highlighting the advantage neuromodulation offers in complex problem setting. In addition to judging the performance based on reward, results are also presented using the success rate metric (introduced in [299] as a metric judge whether or not an agent is able to solve a task) in Figure 4.4. The results again show that the neuromodulated policy network achieved significantly higher average success rate both in CAVIA and PEARL in comparison to the standard policy network.

Configurable Tree graph (CT-graph) Environment

The CT-graph is a sparse reward discrete control graph environment with increasing complexity that is specified via parameters such as branch b and depth d. Each environment instance in the CT-graph is composed of a start state, a crash state, a number of wait states, decision states, end states and a goal state (one

[3]The experiments were conducted using the updated Meta-World (i.e., v2) environment containing the updated reward function.

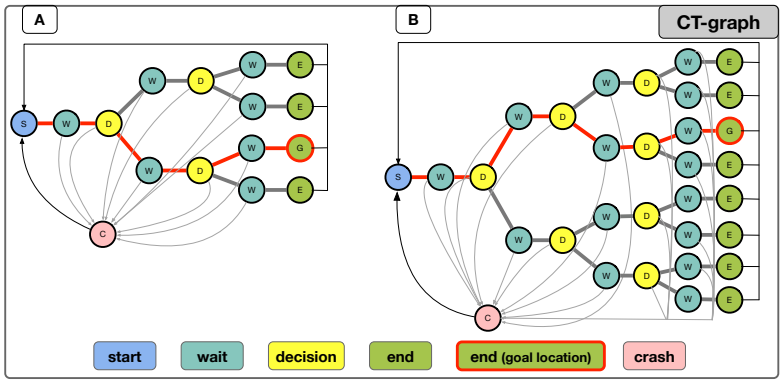


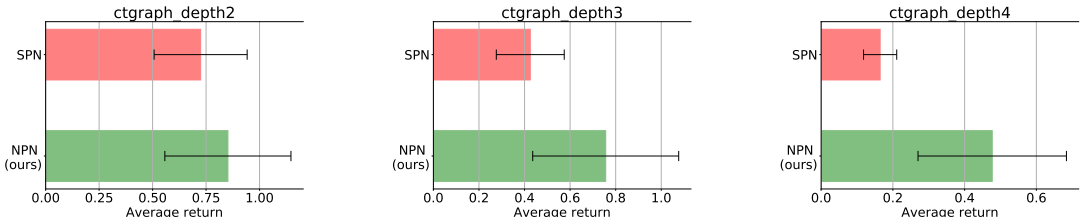
Figure 4.5: (A) CT-graph depth2 and (B) CT-graph depth3. The coloured legends represent the various state types in the environment.

of the end states designated as the goal). A wait state is found between decision states (the tree graph splits at decision states). The wait state requires an agent to take a wait (forward) action, a decision state requires an agent to take one of the decision (turn) actions. Any decision action at a wait state, or wait action at a decision state leads to a crash where the agent is punished with a negative reward of -0.01 and returns to the start. When an agent navigates to the correct end state (the goal location), it receives a positive reward of 1.0. Otherwise, the agent receives a reward of 0.0 at every time step. An episode is terminated either at a crash state or when the agent navigates to any end state. The observations are 1D vector (with full observability of each state) whose length depends on the environment instance configuration.

The environment's complexity is defined via a number of configuration parameters that is used to specify the graph size (using branch b and d), sequence length, reward function, and level of state observability. The three CT-graph instances used in this work were setup with varying depth parameter: with increasing depth, the sequence of actions grows linearly, but the search space for the policy network grows exponentially. The simplest instance has d set to 2 (CT-graph depth2), and the next has d set to 3 (CT-graph depth3) and the most complex instance has d set to 4 (CT-graph depth4). Figure 4.5 depicts a graphical view of CT-graph depth2 and 3. The meta-testing results are presented in Figure 4.6. The results show a significant difference in performance between standard and neuromodulated policy network. The optimal adaption performance from the neuromodulated policy network stems from the rich dynamic representations needed for adaptation as discussed in Section 4.6.2.

CARLA Environment

Additional experiments were conducted in an autonomous driving environment called CARLA [57] to provide evidence on whether the method scales to complex

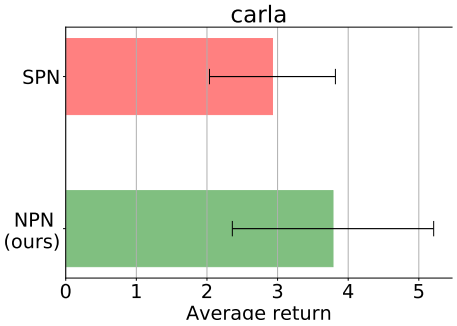


(a) CT-graph $(b = 2, d = 2)$ **(b)** CT-graph $(b = 2, d = 3)$ **(c)** CT-graph $(b = 2, d = 4)$

Figure 4.6: Adaptation performance across tasks of the standard policy network (SPN) and the neuromodulated policy network (NPN) in three discrete control environments using CAVIA meta-RL framework. Across three seed runs, the performance was measured based on the success rate metric from the evaluation.

RGB input distributions such as those in autonomous driving, providing additional validation on the robustness of the proposed approach. CARLA (see Figure 4.7a), is an open source experimentation platform for autonomous driving research. It contains a host of configuration parameters that is used to specify an environment instance (for example, weather). MACAD [202], a wrapper on top of CARLA with OpenAI gym interface, was employed to run the experiments. In this work, the environment was configured to use RGB observations (images of size 64x64x3), 9 discrete actions (coast, turn left, turn right, forward, brake, forward left, forward right, brake left, and brake right), and a clear (sunny) noon weather.

(a) A snapshot of the CARLA environment. *Note, the view of environment in this figure is for the purpose of presentation.* The actual environment instance used in this work is configured as a first person view with a smaller observation (image) size.



(b) Adaptation performance of standard policy network (SPN) and neuromodulated policy network (NPN) in the CARLA environment.

Figure 4.7: CARLA environment and results

The agent (vehicle) is presented with a goal of navigating from a start position to an end position. The start and end points are randomly set from a pre-defined list of co-ordinates. We setup two distinct tasks in the environment - drive aggressively and drive passively - defined by reward functions, that can be sampled from a uniform task distribution. Although the tasks are quite similar, they are challenging due to the domain of the problem (learning to drive) and the RGB

pixel observations from the environment. Therefore, it is a suitable environment to further scale up meta-RL algorithms.

Each experiment processes the environment's observations through a variational autoencoder (VAE) [131, 215] that was pre-trained using samples collected from taking random actions in the environment. Using CAVIA, the latent features from the VAE were concatenated with the context parameters and then passed as input to the policy network. Only the policy network was updated during the meta-training and testing, while the VAE was kept fixed.

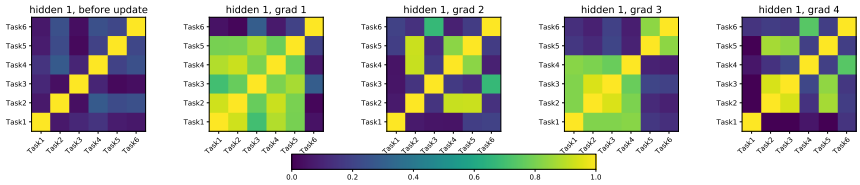
Due to the computational load of the environment, both the standard and the neuromodulated policy network were evaluated for 300 iterations, with 4 sampled tasks per iteration and context parameter size of 10. For each task, 2 episodes are collected before and after one step of inner loop gradient update. The results are presented in Figure 4.7b, with the neuromodulated policy network showing an advantage over the standard policy network. In general, the results show promise towards scaling meta-RL algorithms to even more challenging problem domains.

4.6.2 Analysis

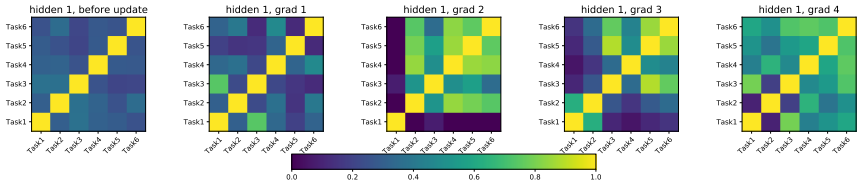
In this section, we conduct analysis on the learnt representations of the standard and neuromodulated policy networks for tasks in the 2D Navigation and CT-graph environments. The policy networks trained using CAVIA was chosen for the analysis as the single neural component in CAVIA (i.e. the policy network) makes it easier to analyse in comparison to PEARL which contain multiple neural components. Furthermore, PEARL experiments were conducted only in continuous control environments (similar to the original paper), whereas CAVIA experiments covered both discrete and continuous control environments. Hence, analysis in CAVIA allowed for more coverage across benchmarks.

To measure representation similarity across task, we employ the use of the centered kernel alignment (CKA) [137] similarity index, comparing per layer representations of both standard and neuromodulated policy networks across different tasks. There exist several similarity index measures such as canonical correlation analysis (CCA) [191], representation similarity analysis (RSA) [140], Hilbert-Schmidt Independence Criterion (HSIC) [85] and more.

The principle behind CKA is the generation of a similarity measure between two representations by comparing the similarity structure of both representations. Each similarity structure is produced from the measure of similarity between pairwise examples or data points in a representation. Furthermore, CKA is a generalised extension of HSIC, with the inclusion of normalisation which introduces the property of isotropic scaling invariance. Describing the formulation of the CKA is outside the scope of this work and we refer readers to the original paper

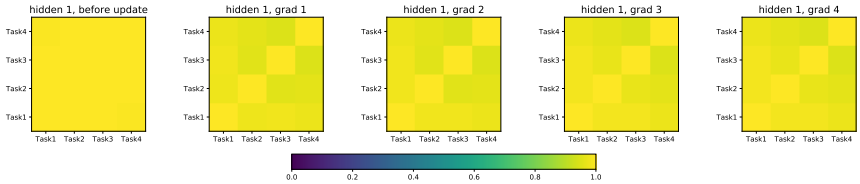


(a) standard policy network, first hidden layer.

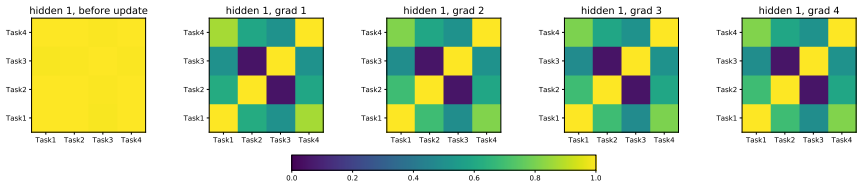


(b) neuromodulated policy network, first hidden layer.

Figure 4.8: Representation similarities between tasks in the 2D Navigation environment.



(a) standard policy network, first hidden layer.



(b) neuromodulated policy network, first hidden layer.

Figure 4.9: Representation similarities between tasks in the CT-graph depth2 environment.

for detailed theoretical formulations. While the RSA similarity index measure employed in [82] is a valid alternative, we chose the CKA due to its robustness to random initialization and enabling comparison within layers of the same network and comparison across networks. Furthermore, CKA has been employed previously in meta-RL setting, e.g., demonstrated in [207].

Analysis: Representation similarities for standard and neuromodulated policy networks across tasks

The per layer output representation similarities between tasks were plotted as heat maps in Figure 4.8 and 4.9. Each heatmap in a row (for example 4.8a) depict the similarity before or after few steps of gradient updates to the layer. Before any gradient updates, the representations are similar between tasks in the figure. After

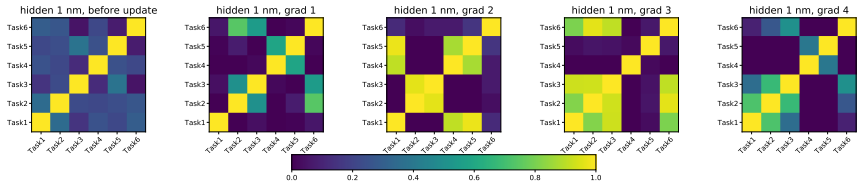
gradient updates, some dissimilarities between tasks begin to emerge. Additional analysis plots are presented in Section 4.9.2.

2D Navigation. For the simple 2D Navigation environment, the plots for the first hidden layer of the standard policy network shown in Figure 4.8a depicts good dissimilarity between tasks, thus highlighting the fact that the learnt representations are sufficient to produce distinct task behaviours. The same is true as well for the first hidden layer of the neuromodulated policy network (see Figure 4.8b). This further justifies why both policies obtained roughly comparable performance in this environment. The simplicity of the problem enables task distinct representations to be obtained easily. Section 4.9.2 contains the plots of the representation similarity for the second hidden layer of both policy networks.

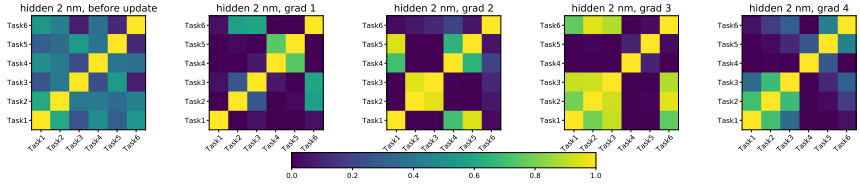
CT-graph. In Figure 4.9a and 4.9b, we compare the representation similarity of the first hidden layer of the standard and neuromodulated policy networks in the CT-graph depth2 environment. We see that representations of the neuromodulated policy are more dissimilar between the tasks than those of the standard policy. Due to the complexity of the environment, the task specific representations required to solve each task are distinct from one another. Therefore, adaptation by fine-tuning the representations of a base network via few gradient steps of parameters update would require a significant jump in the solution space. Standard policy network struggles to enable such jump in the solution space. However, by incorporating neuromodulators that dynamically alters the representations, such jump becomes possible. Section 4.9.2 contains the plots of the representation similarity for the second hidden layer of both policy networks.

Analysis: Representation similarities of the neuromodulatory units across tasks

Now we ask ourselves where the representational diversity (dissimilarity in representations across tasks) comes from. Is the neuromodulatory layer effectively contributing to rich representations as Figure 4.9b appeared to suggest? The analysis we present here shows task representation similarities measured more specifically across the neuromodulatory layers of the proposed architecture. From Figure 4.9b, it appears that such dissimilarity is enhanced by the neuromodulatory activities in the NPN. Again the centered kernel alignment (CKA) was employed and we compare the neuromodulatory activities per layer across different tasks. Figures 4.10, 4.11 and 4.12 present the heat map plots for the 2D navigation, CT-graph depth 2 and ML45 environments (additional plots for other environment are presented in 4.9.2). The non-uniformity in the heatmap plots, in contrast to those of Figure 4.9a, indicates that those layers encode diverse or dissimilar representations across task. We can therefore conclude that the neuromodulatory

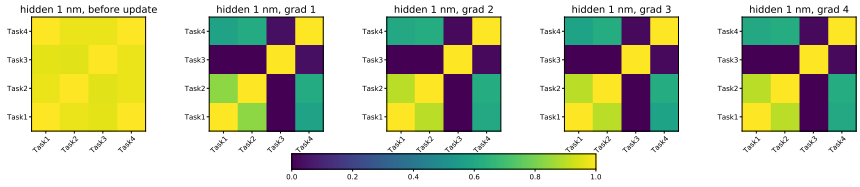


(a) first hidden layer.

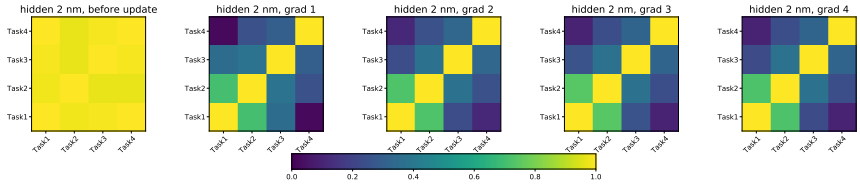


(b) second hidden layer.

Figure 4.10: Representation similarities of neuromodulatory activities h^m between tasks in the 2D Navigation environment across the first hidden layer (a), and the second hidden layer (b) of the network. Non-uniformity across heatmap plots show dissimilar representations emerge from neuromodulatory activity which helped the neuromodulated policy network to solve tasks in complex problem benchmarks.

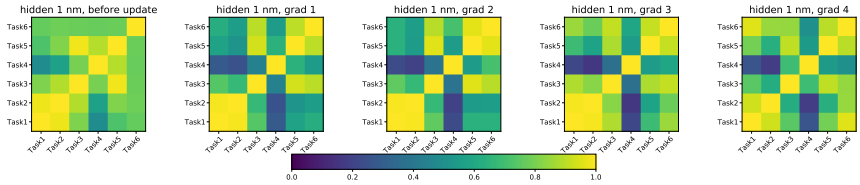


(a) first hidden layer.

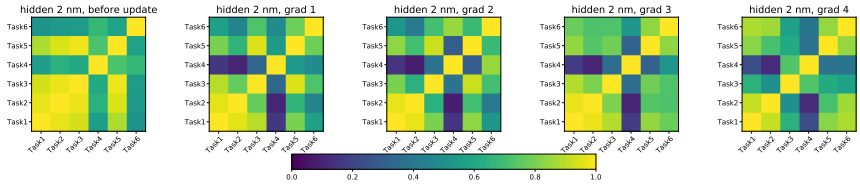


(b) second hidden layer.

Figure 4.11: Representation similarities of neuromodulatory activities h^m between tasks in the CT-graph depth2 environment across the first hidden layer (a), and the second hidden layer (b) of the network. Non-uniformity across heatmap plots show dissimilar representations emerge from neuromodulatory activity which helped the neuromodulated policy network to solve tasks in complex problem benchmarks.



(a) first hidden layer.



(b) second hidden layer.

Figure 4.12: Representation similarities of neuromodulatory activities h^m between tasks in the Meta-World ML45 environment across the first hidden layer (a), and the second hidden layer (b) of the network. Non-uniformity across heatmap plots show dissimilar representations emerge from neuromodulatory activity which helped the neuromodulated policy network to solve tasks in complex problem benchmarks.

activities, when projected onto a layer's standard neurons, produce the desired dissimilar representations across tasks.

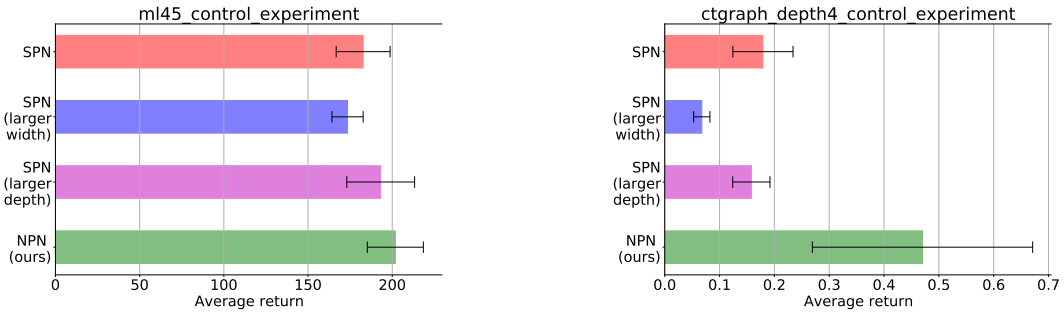
4.6.3 Control Experiments: larger SPN, equaling the number of parameters of the NPN

Since the inclusion of neuromodulators increases the number of parameters in a neuromodulated policy network, a set of control experiments were conducted in which the number of parameters in a standard policy network was configured to approximately match that of a neuromodulated policy network. This was achieved by increasing the size of each hidden layer of the standard policy network (called SPN larger width) in one experiment, and increasing the depth or number of hidden layers by 1, i.e., an additional layer (SPN larger depth) in another experiment. Using CAVIA, experiments were conducted in the CT-graph depth4 and the ML45 meta-world environments, comparing the standard policy network (i.e., the original size), its larger variants and a neuromodulated policy network. The results are presented in Figure 4.13. We observe from the results that the increase in the size of the policy network does not lead to match of the performance of the neuromodulated policy network.

4.7 Discussions

4.7.1 Neuromodulation, gated recurrent networks and attention

The neuromodulatory gating mechanism introduced in this work is reminiscent of the gating in recurrent/memory networks (LSTMs [106] and GRUs [39]). In this



(a) ML45 **(b)** CT-graph depth4

Figure 4.13: Control experiments. Adaptation performance of standard policy network (SPN), a larger SPN variant and neuromodulated policy network (NPN) using CAVIA. Note, the number of parameters in *SPN (larger)* approximately matches that of the NPN in each environment.

respect (with the observation of improved performance as a consequence of neuromodulatory gating in this work), the noteworthy performance demonstrated by meta-RL memory approaches [60, 282] could also be a consequence of such gating mechanisms[4]. Nonetheless, the present study aims to highlight the advantage of a simpler form of gating (i.e., neuromodulatory gating) on a MLP feedforward network, and thus could help to pinpoint the advantage of such dynamics in isolation. Furthermore, the advantage of our approach over gated recurrent variants is somewhat similar to the advantages derived from decoupling attention mechanism from recurrent models (where it was originally introduced) and applying it to MLP networks (i.e., Transformer models) [276]. By decoupling neuromodulatory (gating) mechanism from recurrent models and applying it to MLP models (as in our work), the advantages of faster training and better parallelization were achieved while maintaining the benefit of neuromodulatory gating. Therefore, our proposed approach is faster to train and more parallelizable in comparison to memory variants, while maintaining the advantages that neuromodulatory gating offers. Memory based approaches will still be required for problems where memory is advantageous such as sequential data processing and POMDPs.

Neuromodulation shares some similarity with the attention mechanism due to the gating of representations, particularly in MLP networks. However, they differ in their purpose and setup. The introduced neuromodulatory gating focused on enabling fast adaptation of policies via rapid alteration of the network representations during optimization. Whereas, attention deals with the identification and

[4]Although not the focus of this work, we ran an experiment using RL^2 (a memory based meta-RL method) in the ML45 environment and achieved an average meta-test success rate performance of 10%, which is comparable to the results obtained using neuromodulatory gating mechanism. See Section 4.7.3 for discussions about performance in meta-world, in relation to the performance reported in the original meta-world paper.

capture of relationships between elements in an input sequence, with the goal of producing dynamic representations relative to the position of the elements in the sequence.

4.7.2 Task similarity measure and robust benchmarks

Increasing task complexity was presented in this work by moving from simple 2D point navigation environment to half-cheetah locomotion and then to the complex robotic arm setup of the Meta-World environment. Furthermore, exploiting the benefits of configurable parameters in the CT-graph environment, we were able to control the complexity in the environment. Overall, task complexity was viewed through the perspective of task similarity (i.e., environments with dissimilar task were viewed as more complex and vice versa). Despite these efforts, a precise measure of task complexity and similarity was not clearly outlined in this work and this is widely the case in meta-RL literatures. There is a need for the development of precise metrics for measuring task similarity and complexity in the field. The CT-graph with its configurable parameters allow for tasks to be mathematically defined, which is a first step towards alleviating this issue. However, a separate future research investigation would be necessary to develop explicit metrics that can be incorporated into meta-RL benchmarks.

We hypothesize that such a task similarity metric should be able to capture the precise change points in a task relative to other tasks. For example, a useful metric could be one that capture task change either as a function of change in reward, or state space, or transition function, or a combination of these factors. Most benchmarks in meta-RL have been focused on task change as reward function change. However, a more robust benchmark could include the aforementioned change points in order to further control the complexity. The CT-graph, Meta-world, and the recently developed Alchemy [281] environment are examples of benchmarks with early stage work in this direction, albeit implicitly. Therefore, the development of a precise measure of task similarity and complexity, as well as robust benchmarks with configurable change points (i.e., reward, state/input, and transition) would be highly beneficial to the meta-RL field.

4.7.3 Meta-World Performance

The performance difference between the baselines in the original Meta-World paper and the present submission is due to an update of the reward function in the recent version of the Meta-World environment[5][6]. Issues about the reward function of the originally released Meta-World was reported and discussed in https://github.com/rlworkgroup/metaworld/issues/226. This led the envir-

[5]https://github.com/rlworkgroup/metaworld/pull/312
[6]https://github.com/rlworkgroup/metaworld/commit/ffe2c

onment developers to rewriting many of the reward functions in the environment that are now part of the current version (informally referred to as v2 environments). The results reported in the original Meta-World paper used the old (and now replaced) reward function, while the results in the present submission are based on a recent version cited above. It is nevertheless possible that the baseline results could be improved with better hyperparameter tuning, although the same is true for the novel approach that we propose. As we aim to observe performance differences between the neuromodulated meta-RL and the standard meta-RL, we did not perform hyperparameter search and tuning.

4.8 Conclusion and Future Work

This contribution introduced an architectural extension of the standard meta-RL policy networks to include a neuromodulatory mechanism, investigating the beneficial effect of neuromodulation when augmenting existing meta-RL frameworks (i.e., neuromodulation as complementary tool to meta-RL rather than competing). The aim is to implement richer dynamic representations and facilitate rapid task adaptation in increasingly complex problems. The effectiveness of the proposed approach was evaluated in meta-RL setting using CAVIA and PEARL algorithms. In the experimental setup across environments of increasing complexity, the neuromodulated policy network significantly outperformed the standard policy network in complex problems while showcasing comparable performance in simpler problems. The results highlight the usefulness of neuromodulators to enable fast adaptation via rich dynamic representations in meta-RL problems. The architectural extension, although simple, presents a general framework for extending meta-RL policy networks with neuromodulators that expand their ability to encode different policies. The projected neuromodulatory activity can be designed to perform other functions apart from the one introduced in this work e.g., gating plasticity of weights, or including different neuromodulators in the same layer. The neuromodulatory extension could also be tested with a recurrent meta-RL policy, with the goal of enhancing the memory dynamics of the policy. Our analysis indicates that this framework is most suited to problems that require rapid change in optimal representations across tasks, while its advantage is reduced when tasks can be solved using similar representations.

Acknowledgement

This research contribution is based upon work supported by the United States Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-18-C0103. Any opinions, findings and conclusions or recommendations expressed in this material are those of the

author(s) and do not necessarily reflect the views of the United States Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA).

4.9 Supplementary Materials

4.9.1 Pseudocode

A code snippet demonstrating the extension of the fully connected layer with neuromodulation is presented below using PyTorch code style.

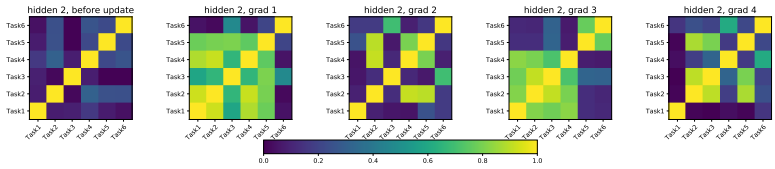
```
class NMLinear(Module):
  def __init__(self, in_features, out_features, nm_features, bias=True, gate=None):
    super(NMLinear, self).__init__()
    self.in_features = in_features
    self.out_features = out_features
    self.nm_features = nm_features
    self.std = nn.Linear(in_features, out_features, bias=bias)
    self.in_nm = nn.Linear(in_features, nm_features, bias=bias)
    self.out_nm =nn.Linear(nm_features, out_features, nm_features)
    self.in_nm_act = F.relu
    self.out_nm_act = torch.tanh
    self.gate = gate

  def forward(self, data, params=None):
    output = self.std(data)
    mod_features = self.in_nm_act(self.in_nm(data))
    projected_mod_features = self.out_nm_act(self.out_nm(mod_features))
    if self.gate == 'strict':
        projected_mod_features = torch.sign(projected_mod_features)
        projected_mod_features[projected_mod_features == 0.] = 1.
    output *= projected_mod_features
    return output
```

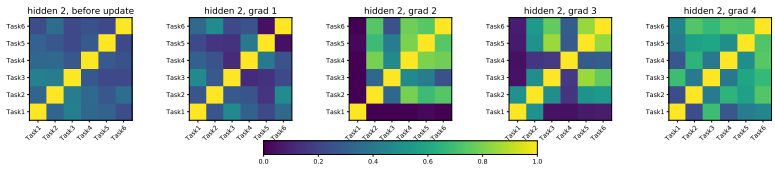
The full implementation (including experimental setup and test scripts) is open sourced at https://github.com/dlpbc/nm-metarl. The codebase is an extension of the original CAVIA and PEARL open source (MIT license) implementations that can be found at https://github.com/lmzintgraf/cavia and https://github.com/katerakelly/oyster respectively.

4.9.2 Additional Analysis Plots

This section presents additional analysis plots of the representation similarity across tasks for the standard and neuromodulated policy networks in the various evaluation environments employed in this work. The additional plots further highlights the usefulness of neuromodulation to facilitate efficient (distinct) representations across tasks in problems of increasing complexity as earlier showcased in Section 4.6.2.

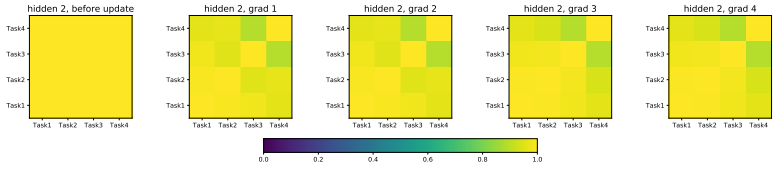


(a) standard policy network, second hidden layer.

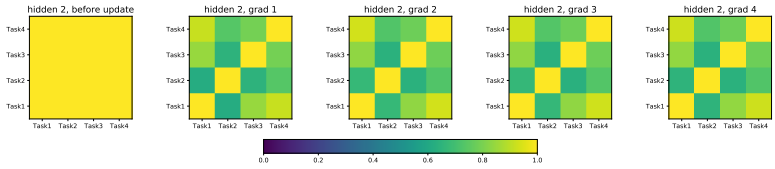


(b) neuromodulated policy network, second hidden layer.

Figure 4.14: Representation similarities between tasks in the 2D Navigation environment.



(a) standard policy network, second hidden layer.



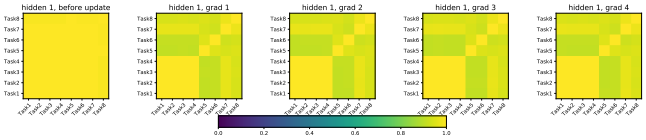
(b) neuromodulated policy network, second hidden layer.

Figure 4.15: Representation similarities between tasks in the CT-graph-depth2 environment.

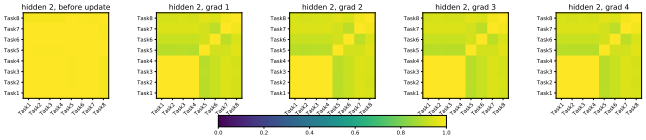
Second hidden layer: 2D Navigation and CT-graph depth2 environments

2D Navigation: Figure 4.14 present the representation similarity between tasks, across inner loop gradient updates for the second hidden layer of both policy networks in the 2D navigation environment. Again, similar patterns as highlighted in the first hidden layer of the policy networks in Figure 4.8 emerge. Both networks are able to learn good (dissimilar) representations between tasks after few steps of inner loop gradient update. Also, both networks, before any gradient update, already have some level of representation disimilarity between tasks. Thus, this further highlights the fact the 2D Navigation environment has a low complexity and requires very little adaptation of network parameters.

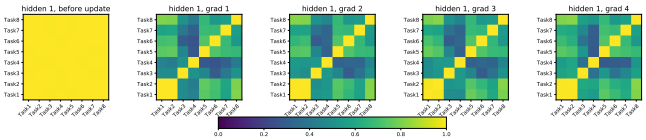
CT-graph depth2: Figure 4.15 present the representation similarity between tasks, across inner loop gradient updates for the second hidden layer of both policy



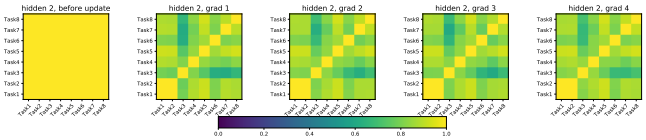
(a) standard policy network, first hidden layer.



(b) standard policy network, second hidden layer.



(c) neuromodulated policy network, first hidden layer.



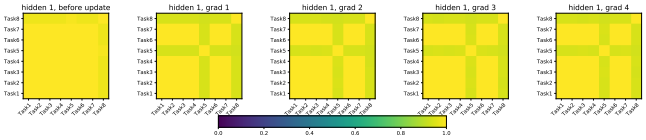
(d) neuromodulated policy network, second hidden layer.

Figure 4.16: Representation similarities between tasks in the CT-graph depth3 environment.

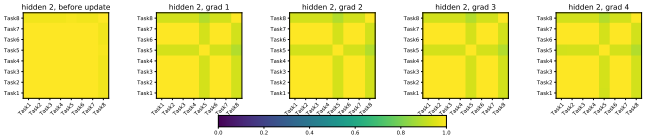
networks in the CT-graph depth2 benchmark. With increased problem complexity in comparison to the 2D navigation, only the neuromodulated policy network succeeds in learning distinct representations across tasks. The distinct representations thus allow the neuromodulated policy network to adapt optimally across tasks while the standard policy network struggles, as indicated by the performance plot in Figure 4.6a.

CT-graph depth3 and depth4 Environments

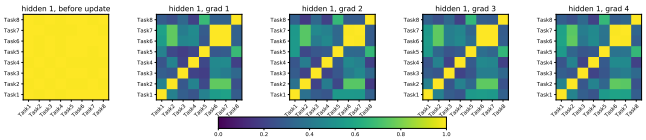
The representation similarity plots (across inner loop gradient updates) between tasks of the hidden layers of both the standard and the neuromodulated policy networks in the CT-graph depth3 and depth4 environment instances are presented in Figure 4.16 and 4.17. Again, as observed in Section 4.6.2, the standard policy network still struggles to learn distinct representations for each task, whereas, the neuromodulated policy network is able to learn the required task-specific representations. Hence, the neuromodulated policy network is able to perform optimally across tasks. This explains the difference in performance (Figures 4.6b and 4.6c) between the policy networks.



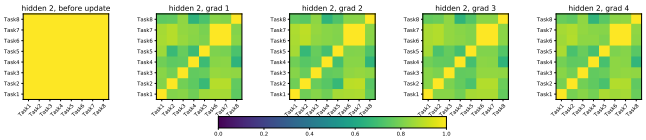
(a) standard policy network, first hidden layer.



(b) standard policy network, second hidden layer.



(c) neuromodulated policy network, first hidden layer.



(d) neuromodulated policy network, second hidden layer.

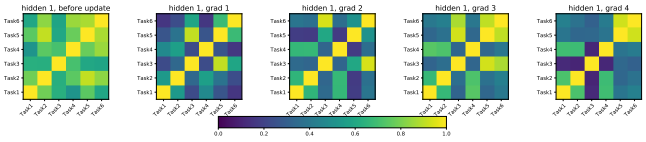
Figure 4.17: Representation similarities between tasks in the CT-graph depth4 environment.

Half-Cheetah and Meta-World Environments

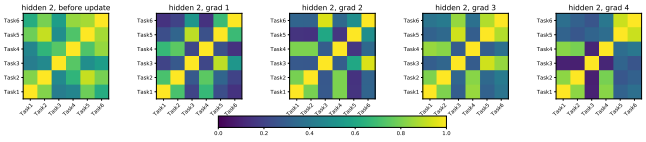
The CAVIA policies analysis plots for the half-cheetah and meta-world benchmarks are presented in this section.

Half-Cheetah: Figure 4.18 and 4.19 shows the representation similarity plots of the standard and neuromodulated policy networks for the Half-Cheetah direction and velocity environments respectively. In this setting where the problems are of simple to medium complexity, both policy networks are able to learn efficient (dissimilar) representations between tasks, further buttressing the discussions in Section 4.6.2. In fact, we observe that the standard policy network learns better dissimilar representations across tasks for the half-cheetah direction environment due to the simplicity of the tasks in the environment in comparison to the velocity environment.

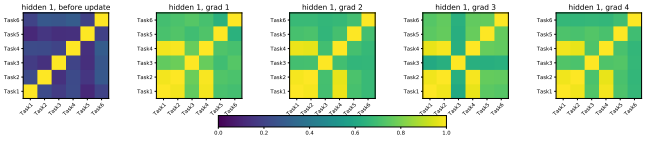
Meta-World: Also, Figure 4.20 and 4.21 depicts the representation similarity plots for the ML1 and ML45 meta-world environment. Similar to the observations in Section 4.6.2, we observe again that as the problem complexity increase (i.e., from ML1 to ML45), the neuromodulated policy network produces better (dissimilar) representations across the sampled tasks in comparison to the standard policy network.



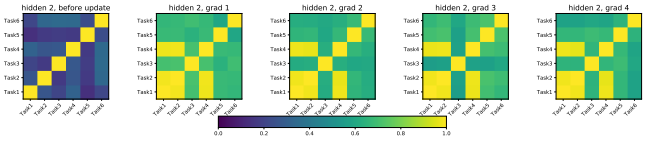
(a) standard policy network, first hidden layer.



(b) standard policy network, second hidden layer.

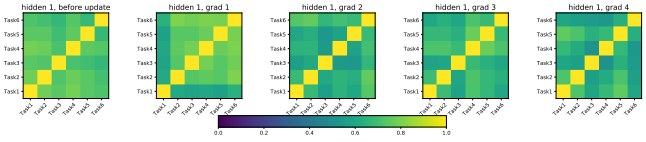


(c) neuromodulated policy network, first hidden layer.

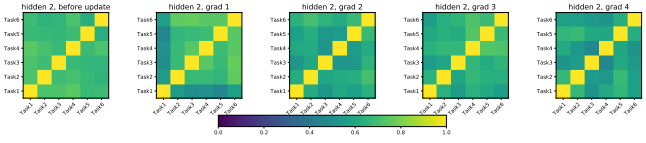


(d) neuromodulated policy network, second hidden layer.

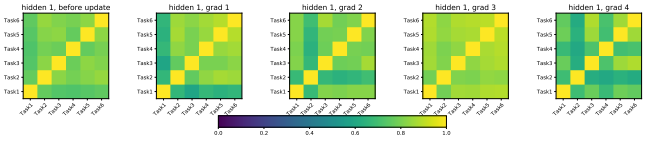
Figure 4.18: Representation similarities between tasks in the Half-Cheetah Direction environment.



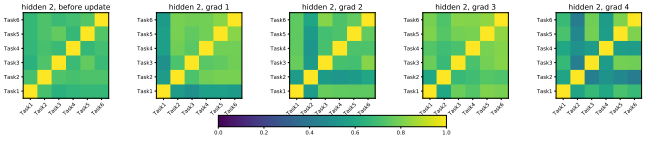
(a) standard policy network, first hidden layer.



(b) standard policy network, second hidden layer.

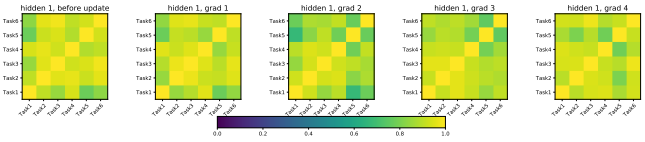


(c) neuromodulated policy network, first hidden layer.

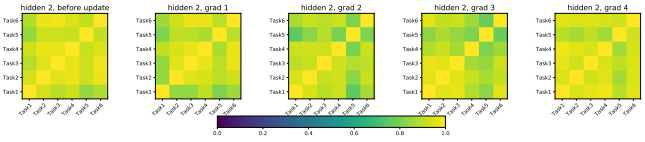


(d) neuromodulated policy network, second hidden layer.

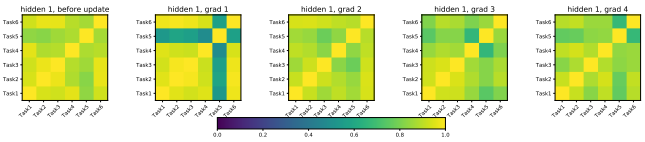
Figure 4.19: Representation similarities between tasks in the Half-Cheetah Velocity environment.



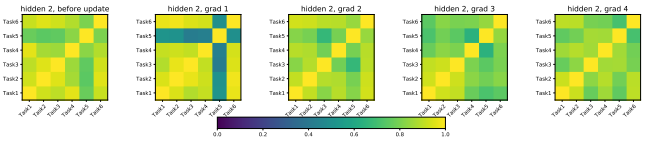
(a) standard policy network, first hidden layer.



(b) standard policy network, second hidden layer.

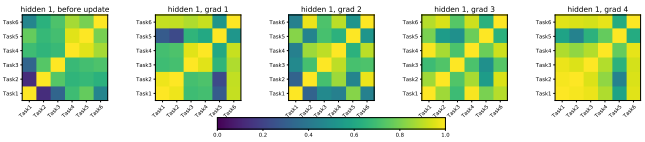


(c) neuromodulated policy network, first hidden layer.

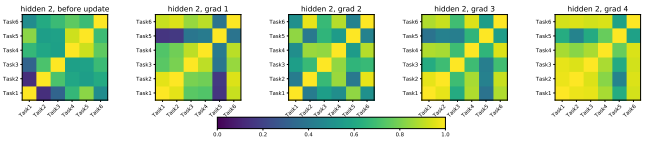


(d) neuromodulated policy network, second hidden layer.

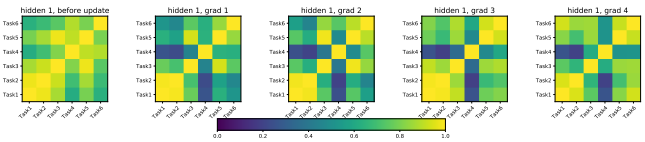
Figure 4.20: Representation similarities between tasks in the ML1 (meta-world) environment.



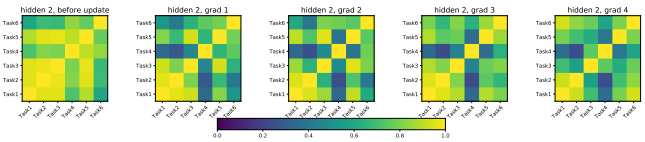
(a) standard policy network, first hidden layer.



(b) standard policy network, second hidden layer.

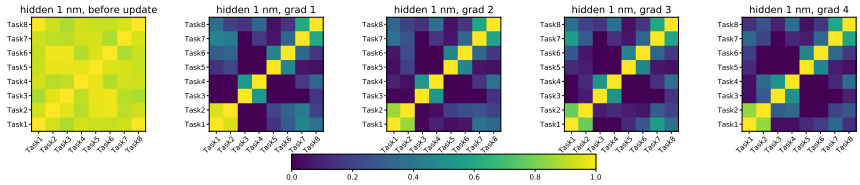


(c) neuromodulated policy network, first hidden layer.

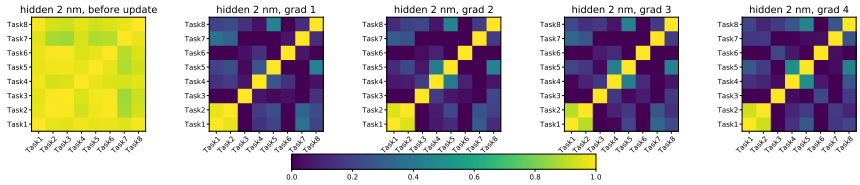


(d) neuromodulated policy network, second hidden layer.

Figure 4.21: Representation similarities between tasks in the ML45 (meta-world) environment.

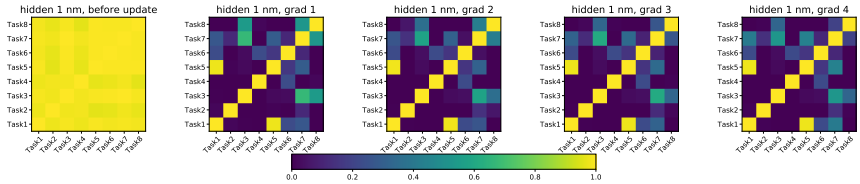


(a) first hidden layer.

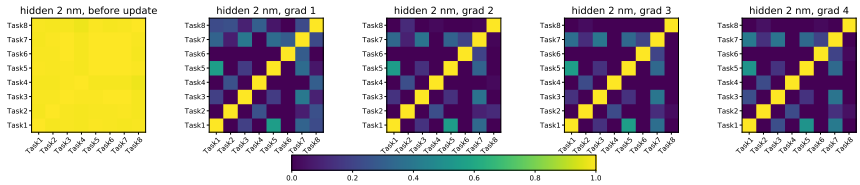


(b) second hidden layer.

Figure 4.22: Representation similarities of neuromodulatory activities h^m between tasks in the CT-graph depth3 environment.



(a) first hidden layer.

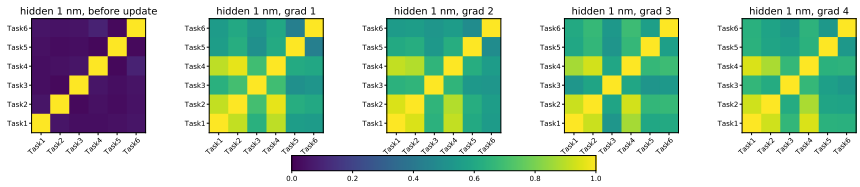


(b) second hidden layer.

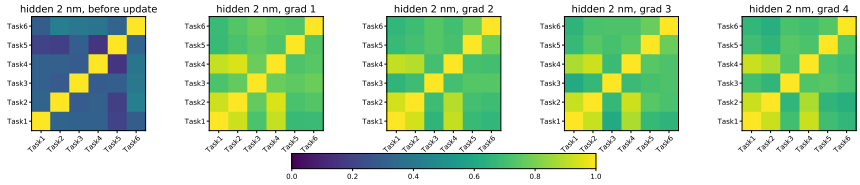
Figure 4.23: Representation similarities of neuromodulatory activities h^m between tasks in the CT-graph depth4 environment.

Representation similarity of neuromodulatory activities across tasks

The representation similarity plots of the neuromodulatory activities h^m across tasks are presented for the CT-graph depth 3 and 4, Half-Cheetah Velocity and Direction, and Meta-World ML1 environments. Again, we observe dissimilar representation across tasks for the neuromodulatory activities. Hence, neuromodulation facilitates the dynamic representations in the policy network enabling optimal adaptation behaviour in complex problems with dissimilar tasks.

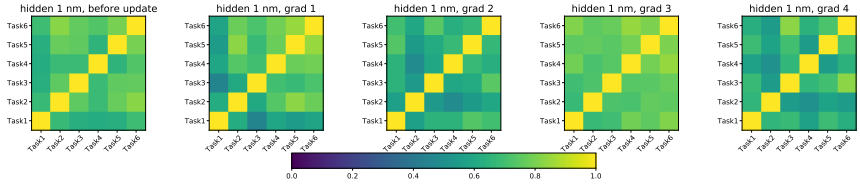


(a) first hidden layer.

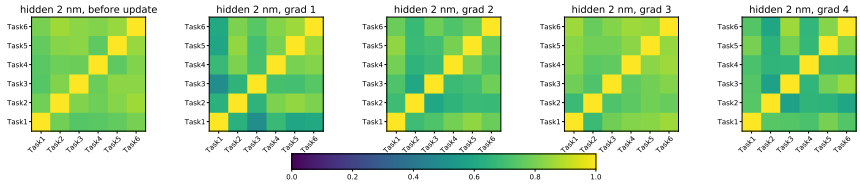


(b) second hidden layer.

Figure 4.24: Representation similarities of neuromodulatory activities h^m between tasks in the Half-Cheetah Direction environment.

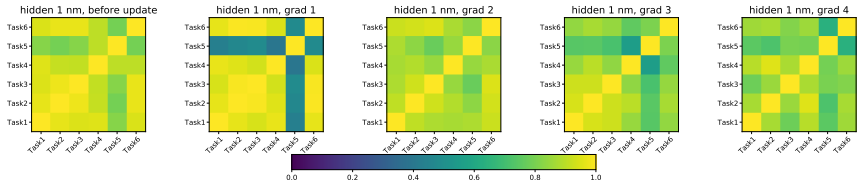


(a) first hidden layer.

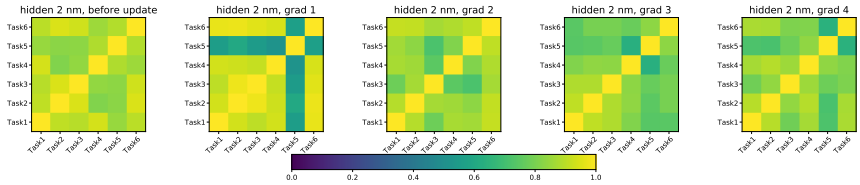


(b) second hidden layer.

Figure 4.25: Representation similarities of neuromodulatory activities h^m between tasks in the Half-Cheetah Velocity environment.



(a) first hidden layer.



(b) second hidden layer.

Figure 4.26: Representation similarities of neuromodulatory activities h^m between tasks in the Meta-World ML1 environment.

Chapter 5

Lifelong Reinforcement Learning with Modulating Masks

5.1 Introduction

Lifelong reinforcement learning (LRL) is a recent and active area of research that seeks to enhance current RL algorithms with the following capabilities: learning multiple tasks sequentially without forgetting previous tasks, exploiting knowledge of previous tasks to accelerate learning of new tasks, or building solutions to increasingly complex tasks. The remarkable progress of RL in recent years, stemming in particular from the introduction of deep RL [188], has demonstrated the potential of algorithms that can learn to act in an environment with complex inputs and rewards. However, the limitation of learning only one task means that such algorithms have a narrow focus, and might not be able to scale to complex tasks that first require learning of sub-tasks. It is arguable that acquiring knowledge of multiple tasks over a lifetime, generalizing across such tasks, exploiting acquired knowledge to master new tasks more quickly, or composing simple tasks to solve more complex tasks are necessary to develop more powerful AI systems that better mimic biological intelligence.

LRL shares some of the objectives of lifelong supervised learning (LSL) [203, 90, 52], but due to the unique objective and domains of RL, it has developed into a separate field [128]. For example, tasks in LRL can vary across reward function, input distribution or transition function, while tasks in LSL largely vary in the input distribution. A variety of recently developed approaches in LSL can be placed under three categories: memory methods, synaptic consolidation methods, and parameter isolation or modular methods. In LRL, [128] propose a taxonomy in which approaches can be classified as explicit knowledge retention, leveraging shared structures, and learning to learn. It can be appreciated that LRL exploits most advances in LSL while also developing RL-specific algorithms.

Among the LSL categories mentioned above, parameter isolation methods have

shown state-of-the-art results in classification with approaches such as Progressive Networks [224, 297], PackNet [174], Piggyback [173] and Supermasks [287]. The key concept in such methodologies is to find smart and efficient ways to isolate parameters for each task. In [174], redundancies in large networks are exploited to generate masks via iterative pruning, and thus free parameters to be used only for specific tasks, thus "packing" multiple tasks in one single network. Other approaches instead use the concept of directly learned masks [310, 210, 287] to select parts of a backbone network for each task. Instead of learning the parameters of a backbone network, which are set randomly, masking approaches focus on learning a multiplicative (or modulating) mask for each task. Masks can be effective in binary formats, enabling or disabling parts of the backbone network while requiring low memory. Interestingly, masks can be interpreted as *modulatory* mechanisms with biological inspiration [143] because they exploit a gating mechanism on parameters or activations.

Surprisingly, while the application of masks has been tested extensively in LSL for classification, very little is known on their effectiveness in LRL. Also, the current mask methods lack the ability of exploit knowledge from previously learned task (forward transfer), since each mask is derived independently. This study introduces the use of directly learned masks with policy optimization algorithms, specifically PPO [235] and IMPALA [65]. We explore (1) the effectiveness of masks to maximize lifelong learning evaluation metrics, and (2) the suitability of masks to reuse knowledge and accelerate learning (forward transfer). To demonstrate the first point, we test the approach on RL curricula with the Minigrid environment [38], the CT-graph [254, 250], Metaworld [299], and ProcGen [41], and assess the lifelong learning metrics [197, 20] when learning multiple tasks in sequence. To demonstrate the second point, we exploit linear combinations of masks and investigate learning with curricula in which tasks have similarities and measure the forward transfer. The investigation of the second point give rise to the understanding of the effectiveness of mask knowledge reuse in RL, the initial parameter configuration ideal for mask reuse, and the benefits obtained when task curriculum grow in complexity or contains interfering tasks. To ensure reproducibility, the hyper-parameters for the experiments are reported in Appendix 5.9.2. The code is published at `https://github.com/dlpbc/mask-lrl`.

Contributions. This study introduces the use of directly learned modulating masks in LRL. Although masking has been previously applied in LRL via PackNet, such masks are generated from iterative pruning and not directly learned. In particular, we employ masks to learn curricula of RL tasks and assess lifelong learning metrics and forward transfer for both discrete and continuous problems. We show that (1) modulating masks are effective in LRL because they maximize

lifelong learning metrics when they operate in combination with RL algorithms such as PPO and IMPALA; (2) the gradient search on parameters of a linear combination of masks results in the acceleration of learning of new tasks when such new tasks share properties with previously learned tasks.

5.2 Related work

5.2.1 Deep Reinforcement Learning (DeepRL)

The use of deep networks as function approximators in reinforcement learning (RL) has garnered widespread adoption, showcasing results such as learning to play video games [188, 274, 245, 123, 187, 14, 65], and learning to control actual and simulated robots [156, 233, 235, 75, 89]. These algorithms enable the agent to learn how to solve a single task in a given evaluation domain. In a lifelong learning scenario with multiple tasks, they suffer from lifelong learning challenges such as catastrophic forgetting. Nevertheless, they serve as the basis for lifelong learning algorithms. For example, DQN [188] was combined with the elastic weight consolidation (EWC) algorithm to produce a lifelong learning DQN agent [132, 127].

5.2.2 Lifelong (Continual) Learning

Several advances have recently been introduced in lifelong (continual) learning [273, 203, 90, 52, 128], addressing challenges such as maintaining performance on previously learned tasks while learning a new task (overcoming forgetting), reusing past knowledge to rapidly learn new tasks (forward transfer), improving performance on previously learned tasks from newly acquired knowledge (backward transfer), the efficient use of model capacity to reduce intransigence, and reducing or avoiding interference across tasks [127]. A large body of work focused on lifelong learning in the supervised learning domain and overcoming the challenge of forgetting [178].

Lifelong learning algorithms can be clustered into key categories such as synaptic consolidation approaches [132, 303, 4, 136], memory approaches [167, 302, 37, 219, 158], modular approaches [224, 174, 173, 287, 178] or a combination of the above. Synaptic consolidation approaches tackle lifelong learning by discouraging the update of parameters useful for solving previously learned tasks through a regularization penalty. Memory approaches either store and replay samples of previous and current tasks (from a buffer or a generative model) during training [167, 37, 86, 279] or project the gradients of the current task being learned in an orthogonal direction to the gradients of previous tasks [302, 66, 225, 158]. Memory methods aim to keep the input-output mapping for previous tasks unchanged while learning a new task. Modular approaches either expand the network

as new tasks are learned [224, 297, 176] or select sub-regions of a fixed-sized network (via masking) for each tasks [287]. The masks can be applied on (i) the neural representation [249, 243], (ii) or synapses, where they are directly learned [287] or derived through iterative pruning [174]. The masks can be viewed as a form induced sparsity in the neural lifelong learner [280, 108].

In the RL domain, a lifelong learner is usually developed by combining a standard deep RL algorithm, either on-policy or off-policy (for example, DQN, PPO [235], or SAC [89]), with a lifelong learning algorithm. CLEAR [219] is a lifelong RL that combines IMPALA with a replay method and behavioural cloning. Progress & Compress [238] demonstrated the combination of IMPALA with EWC and policy distillation techniques. Other notable methods include the combination of a standard deep RL agent (SAC) with mask derived from pruning in PackNet [174, 239] as demonstrated in the Continual World robotics benchmark [286]. In addition, [178] developed an algorithm combining neural composition, offline RL, and PPO to facilitate knowledge reuse and rapid task learning via functional composition of knowledge. To tackle a lifelong learning scenario with interference among tasks, current methods [126, 127] employ a multi-head policy network (i.e., a network with shared feature extractor layers connected to different output layers/heads per task) that combine a standard RL algorithm (e.g. DQN or SAC) with synaptic consolidation methods. In another class of LRL approach, [121] demonstrated a lifelong RL agent that combined DQN with multiple-time scale learning at synaptic level [29], and the model was adapted to multiple timescale learning at policy level [122] via the combination of knowledge distillation and PPO.

5.2.3 Modulation

Neuromodulatory processes [11, 22] enable the dynamic alteration of neuronal behaviour by affecting synapses or the neurons connected to synapses. Modulation in artificial neural networks [67, 58] draws inspiration from modulatory dynamics in biological brains that have proven particularly effective in reward-based environments [236, 2], in the evolution of reward-based learning [251, 255], and in meta RL [25]. Modulatory methods in lifelong learning are set up as masks that alter either the neural activations [243] or weights of neural networks [174, 173, 287, 138]. The key insight is the use of a modulatory mask (containing binary or real values) to activate particular network sub-regions and deactivate others when solving a task. Therefore, each task is solved by different sub-regions of the network. For each task, PackNet [174] trains the model based on available network capacity and then prunes the network to select only parameters that are important to solving the task. A binary mask representing important and unimportant parameters is then

generated and stored for that task, while ensuring that important parameters are never changed when learning future tasks. PiggyBack [173] keeps a fixed untrained backbone network, while it trains and stores mask parameters per task. During learning or evaluation of a particular task, the mask parameters for the task are discretized and applied to the backbone network by modulating its weights. The Supermask [287] is a generalization of the PiggyBack method that uses a k-winner approach to create sparse masks.

5.3 Background

5.3.1 Problem Formulation

A reinforcement learning problem is formalized as a Markov decision process (MDP), with tuple $\mathcal{M} = \langle \mathcal{S}, \mathcal{A}, p, r, \gamma \rangle$, where $\mathcal{S}$ is a set of states, $\mathcal{A}$ is a set of actions, $p : \mathcal{S} \times \mathcal{A} \to \mathcal{S}$ is a transition probability distribution $p(s_{t+1}|s_t, a_t)$ of the next state given the current state and action taken at time t, $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function that produces a scalar value based on the state and the action taken at the current time step, $\gamma \to [0, 1]$ is the discount factor that determines how importance future reward relative to the current time step t. An agent interacting in the MDP behaves based on a defined policy (either a stochastic π or a deterministic μ policy). The objective of the agent is to maximize the total reward achieved, defined as an expectation over the cumulative discounted reward $\mathbb{E}[\sum_{t=0}^{H-1} \gamma^t r(s_t, a_t)]$ in a finite horizon H. It achieves this by learning an optimal policy.

In a lifelong learning setup, a lifelong RL agent is exposed to a sequence of tasks $\mathcal{T}$. Given N tasks, the agent is expected to maximize the RL objective for each task in $\mathcal{T} = \{\tau_1, \tau_2, \ldots, \tau_n\}$. As the agent learns one task after another in the sequence, it is required to maintain (avoid forgetting) or improve (backward transfer of knowledge) performance on previously learned tasks. A desired property of such an agent is the ability to reuse knowledge from previous tasks to rapidly learn the current or future tasks.

5.3.2 Modulating masks

The concept of modulating masks has recently emerged in the area of supervised classification learning [173, 243, 210, 287, 249, 138]. Masks work by modulating the weights or the neural representations of a network, and they can be directly learned (optimized) or derived via iterative pruning [108], with a goal of implementing sparsity in the network. The learned mask approach is employed in this work, and masks are used to modulate the weights of a network. By modulating the weights, sub-regions of the network is activated, while other regions are deactivated. Given a network with layers $1, \ldots L$, with each layer l containing parameters $W^l \in \mathbb{R}^{m \times n}$,

for each layer l, a score parameter $S^l \in \mathbb{R}^{m \times n}$ is defined.

During a forward pass (in training or evaluation), a binary mask $M^l \in \{0,1\}^{m \times n}$ is generated from S^l based on an operation $g(S^l)$ according to one of the following: (i) a threshold ϵ (where values greater than ϵ are set to 1, otherwise 0) [173], or (ii) top-k values (the top $k\%$ values in S^l yields 1 in M^l, while the rest are set to 0) [210], or (iii) probabilistically sampled from a Bernoulli distribution, where the p parameter for the distribution is derived from the sigmoid function $\sigma(S^l)$ [310]. An alternative approach is the generation of ternary masks (i.e., with values $\{-1, 0, 1\}$) from S^l, as introduced in [138]. The binary or ternary mask modulates the layer's parameters (or weights), thus activating only a subset of the W^l.

Given an input sample $\mathbf{x} \in \mathbb{R}^m$, a forward pass through the layer is given as $f(\mathbf{x}, W^l, S^l) = (W^l \odot M^l) \cdot \mathbf{x}$, where $\odot$ is the element wise multiplication operation. Only the score parameters S^l are updated during training, while the weights W^l are kept fixed at their initial values. For brevity, the weights and scores across layers are denoted as W and S.

The training algorithm updating S^l depends on the operation used to generate the binary mask. When the binary masks are generated from Bernoulli sampling, standard backpropagation is employed. However, in the case of thresholding or selecting top $k\%$, an algorithm called edge-popup is employed, which combines backpropagation with the straight-through gradient estimator (i.e., where the gradient of the binary mask operation is set to identity to avoid zero-value gradients) [28, 43].

In a lifelong learning scenario with multiple tasks, for each task τ_k, a score parameter S_k is learned and used to generate and store the mask M_k. During evaluation, when the task is encountered, the mask learned for the task is retrieved and used to modulate the backbone network. Since the weights of the network are kept fixed, it means there is no forgetting. However, this comes at the cost of storage (i.e., storing a mask for each task).

5.4 Methods

We first introduce the adaptation of modulating masks to RL algorithms (Section 5.3.2). Following, we hypothesize that previously learned masks can accelerate learning of unseen tasks by means of a linear combination of masks (Section 5.4.2). Finally, we suggest that continuous masks might be necessary to solve RL tasks in continuous environments (Section 5.4.3).

5.4.1 Modulatory masks in Lifelong RL

We posit that a stochastic control policy $\pi_{\theta,\Phi}$ can be parameterized by θ, the weights of a fixed backbone network, and $\Phi = \{\phi_1 \dots \phi_k\}$, a set of mask score

parameters for tasks $1 \dots k$. ϕ_k are the scores of all layers of the network for task k, comprising the layers $1 \dots L$, i.e., $\phi_k = \{S_k^1 \dots S_k^L\}$. The weights θ of the network are randomly initialized using the signed Kaiming constant method [210], and are kept fixed during training across all tasks. The mask score parameters Φ are trainable, but only ϕ_k is trained when the agent is exposed to task k. To reduce memory requirements, [287] discovered that masks can be reduced to binary without significant loss in performance. We test this assumption by binarizing masks using an element-wise thresholding function

$$g(\phi_k) = \begin{cases} 1 & \phi_{k,\{i,j\}} > \epsilon \\ 0 & \text{otherwise} \end{cases} \tag{5.1}$$

where ϵ is set to 0. The resulting LRL algorithm is described in Algorithm 3. To assess this algorithm, we paired it with the on-policy PPO [235] algorithm and the off-policy IMPALA [65] algorithm.

Algorithm 3 Lifelong RL Algorithm with modulating masks

Require: Number of tasks N, maximum training steps per task P
Require: Rollout length (steps) z
Initialize policy $\pi_{\theta,\Phi}$
for k $\leftarrow$ 1 ... N **do**
  Set task k
  Set *steps* $\leftarrow 0$
  while step $<$ P **do**
    Rollout experiences $\{(s_1, a_1, r_1, s'_1) \dots (s_z, a_z, r_z, s'_z)\}$ using π_{θ,ϕ_k}.
    Update *steps* $\leftarrow$ *steps* $+ z$
    Compute loss $\mathcal{L}_k(\pi_{\theta,\phi_k})$ based on an RL algorithm objective.
    Compute gradients $\nabla_{\phi_k}\mathcal{L}_k(\pi_{\theta,\phi_k})$ with respect to ϕ_k.
    Update mask score parameter for task k, $\phi_k \leftarrow \phi_k - \alpha\nabla_{\phi_k}\mathcal{L}_k(\pi_{\theta,\phi_k})$
  end while
  Store mask score ϕ_k or the binary mask $g(\phi_k)$
end for

5.4.2 Exploiting previous knowledge to learn new tasks

One assumption in LRL, often measured with metrics such as forward and backward transfer [197], is that some tasks in the curriculum have similarities. Thus, previously acquired knowledge, stored in masks, may be exploited to learn new tasks. To test this hypothesis, we propose an incremental approach to using previously learned masks when facing a new task. Rather than learn a large number of masks and infer which mask is useful for solving a task at test time via gradient optimized linear combination, as in [287], we instead start to exploit previous knowledge from any small number of masks combined linearly, plus a trainable

random mask $\phi_{k+1} = \{S^1_{k+1} \dots S^L_{k+1}\}$. The intuition is that strong linear combination parameters can be discovered quickly if similarities are strong, otherwise more weight is placed on the newly trained mask.

A new mask at layer l is given by

$$S^{l,lc} = \beta^l_{k+1} S^l_{k+1} + \sum_{i=1}^{k} \beta^l_i S^{l,*}_i \tag{5.2}$$

where $S^l_{k+1} \in \phi_{k+1}$ are the scores of layer l in task $k+1$, $S^{l,lc}$ denotes the transformed scores for task $k+1$ after the linear combination (lc) operation, $S^{l,*}_i$ denotes the optimal scores for previously learned task i, and $\beta^l_1, \dots \beta^l_{k+1}$ are the weights of the linear combination (at layer l). To maintain a normalized weighted sum, a Softmax function is applied to the linear combination parameters before they are applied in Equation 5.2. When no knowledge is present in the system, the first task is learned starting with a random mask. Task two is learned using $\bar{\beta}_1 = 0.5$, weighting task one's mask, and $\bar{\beta}_2 = 0.5$, weighting the new random mask. The third task will have $\bar{\beta}_1 = \bar{\beta}_2 = \bar{\beta}_3 = 0.33$, and so on. Note that $\bar{\beta}_k$ denotes a vector of size L which contains the co-efficient parameters for task k across all L layers of the network (i.e., $\bar{\beta}_k = \{\beta^1_k \dots \beta^L_k\}$).

In short, two approaches can be devised. The first one in which each mask is learned independently of the others. Experimentation of this approach will determine the baseline capabilities of modulating masks in LRL. We name this *Mask Random Initialization* (Mask_{RI}). The second approach attempts to exploit the knowledge acquired so far during the curriculum learning by using a linear combination of masks to learn a new one. Experimentation of this second approach will determine the capabilities of modulating masks to exploit previously learned knowledge. We name this second approach *Mask Linear Combination* (Mask_{LC}). The idea is graphically summarized in Figure 5.1.

It can be noted that, as the number of known tasks increases, the relative weight of each mask decreases in Mask_{LC}. This could be a problem, particularly as the weight of the new random mask is reduced, possibly biasing the search excessively towards an average of previous policies. Therefore, we introduce a third approach that attempts to combine the benefits of both Mask_{RI} and Mask_{LC}: we set the initial weight of the new random mask to 0.5, while the remaining 0.5 weight is shared by the masks of all known tasks. We name this third approach *Balanced Linear Combination* (Mask_{BLC}). It must be noted that the difference between Mask_{LC} and Mask_{BLC} is only in the initialization of weights when a new task is encountered, where $\bar{\beta}_{k+1} = 1/(k+1)$ in Mask_{LC} and $\bar{\beta}_{k+1} = 0.5$ in Mask_{BLC}. However, the parameters $\bar{\beta}$ can be modified arbitrarily by backpropagation during

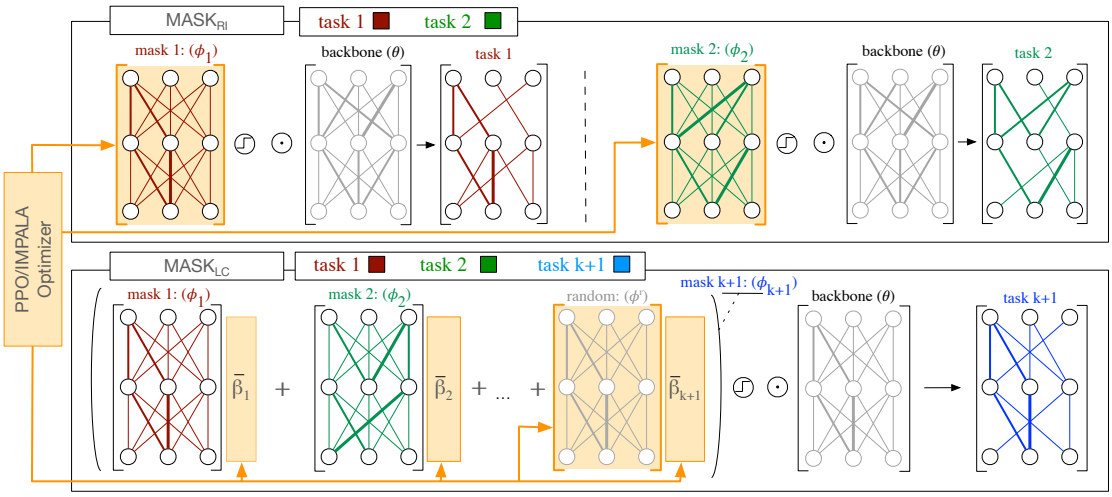


Figure 5.1: Graphical representations of the methods Mask_{RI} (top) and Mask_{LC} (bottom). Mask_{RI} searches for a new mask for each new task independently. Mask_{LC} attempts to exploit previous knowledge to learn a new task: known masks are linearly combined with a new randomly initialized mask while learning a new task. Gradient updates search for the parameters $\bar{\beta}_1, ..., \bar{\beta}_n$ and the new random mask.

training.

For both Mask_{LC} and Mask_{BLC}, updates are made only to S^l_{k+1} and $\bar{\beta}^l_i \dots \bar{\beta}^l_{k+1}$ across each layer l. After the training on task $k+1$ is completed, the knowledge from the linear combination is consolidated into the scores for the current task S^l_{k+1} by applying Equation 5.2. Therefore, the other masks and the linear combination parameters are no longer required. Algorithm 4 reports the forward pass operations for Mask_{LC}.

Algorithm 4 Forward pass in network layer l in Mask_{LC}

Require: Number of task learned so far k
Require: $S^*_1 \dots S^*_k$, S_{k+1}, $\bar{\beta}_1 \dots \bar{\beta}_{k+1}$, W
procedure FORWARDPASS($\mathbf{x}$)
  if $k \leftarrow 0$ **then**
    Set $P \leftarrow S_1$
  else
    Compute the task score from linear combination: $P \leftarrow \bar{\beta}_{k+1} S_{k+1} + \sum_{i=1}^{k} \bar{\beta}_i S^*_i$
  end if
  Compute mask from score $M_{k+1} \leftarrow g(P)$
  Modulate weight $W^{mod} \leftarrow (W \odot M_{k+1})$
  Compute output $\mathbf{y} \leftarrow W^{mod} \cdot \mathbf{x}$
  return y
end procedure

5.4.3 Continuous Modulatory Masks for Continuous RL problems

In previous studies, binarization of masks was discovered to be effective in classification to reduce the memory requirement and improve scalability. As shown in our simulations, this approach was effective also in discrete RL problems. However, in continuous action space RL environments, we discovered that binary masks did not lead to successful learning. It is possible that the quantization operation hurts the loss landscape in such environments since the input-output mapping is continuous to continuous values. Therefore, a modification of the supermask method can be devised to support the use of continuous value masks. In the thresholding mask generation operation (given a threshold ϵ), the modified version becomes

$$g(\phi_k) = \begin{cases} \phi_{k,\{i,j\}} & \phi_{k,\{i,j\}} > \epsilon \\ 0 & \text{otherwise} \end{cases} . \tag{5.3}$$

Such a modification still maintains the ability to learn sparse masks, but replaces the unitary positive values with continuous values. The results of the empirical investigation of the binary and continuous masks in a continuous action space environment is reported in Section 5.5.3.

5.5 Experiments

The three novel approaches, Mask_{RI}, Mask_{LC} and Mask_{BLC}, are tested on a set of LRL benchmarks and compared with a lifelong learning baseline, online EWC multi-head (EWC_{MH}), and with the non-lifelong learning algorithm PPO. Experiments with a PPO single task expert (STE) were also conducted to enable the computation of the forward transfer metric for each method, following the setup in [286]. Control experiments with EWC single head, denoted as EWC_{SH}, performed poorly as a confirmation that our benchmarks contain interfering tasks [127]: we report those results in the Section 5.9.6.

The benchmarks were chosen to assess the robustness of the method against the following aspects: discrete and continuous environments; variations across tasks in input, transition and reward distributions. The CT-graph [254] (sparse reward, fast, scalable to large search spaces, variation of reward), Minigrid [38] (variation of input distributions and reward functions) and Continual World [286] (continuous robot-control) were used to assess the approaches $\text{Mask}_{\text{RI/LC/BLC}} + \text{PPO}$. A complete set of hyper-parameters for each experiment is reported in Section 5.9.2.

Additionally, we employed the ProcGen benchmark [41] that consists of a set of video games with high diversity, fast computation, procedurally generated scenarios, visual recognition and motor control. The masking methods were implemented with IMPALA: $\text{Mask}_{\text{RI/LC/BLC}} + \text{IMPALA}$ [65] and tested following the

ProcGen lifelong RL curriculum presented in [205] (a subset of games in ProcGen). The properties of the benchmark make the curriculum challenging (e.g., high dimensional RGB observations and procedural generations of levels). The curriculum was designed to test generalization abilities: for each task, the lifelong evaluations are carried out using procedurally generated levels that are unseen by the agents during training. Baselines for this benchmark include online EWC, P&C [238], CLEAR [219], and IMPALA.

The metrics report a lifelong evaluation across all tasks at different points during the lifelong training, computed as the average sum of reward obtained across all tasks in the curriculum. The area under the curve (AUC) is reported in corresponding tables. A forward transfer metric, following the formulation employed in [286], is computed for the CT-graph, Minigrid and Continual World. For each task, the forward transfer is computed as the normalized difference between the AUC of the training plot for the lifelong learning agent and the AUC for the reference single task expert. For the ProcGen experiments, the lifelong training and evaluation plot format reported in [205] was followed to enable an easier comparison with the results in the original paper. As tasks are learned independently of other tasks in Mask_{RI}, there is no notion of forward transfer in the method. Therefore, the method is omitted when forward the transfer metrics are reported.

The results presented in the evaluation plots and the total evaluation metric reported in the tables below were computed as the mean of the 3 seed runs, with the error bars denoting the 95% confidence interval. While the sample size for the evaluation metric is 3, the sample size used for computing the mean and 95% confidence intervals for the forward transfer metric and the training plots is 3 seeds multiplied by the number of tasks per curriculum (i.e., 24, 36, 24, 30, 30 for *CT8*, *CT12*, *CT8 multi depth*, *MG10*, and *CW10* respectively). A significance test was conducted on the results obtained to validate the performance difference between algorithms. The result of the test is reported in Section 5.9.1.

5.5.1 CT-graph

The configurable tree graph (CT-graph) [254] is a sparse reward, discrete action space environment with configurable parameters that define the search space by setting the depth and breadth of a tree graph. Due to the exponential growth of the search space with the depth of the tree graph, this benchmark can be set up to the appropriate complexity to test the limits of RL algorithms. Each instance of the environment contains a start state followed by a number of states (2D patterned images) that lead to leaf states and rewards only with optimal policies. A task is defined by setting one of the leaf states as a desired goal state that can be reached only via one trajectory. Two experimental setups were employed:

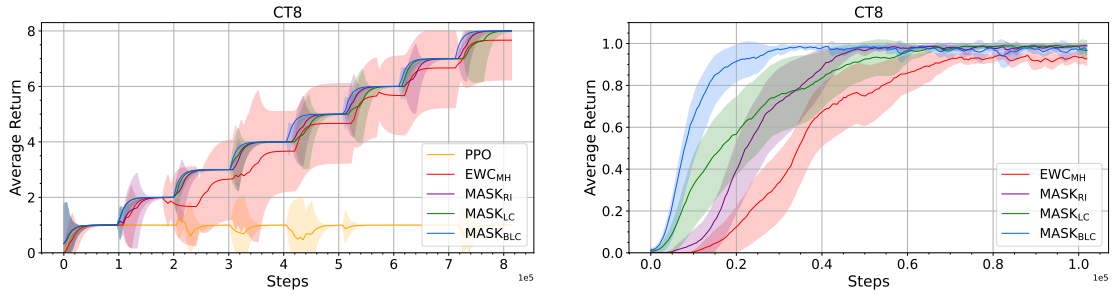


Figure 5.2: Performance in the *CT8* curriculum. (Left) Lifelong evaluation performance on all tasks (mean and 95% confidence interval on 3 seeds/samples). (Right) Training performance on each task, measured as the average return across all tasks and seeds runs (mean and 95% confidence interval on 8 tasks and 3 seeds, i.e., 24 samples). In the lifelong evaluation, a clear difference can be noted between the lifelong learning algorithms and the non-lifelong learning algorithm PPO, which is therefore not reported in the training plot (Right). The numerical values for the AUC are reported in Tables 5.1. The training curves show how Mask_{BLC} is on average faster at learning new tasks, followed by Mask_{LC} and Mask_{RI}. The performance on each individual task is reported in the Section in Figure 5.19

Method	Total Eval	Fwd Trnsf.
PPO	151.00 ± 8.61	0.19 ± 0.20
EWC_{MH}	646.33 ± 166.67	-0.15 ± 0.22
Mask_{RI}	699.33 ± 12.75	–
Mask_{LC}	699.33 ± 9.40	0.40 ± 0.19
Mask_{BLC}	717.33 ± 2.87	0.68 ± 0.07

Table 5.1: Total evaluation return (AUC of the lifelong evaluation plot in Figure 5.2(Left)) and forward transfer during lifelong training in the *CT8*. Mean ± 95% confidence interval reported.

the first with 8 leaf states (reward locations) that serves for 8 tasks, denoted as *CT8* curriculum (depth-3, breadth-2 graph with $2^3 = 8$ leaves). In the second setup, two graph types with 4 and 8 different reward locations, with depth 2 and 3 respectively, result in combined curriculum of 12 tasks, denoted as *CT12*. Such tasks have levels of similarities due to similar input distributions, but also interfere due to opposing reward functions and policies for the same inputs. Additionally, the 8-task graph has a longer path to the reward that introduces variations in both the transition and reward functions. Graphical illustrations of the graphs are provided in Section 5.9.4.

Each task is trained for 102.4K time steps. Figures 5.2 and 5.3 report evaluations in the *CT8* and *CT12* curricula as agents are sequentially trained across tasks. The forward transfer and the total evaluation performance metrics are presented in Tables 5.1 and 5.2 respectively.

The plots show that the masking methods (Mask_{RI}, Mask_{LC}, and Mask_{BLC}) are capable of avoiding forgetting and obtain high evaluation performance, with

Method	Total Eval	Fwd Trnsf.
PPO	374.00 ± 39.04	-0.10 ± 0.23
EWC_{MH}	1531.33 ± 45.56	-0.14 ± 0.21
Mask_{RI}	1533.67 ± 26.33	–
Mask_{LC}	1542.67 ± 3.79	0.49 ± 0.11
Mask_{BLC}	1557.33 ± 7.59	0.63 ± 0.07

Table 5.2: Total evaluation return (AUC of the lifelong evaluation plot in Figure 5.3(Left)) and forward transfer during lifelong training in the *CT12*. Mean ± 95% confidence interval reported.

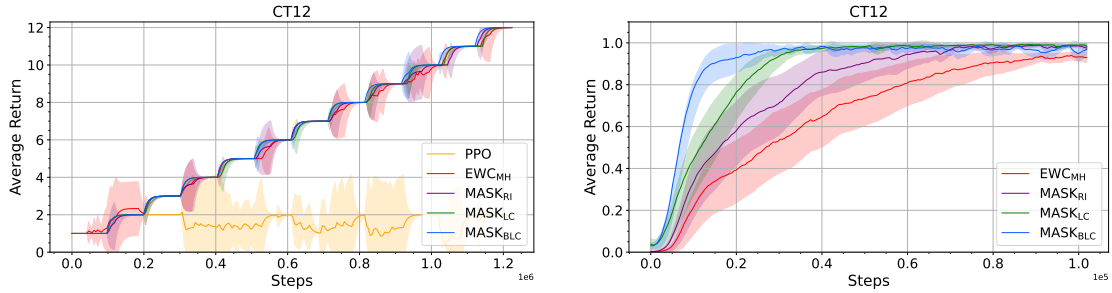


Figure 5.3: Performance in the *CT12* curriculum. (Left) Lifelong evaluation performance on all tasks (mean and 95% confidence interval on 3 seeds/samples). (Right) Training performance on each task, measured as the average return across all tasks and seeds runs (mean and 95% confidence interval on 12 tasks and 3 seeds, i.e., 36 samples).

significantly better forward transfer in comparison to EWC_{MH} and PPO. On average, the Mask_{LC} approach recovers performance faster than Mask_{RI}, and Mask_{BLC} performs best. An expanded version of the training plots showing the learning curves per tasks and averaged across seed runs is reported in the Section 5.9.6.

5.5.2 Minigrid

The Minigrid [38] is a discrete action space, sparse reward, grid-world navigation environment. The environment consists of a number of predefined partially observable tasks with varying levels of complexity. A consistent theme across all tasks is the requirement of an agent to navigate to a defined location in the grid-world. To achieve this, an agent may need to avoid obstacles such as walls, lava, moving balls, etc. The experiment protocol employs a curriculum of ten tasks (referred to as *MG10*), which consist of two variants of each of the following: SimpleCrossingS9N1, SimpleCrossingS9N2, SimpleCrossingS9N3, LavaCrossingS9N1, LavaCrossingS9N2. Screenshots of all tasks are reported in the Section (5.9.4). The variations across tasks include change in the state distribution and reward function. The results for the *MG10* experiments are presented in Figure 5.4 and Table 5.3. The masking methods obtained better performance in comparison to the baselines, with Mask_{BLC} obtaining the best performance. Section 5.9.6 provides a full experimental details.

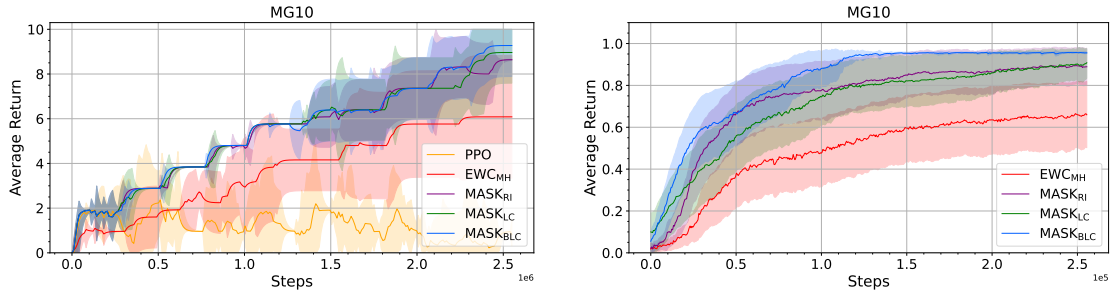


Figure 5.4: Performance in the *MG10* curriculum. (Left) Lifelong evaluation performance on all tasks (mean and 95% confidence interval on 3 seeds/samples). (Right) Training performance on each task, measured as the average return across all tasks and seeds runs (mean and 95% confidence interval on 10 tasks and 3 seeds, i.e., 30 samples).

Method	Total Eval	Fwd Trnsf.
PPO	296.85 ± 52.09	-0.40 ± 0.41
EWC_{MH}	933.56 ± 332.87	-1.09 ± 0.45
Mask_{RI}	1362.15 ± 119.14	–
Mask_{LC}	1360.80 ± 151.81	-0.19 ± 0.32
Mask_{BLC}	1388.09 ± 156.18	0.22 ± 0.22

Table 5.3: Total evaluation return (AUC of the lifelong evaluation plot in Figure 5.4(Left)) and forward transfer in the *MG10*. Mean ± 95% confidence interval reported.

5.5.3 Continual World

We evaluated the novel methods in a robotics environment with continuous action space, the Continual World [286]. The environment was adapted from the MetaWorld environment [299] that contains 50 classes of robotics manipulation tasks. The CW10 curriculum consists of 10 robotics tasks: visual screenshots are provided in the Section (5.9.4). The results for all methods were measured using the success rate metric introduced in [299], which awards a 1 if an agent solves a task or 0 otherwise. For the masking methods in this curriculum, the standard quantization of masks into binary performs poorly. To demonstrate this, two variants are run: the standard setting, where a binary mask is derived from the scores, denoted as $\text{Mask}_{\text{RI\_D}}$, and another where a continuous mask is derived from the scores (discussed in Section 5.4.3), denoted as $\text{Mask}_{\text{RI\_C}}$. The results from Figure 5.5 and Table 5.4 show that $\text{Mask}_{\text{RI\_C}}$ performs significantly better than $\text{Mask}_{\text{RI\_D}}$. Motivated by the results, the linear combination of masks method Mask_{LC} presented for this curriculum also employed the use of continuous masks. Mask_{LC} and Mask_{BLC} performed markedly better than the baseline EWC_{MH} that appears to struggle on this benchmark. Section 5.9.6 reports the details for all methods.

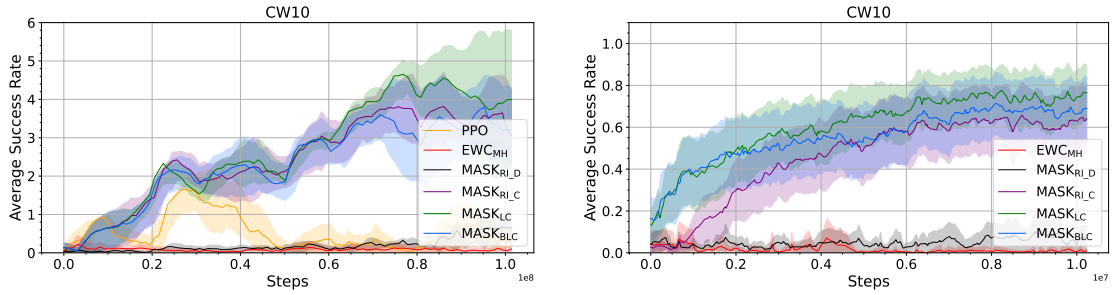


Figure 5.5: Performance in the *CW10* curriculum measured using the success rate metric. (Left) Lifelong evaluation performance on all tasks (mean and 95% confidence interval on 3 seeds/samples). (Right) Training performance on each task, measured as the average success rate across all tasks and seeds runs (mean and 95% confidence interval on 10 tasks and 3 seeds, i.e., 30 samples).

Method	Total Eval	Fwd Trnsf.
PPO	53.83 ± 72.32	-4.06 ± 2.58
EWC_{MH}	8.60 ± 12.91	-7.39 ± 3.76
$\text{Mask}_{\text{RI\_D}}$	20.45 ± 51.46	–
$\text{Mask}_{\text{RI\_C}}$	246.83 ± 157.39	–
Mask_{LC}	272.43 ± 124.92	-0.33 ± 0.36
Mask_{BLC}	237.00 ± 167.25	-0.61 ± 0.52

Table 5.4: Total evaluation success metric (AUC of the lifelong evaluation plot in Figure 5.5(Left)) and forward transfer in the *CW10*. Mean ± 95% confidence interval reported.

5.5.4 ProcGen

The ProcGen [41] is a discrete action, procedurally generated environment that contains a number of visually diverse video games. It was proposed as a replacement of the Atari games benchmark for lifelong RL, while being computationally faster to simulate than Atari. The procedural nature of the environment facilitates testing of lifelong RL agents in unseen environments, thus evaluating also generalization capabilities. Variation in tasks exists across the state and transition distributions.

The experimental protocol employed follows the setup of [205], with a sequence of six tasks ($0-$ Climber, $1-$ Dodgeball, $2-$ Ninja, $3-$ Starpilot, $4-$ Bigfish, $5-$ Fruitbot) with five learning cycles. Screenshots of the games are reported in the Section (5.9.4). Several environment instances, game levels, and variations in objects, texture maps, layout, enemies, etc., can be generated within a single task. For each task, agents are trained on 200 levels. However, the evaluation is carried out on the distribution of all levels, which is a combination of levels seen and unseen during training.

IMPALA was used as the base RL optimizer on which we deployed the novel masking methods. The results are reported in Figure 5.6, following the presenta-

tion style used in [205]. The masking methods (Mask_{RI}, Mask_{LC}, and Mask_{BLC})

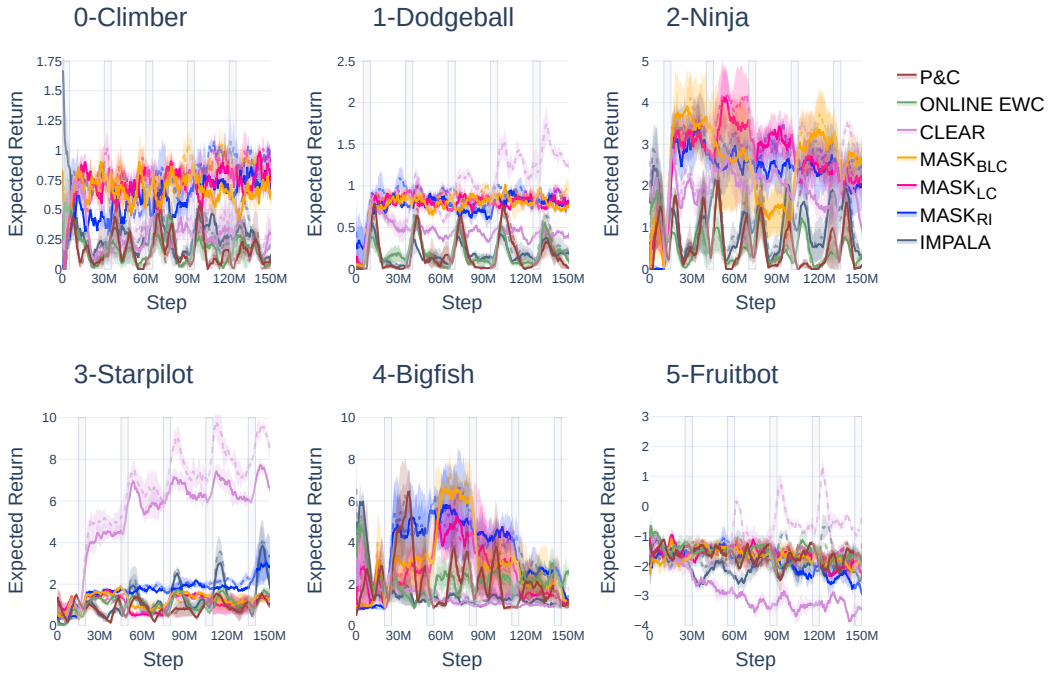


Figure 5.6: Evaluation results in the ProcGen environment (6 tasks, 5 cycles). The solid line represents evaluation on unseen environments, while the dotted line represents evaluation on the training environments. The gray shaded rectangles show at what point in time an agent is been trained on each task.

show better performance with respect to other baselines across most tasks, while maintaining generalization capabilities across training and evaluation environments. As the tasks are visually diverse, possibly resulting in less similarity across tasks, reusing previous knowledge may not offer much advantage. Nevertheless, the evaluation performance for each method reported in Table 5.5 illustrates a significant advantage of the masking methods with respect to the baselines, particularly in the test tasks, where Mask_{LC} is 44% better than the closest runner up (CLEAR) and over 300% better than P&C.

5.6 Analysis

The results of the previous section prompt the following questions: what linear coefficients emerge after learning? How is rapid learning in Mask_{LC} achieved? How is knowledge reused?

5.6.1 Coefficients for the linear combination of masks

To validate whether the proposed linear combination process can autonomously discover important masks that are useful for the current task, a visualization of the co-efficients after learning each task is presented.

Figure 5.7 presents the visualization of the coefficients (for the input and out-

Method	Evaluation Performance	
	Train Tasks	Test Tasks
IMPALA	3687.57 ± 1194.23	2818.44 ± 1214.34
Online EWC	4645.87 ± 1963.42	3831.94 ± 1258.38
P&C	3403.59 ± 3428.37	3390.97 ± 1795.58
CLEAR	11706.34 ± 5271.45	8447.64 ± 2286.21
Mask_{RI}	12462.82 ± 1057.68	12222.10 ± 6064.18
Mask_{LC}	12488.31 ± 3416.90	12377.77 ± 1440.80
Mask_{BLC}	11683.21 ± 3172.61	11913.48 ± 3869.95

Table 5.5: Total evaluation return (AUC of the lifelong evaluation plot in Figure 5.6) on the train and test tasks in ProcGen curriculum. Mean ± 95% confidence interval reported.

put layers) for a Mask_{LC} agent trained in the *CT8*, *MG10*, and *CW10* curricula respectively. In each plot, each row reports the final set of coefficients after training on a task. For example, the third row in each plot represents the third task in the curriculum and reports three coefficients used for the linear combination of the masks for two tasks and the new mask. For the first task (first row), there are no previous masks to combine.

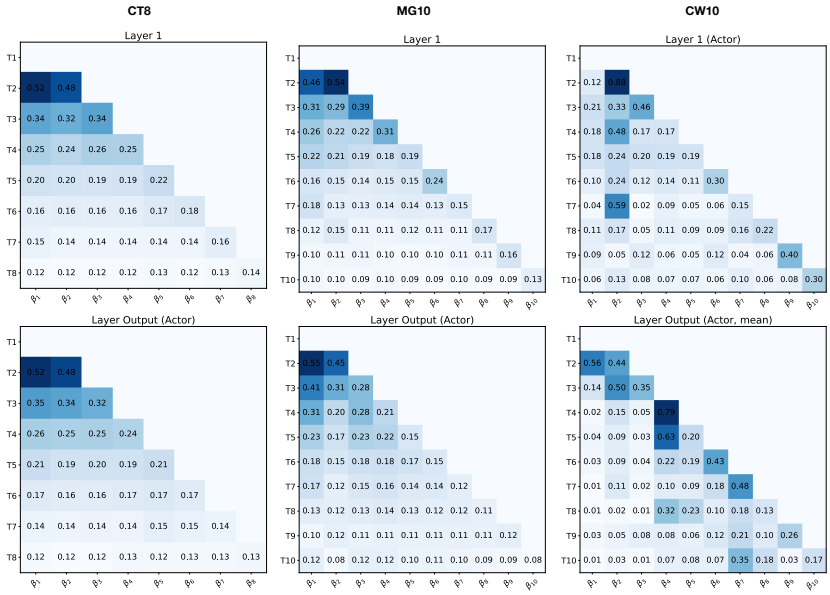


Figure 5.7: Coefficients $\bar{\beta}$ in Mask_{LC} after training on the *CT8*, *MG10*, and *CW10* curricula. Each row represents a task, and the values (which sums to 1) in it are the final set of coefficients after training on the task: the higher the value of a cell, the higher the level of importance of the corresponding mask in the linear combination operation. The figure only shows coefficients for the first layer and the output (actor) layer. See Section 5.9.5 for plots of all other layers.

For the *CT8* and *MG10*, the plots show that the coefficients have similar weights for each task (i.e., row wise in Figure 5.7). This observation is consistent

across the layers of the network (see Section 5.9.5 for plots across all layers). The uniform distribution across co-efficients means that the knowledge from all previous tasks is equally important and reused when learning new tasks, possibly indicating that tasks are equally diverse or similar to each other. The knowledge reuse of previous tasks thus accelerates learning and helps the agent quickly achieve optimal performance for the new task. For example, in the CT-graph curricula where navigation abilities are essential to reach the goal, the knowledge on how to navigate/traverse to different parts of the graph is encoded in each previously learned task. Rather than re-learn how to navigate the graph in each task, and subsequently the solve the task, the agent can leverage on the existing navigational knowledge. The performance improvement therefore comes from the fact the agent with knowledge reuse can leverage existing knowledge to learn new tasks quickly. Note that we will not expect any performance improvement if the new task bears no similarity with the previously learned task.

Comparing the values across layers (i.e., column wise in Figure 5.7), we note that there is little variation. In other words, the standard deviation of each vector $\bar{\beta}$ is low, as all values are similar. From such an observation, it follows that the vector $\bar{\beta}$ could be replaced by a scalar for these two benchmarks.

A different picture emerges from the analysis of the coefficients in the CW10 curriculum (Figure 5.7 right-most column). Here the coefficients appear to have a larger standard deviation both across masks and across layers. Particular values may suggest relationships between tasks. E.g., the input layer coefficient β_2^1 (from layer 1, mask 2) is high in task 4 and 7. Similar diverse patterns can be seen in the output layer. We speculate that tasks in CW10 are more diverse and the optimization process is enhancing specific coefficients to reuse specific skills. The non-uniformity of the co-efficients in the analysis could be a consequence of different levels of task similarities as reported in the forward transfer matrix in [286] for the CW10.

5.6.2 Exploitation of previous knowledge

The linear combination of masks appears to accelerate learning significantly as indicated in Figures 5.2(Right), 5.3(Right), 5.4(Right), and 5.5(Right). To investigate the causes of such learning dynamics, we plot the probabilities of actions during an episode in a new task. The idea is to observe what are the softmax probabilities that are provided by the linear combination of masks at the start of learning for a new task. A full analysis would require the unfeasible task of observing all trajectories: we therefore focus on the optimal trajectory by measuring probabilities when traversing such an optimal trajectory.

Figure 5.8 shows the analysis for the following cases: facing a 4th task after

learning 3 tasks in the *CT8* benchmark; facing the 12th task after learning 11 tasks in the *CT12* benchmark; facing the 7th task after learning 6 tasks in the *MG10* benchmark. The chosen task for each curriculum is set to test different instances of knowledge reuse. For *CT8*, the 4th task tests the agent's ability to reuse knowledge after learning a few similar tasks, while the 12th task in *CT12* investigates the agent's behaviour after learning many tasks. In *MG10*, the input distribution of the 7th task differs from the previous ones (i.e., from tasks with no lava to tasks with lava), thus testing the agent's ability to reuse previously learned navigation skills while dealing with new input scenarios. The result of the analysis is generalizable to other task changes in the curricula.

In the analysis, EWC_{MH} produced a purely random policy, with a uniform distribution over actions for each time step. Despite learning previous tasks, the new random head results in equal probabilities for all actions. On the contrary, both Mask_{LC} and Mask_{BLC} use previous masks to express preferences. In particular, in the *CT8* and *CT12*, the policy at steps 1, 3, 5 and 7 coincides with the optimal policy for the new task, likely providing an advantage. However, at steps 4 and 6 for the *CT8*, and 2, 4, and 6 for the *CT12*, Mask_{LC} has a markedly wrong policy: this is due to the fact that the new task has a different reward function and is therefore interfering. Due to the balanced combination of previous knowledge with a new mask, Mask_{BLC} seems to strike the right balance between trying known policies and exploring new ones. Such a balanced approach is also visible in the *MG10* task. Here, task 7 (see the Section 5.9.4) consists of avoiding the walls and the lava while proceeding ahead, then turning left and reaching the goal: most such skills are similar to those acquired in previously seen tasks, and therefore task 7 is learned starting from a policy that is close to being optimal.

If Mask_{LC} and Mask_{BLC} are capable of exploiting previous knowledge, it is natural to ask whether such knowledge can be exploited to learn increasingly more difficult tasks. The CT-graph [254] environment allows for increasing the complexity by increasing the depth of the graph. In particular, with a depth of 5, the benchmark results in a highly sparse reward environment with a probability of getting a reward with a random policy of only one in $3^{11} = 177,147$ episodes [250]. We therefore designed a curriculum, the *CT8 multi depth*, composed of a set of related but increasingly complex tasks with depth 2, 3, 4 and 5 (two tasks each depth with a different reward function).

Figure 5.9 shows the performance in the *CT8 multi depth* curriculum. EWC_{MH} was able to learn the first 4 tasks with depth 2 and 3, but failed to learn the last 4 more complex tasks. Interestingly, Mask_{BLC} managed to learn task 5 and 6 only partially. Mask_{LC} was able to learn all tasks, demonstrating that it could reuse previous knowledge to solve increasingly difficult tasks. Figure 5.10 presents the

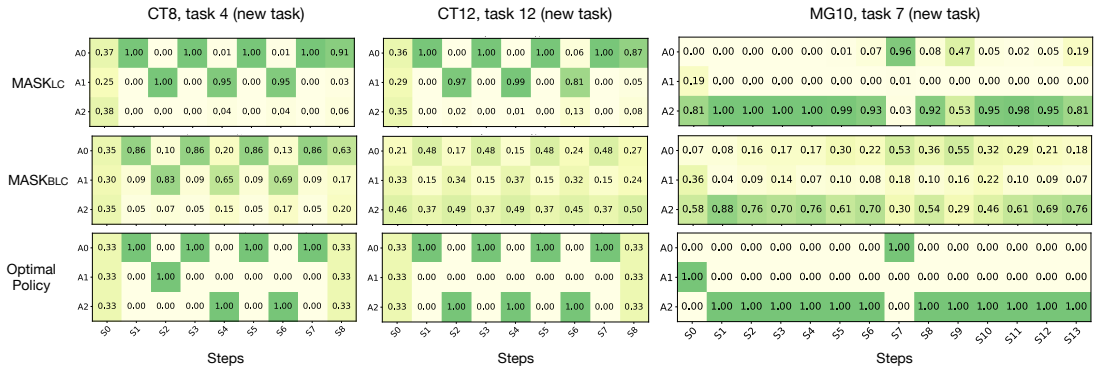


Figure 5.8: Knowledge reuse when learning a new tasks. The softmax output probabilities are shown during an episode with an unforeseen task (task 4 in *CT8*, task 12 in *CT12*, and task 7 in *MG10*) as the agent is guided through an optimal policy. From the top row down, Mask_{LC} displays biased probabilities representing an average behaviour across previous tasks. Mask_{BLC} shows biased probabilities but contains more randomness in comparison to Mask_{LC}. A visual comparison with the optimal policy (bottom row) suggests that both Mask_{LC} and Mask_{BLC} start learning unforeseen tasks with useful knowledge. Note, EWC_{MH} produced balanced probabilities (uniform distribution) for all actions at each time step, and is thus omitted from the figure.

training performance for each individual task, highlighting were most methods fail in the curriculum.

5.7 Discussion

The proposed approach employs learned modulatory masking methods to deliver lifelong learning performance in a range of reinforcement learning environments. The evaluation plots (the left panels of Figures 5.2, 5.3, 5.4 and 5.5) show how the performance increases as the agent learns through a given curriculum. The monotonic increase, indicating minimal forgetting, is most clear for the CT-graph and Minigrid, while the Continual World appears to have a more noisy perform-

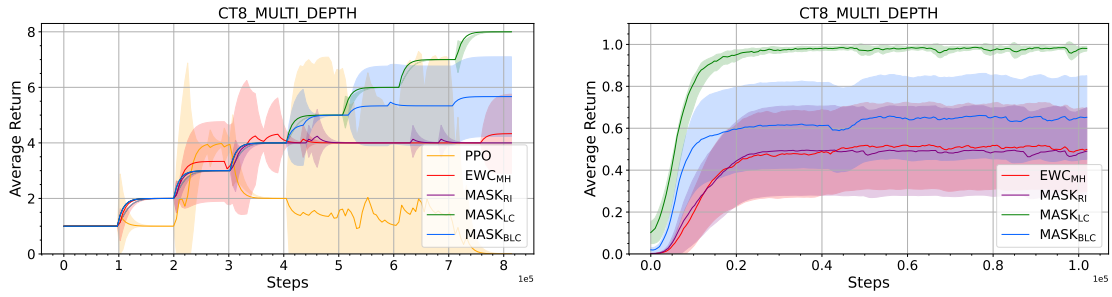


Figure 5.9: Performance in the *CT8 multi depth* curriculum. (Left) Lifelong evaluation performance on all tasks (mean and 95% confidence interval on 3 seeds/samples). (Right) Training performance on each task, measured as the average return across all tasks and seeds runs (mean and 95% confidence interval on 8 tasks and 3 seeds, i.e., 24 samples). Excluding MASK_{LC}, the training performance for other methods are sub-optimal, due to the failure to solve later tasks in the curriculum.

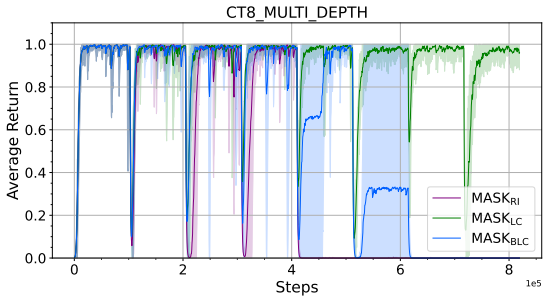


Figure 5.10: Training performance on each task, measured as the average return across all seeds runs (mean and 95% confidence interval on 3 seeds/samples), expanded from Figure 5.9(Right).

Method	Total Eval	Fwd Trnsf.
PPO	238.67 ± 283.86	0.10 ± 0.14
EWC_{MH}	530.33 ± 43.48	0.11 ± 0.12
Mask_{RI}	520.00 ± 2.48	–
Mask_{LC}	719.33 ± 1.43	0.77 ± 0.08
Mask_{BLC}	624.67 ± 63.83	0.44 ± 0.15

Table 5.6: Total evaluation return (AUC of the lifelong evaluation plot in Figure 5.9(Left)) and forward transfer during lifelong training in the *CT8 multi depth*. Mean ± 95% confidence interval reported.

ance. The EWC_{MH} baseline algorithm shows to be a highly performing baseline in the *CT8* and *CT12* curricula, it performs less well in the MG10 curriculum, and poorly in the *CW10* curriculum. The masking methods, instead, perform consistently in all benchmarks, with the linear combination methods, Mask_{LC} and Mask_{BLC}, showing some advantage over the random initialization Mask_{RI}. Evaluations on the ProcGen environments, while noisy to interpret from Figure 5.6, reveal that the masking methods outperform IMPALA, Online EWC, P&C and CLEAR by significant margins (Table 5.5).

While the learning dynamics of the core algorithm Mask_{RI} indicate superior performance to the baselines, we focused in particular on two extensions of the algorithm, Mask_{LC} and Mask_{BLC}. These use a linear combination of previously learned masks to search for the optimal policy in a new unforeseen task. These two variations combine previously learned masks with a new random mask to search for the optimal policy. One catch with this approach is that the performance on a new task will depend on which and how many tasks were previously learned. However, this is a property of all lifelong learning algorithms that leverage on previous knowledge. The balanced approach Mask_{BLC} starts with a 0.5 weight on the new random mask and can be seen as a blend between the random initialization Mask_{RI} and the linear combination Mask_{LC}. On average, it appears to achieve slightly better performance than either Mask_{RI} or Mask_{LC}. The standard linear

combination Mask_{LC} was the only algorithm able to learn the most difficult task on the CT-graph. This suggests that the algorithm is capable of exploiting previous knowledge to solve challenging RL problems. The fact that both Mask_{LC} and Mask_{BLC} have superior performance to the core random initialization Mask_{RI} validates the hypothesis that previous knowledge stored in masks can be reused.

The analysis of the coefficients of the linear combination (Section 5.6.1) reveals that the optimization can tune them to adapt to the nature of the curriculum. In the CT-graph and Minigrid curricula, previous masks are used in balanced proportions. On the Continual World environment, instead, particular masks, and layers within those masks, had significantly larger weights than others. From this observation, we conclude that the new proposed approach may be flexible enough to adapt to a variety of different curricula with different degrees of task similarity.

One concern with modulating masks is that memory requirements increase linearly with the number of tasks. This makes the approach not particularly scalable to large numbers of tasks. However, the promising performance of the linear combination approaches suggests that an upper limit could be imposed on the number of masks to be stored. After such a limit has been reached, new tasks can be learned solely as linear combinations of known masks, significantly reducing memory requirements. While this paper tested vector parameters with a scalar for each network layer, the analysis of the tuned parameters suggests that in some cases a single scalar for each mask could be used, further reducing memory requirements. Other suggestions to combat memory requirements in the mask approach are discussed in Section 5.8.

In the modulating masking setup, it is assumed that a task oracle exist that informs the lifelong RL agent which task it is presented at any given time (during training and evaluation/testing). This is made possible by providing a task identifier to the agent which it uses to select the correct mask to apply. A number of solutions could be combined with the mask method to eliminate this assumption in the framework. One approach could involve the use of *forget-me-not* Bayesian approach to task detection [183], as was employed in [132]. Another approach that could be explored is the use of optimal transport methods [5, 166] to measure distance of states/input across tasks. Also, the few-shots optimization of the linear superposition of masks via gradient descent that was employed in [287] for LSL could be employed to infer task mask in the LRL setup.

Given the fixed nature of the backbone network and the use of binary masks to sparsify the network, the representational capacity of the network could be affected. Masking approaches have been extensively studied in fixed neural networks [310, 210], including lifelong supervised learning setup [173, 287], with discussions

about representational capacity and generalization. While representation capacity could be affected, there are gains in generalization [312, 72] and robustness to noise [8] in sparse networks. In the future investigations, the gains in generalization could be useful to model-based RL approaches in lifelong learning.

The continuous masks in Equation 5.3 enabled the modulating masking agents to solve continuous action space problems that could not be solved by binary masks. While such feat could have stemmed from its improved representation capacity in comparison to binary masks, the gains were achieved at the cost of additional memory footprint. In addition to the discussions about the memory requirement reduction in masks, a thorough investigation into the trade-off between memory footprint and the representation capacity threshold required to solve a continuous action space problem could be explored in further studies.

5.8 Conclusion and Future Work

This work introduces the use of modulating masks for lifelong reinforcement learning problems. Variations of the algorithm are devised to ignore or exploit previous knowledge. All versions demonstrate the ability to learn on sequential curricula and show no catastrophic forgetting thanks to separate mask training. The analysis revealed that using previous masks to learn new tasks is beneficial as the linear combination of masks introduces knowledge in new policies. This finding promises potential new developments for compositional knowledge RL algorithms. Exploiting previous knowledge, one version of the algorithm was able to solve extremely difficult problems with reward probabilities as low as $5.6 \cdot 10^{-6}$ per episode simply using random exploration. These results suggest that modulating masks are a promising tool to expand the capabilities of lifelong reinforcement learning, in particular with the ability to exploit and compose previous knowledge to learn new tasks.

In the current study, the binarization process, key to reduce memory use, was successful in the discrete benchmarks and only after performing the linear combination in Mask_{LC} and Mask_{BLC}. Binarized masks resulted in poor performance in the continuous value Continual World benchmark, and if binarization was performed before the linear combination. Further studies could investigate this issue in more detail. It is possible that binarized masks could result in good performance if only the head layer was made continuous, thus ensuring smoothness in the output.

A different approach to reduce memory consumption is to take advantage of the apparent equal representations of known masks (as shown in Figure 5.7). If the advantage of previous knowledge can be represented as an average of previous masks, it is possible to modify the algorithm to maintain only a moving average

of all previous masks. In such a case, the algorithm will combine a new mask with the average of all previous masks. This extreme version of the algorithm is memory efficient, but may under-perform in curricula where coefficients are tuned to be diverse as in the CW10 benchmark (Figure 5.7 right-most column). Building on this idea, a limited number of *template* masks can be used instead of a single average mask. Each template can be a running average of a cluster of tasks, simply determined by L2 distances of masks, that will ensure good forward transfer while maintaining scalability.

One further approach to reduce memory is to experiment with high level of sparsity in the masks (Equations 5.1 and 5.3). Increasing the threshold (currently set to 0), or applying top k-winners as in [287] for supervised learning, may lead to a meta optimization process where significantly smaller masks maintain acceptable levels of performance. In summary, while more work is required to improve the memory efficiency of the proposed approaches, the success of the linear combination methods suggests venues of research to reduce memory consumption, while the performance advantages justify further research of masking methods in LRL.

Acknowledgement

This research contribution is based upon work supported by the United States Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA) under Contract No. FA8750-18-C-0103 (Lifelong Learning Machines) and Contract No. HR00112190132 (Shared Experience Lifelong Learning). Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the United States Air Force Research Laboratory (AFRL) and Defense Advanced Research Projects Agency (DARPA).

5.9 Supplementary Materials

5.9.1 Significance Testing

Results obtained from experiments in RL usually produce high variance across seeds [102]. This issue further leads to challenge of reproducible results. To address this concern, a difference test [42], using the Welch t-test and bootstrap confidence interval (BCI) were performed on the main results, evaluation performance and forward transfer. The tests were carried out at a significance level of 0.05. The BCI tests were run with 10,000 bootstrap iterations.

	CT8		CT12		CT8 MD	
Method	p-value	BCI	p-value	BCI	p-value	BCI
PPO	5.82e-9	[5.4e2, 5.52e2]	5.19e-5	[1.15e2, 1.17e2]	1.83e-2	[4.0e2, 6.1e2]
EWC_{MH}	3.05e-1	[-24.00, 95.67]	3.97e-1	[-2.67, 31.67]	2.82e-3	[171.67, 206.67]
MASK_{RI}	1.00e+0	[-5.67, 5.67]	2.78e-1	[0.00, 20.00]	3.18e-8	[198.33, 200.33]
MASK_{BLC}	9.45e-3	[-21.00, -13.67]	5.38e-3	[-18.00, -11.67]	2.36e-2	[71.00, 122.00]

Table 5.7: Total evaluation performance significance testing for the CT-graph curricula. Welch t-test (p-value) and bootstrap confidence interval significance difference testing at 5%, for $\mu_1 - \mu_2$, where μ_1 is the average total evaluation performance achieved by MASK_{LC} and μ_2 that of the comparisons (rows) in the table. Green coloured cells are statistically significant result where there is enough evidence to establish an order (difference) between μ_1 and μ_2, and vice versa cells are coloured red.

The outcome of the evaluation performance tests are reported in Tables 5.7, 5.8, 5.9, and 5.10, while the forward transfer tests are reported in Tables 5.11, 5.12, and 5.13. The MASK_{LC} was chosen as the method to compare against for the difference testing. The table cells coloured green signify that the test reported enough evidence to establish an order relationship between compared

	MG10	
Method	p-value	BCI
PPO	3.62e-04	[1009.20, 1128.32]
EWC_{MH}	1.79e-02	[299.21, 573.21]
MASK_{RI}	9.77e-01	[-70.74, 68.03]
MASK_{BLC}	6.18e-01	[-104.47, 49.88]

Table 5.8: Total evaluation performance significance testing for the Minigrid curriculum. Welch t-test (p-value) and bootstrap confidence interval (BCI) significance difference testing at 5%, for $\mu_1 - \mu_2$, where μ_1 is the average total evaluation performance achieved by MASK_{LC} and μ_2 that of the comparisons (rows) in the table. Green coloured cells are statistically significant result where there is enough evidence to establish an order (difference) between μ_1 and μ_2, and vice versa cells are coloured red.

Method	CW10 p-value	CW10 BCI
PPO	5.98e-03	[161.63, 268.27]
EWC_{MH}	1.12e-02	[206.87, 301.57]
$MASK_{RI\_D}$	1.18e-02	[195.02, 290.22]
$MASK_{RI\_C}$	6.14e-01	[-46.63, 96.80]
$MASK_{BLC}$	5.09e-01	[-41.97, 106.07]

Table 5.9: Total evaluation performance significance testing for the Continual World curriculum. Welch t-test (p-value) and bootstrap confidence interval (BCI) significance difference testing at 5%, for $\mu_1 - \mu_2$, where μ_1 is the average total evaluation performance achieved by $MASK_{LC}$ and μ_2 that of the comparisons (rows) in the table. Green coloured cells are statistically significant result where there is enough evidence to establish an order (difference) between μ_1 and μ_2, and vice versa cells are coloured red.

Method	Welch test p-value for $\mu_1 - \mu_2$ Train tasks	Welch test p-value for $\mu_1 - \mu_2$ Test tasks	Confidence Interval for $\mu_1 - \mu_2$ Train tasks	Confidence Interval for $\mu_1 - \mu_2$ Test tasks
IMPALA	4.09e-03	3.26e-05	[7.4e3, 10e3]	[8.9e3, 10.2e3]
Online EWC	2.63e-03	4.94e-05	[6.3e3, 9.2e3]	[7.9e3, 9.2e3]
P&C	1.28e-03	1.01e-04	[7.4e3, 10.8e3]	[8.2e3, 9.8e3]
CLEAR	6.25e-01	5.77e-03	[-1.5e3, 3.1e3]	[3.0e3, 4.9e3]
$MASK_{RI}$	9.78e-01	9.23e-01	[-1.5e3, 1.2e3]	[-1.8e3, 2.8e3]
$MASK_{BLC}$	4.99e-01	6.67e-01	[-0.87e3, 2.5e3]	[-0.85e3, 2.2e3]

Table 5.10: ProcGen total evaluation performance: Welch t-test and bootstrap confidence interval significance difference testing at 5%, for $\mu_1 - \mu_2$, where μ_1 is the average total evaluation performance achieved by $MASK_{LC}$ and μ_2 that of the comparisons (rows) in the table. Green coloured cells are statistically significant result where there is enough evidence to establish an order (difference) between μ_1 and μ_2, and vice versa cells are coloured red.

methods, and vice versa for cells coloured red. Also, when a positive only interval is reported in the BCI test, it signifies that $MASK_{LC}$ has a higher value than the method compared, and vice versa for negative only interval. The number of samples for the evaluation performance test is 3, the number of seed runs per method, while the number of samples for the forward transfer test is 3 multiplied by the number of tasks in each curriculum.

5.9.2 Hyper-parameters

In the experiments across the CT-graph, Minigrid and Continual World, all lifelong RL agents were built on top of the PPO algorithm. The hyper-parameters for the experiments are presented in Table 5.14. The EWC_{MH} and EWC_{SH} lifelong RL methods contain additional hyper-parameters which defines the weight preservation (consolidation) loss coefficient λ and the weight of the moving average α, for the online estimation of the fisher information matrix parameters following [36].

	CT8		CT12		CT8 MD	
Method	p-value	BCI	p-value	BCI	p-value	BCI
PPO	1.16e-01	[-0.03, 0.47]	1.57e-05	[0.34, 0.82]	1.71e-10	[0.53, 0.83]
EWC_{MH}	3.25e-04	[0.28, 0.82]	2.07e-06	[0.40, 0.86]	1.17e-11	[0.53, 0.80]
$MASK_{BLC}$	8.96e-03	[-0.45, -0.07]	2.71e-02	[-0.26, -0.02]	1.72e-04	[0.18, 0.49]

Table 5.11: Forward transfer significance testing for the CT-graph curricula. Welch t-test (p-value) and bootstrap confidence interval significance difference testing at 5%, for $\mu_1 - \mu_2$, where μ_1 is the average total evaluation performance achieved by $MASK_{LC}$ and μ_2 that of the comparisons (rows) in the table. Green coloured cells are statistically significant result where there is enough evidence to establish an order (difference) between μ_1 and μ_2, and vice versa cells are coloured red.

	MG10	
Method	p-value	BCI
PPO	4.13e-01	[-0.29, 0.67]
EWC_{MH}	1.80e-03	[0.34, 1.40]
$MASK_{BLC}$	3.67e-02	[-0.77, -0.05]

Table 5.12: Forward transfer significance testing for the Minigrid curriculum. Welch t-test (p-value) and bootstrap confidence interval (BCI) significance difference testing at 5%, for $\mu_1 - \mu_2$, where μ_1 is the average total evaluation performance achieved by $MASK_{LC}$ and μ_2 that of the comparisons (rows) in the table. Green coloured cells are statistically significant result where there is enough evidence to establish an order (difference) between μ_1 and μ_2, and vice versa cells are coloured red.

	CW10	
Method	p-value	BCI
PPO	6.37e-03	[0.94, 5.93]
EWC_{MH}	6.34e-04	[3.30, 10.29]
$MASK_{BLC}$	3.66e-01	[-0.34, 0.83]

Table 5.13: Forward transfer significance testing for the Continual World curriculum. Welch t-test (p-value) and bootstrap confidence interval (BCI) significance difference testing at 5%, for $\mu_1 - \mu_2$, where μ_1 is the average total evaluation performance achieved by $MASK_{LC}$ and μ_2 that of the comparisons (rows) in the table. Green coloured cells are statistically significant result where there is enough evidence to establish an order (difference) between μ_1 and μ_2, and vice versa cells are coloured red.

Hyper-parameter	CT8 / CT12 / CT8 MD	MG10	CW10
Learning rate	0.00015	0.00015	0.0005
Optimizer	RMSprop	RMSprop	Adam
Discount factor	0.99	0.99	0.99
Gradient clip	5	5	5
Entropy	0.1	0.1	0.005
GAE	0.99	0.99	0.97
Rollout length	128	128	5120
Num. of workers	4	4	1
PPO ratio clip	0.1	0.1	0.2
PPO optim. epochs	8	8	16
PPO optim. mini batch	64	64	160
Train steps per task	102,400	256,000	10.24M
Train iterations per task: $\frac{\text{trainsteps}}{\text{rollout}\times\text{workers}}$	200	500	2000
Eval. interval	10	20	200
Eval. episodes	10	10	10

Table 5.14: Hyper-parameters for curricula in the CT-graph (CT8, CT12, CT8 Multi Depth), Minigrid (MG10) and Continual World (CW10) curricula.

For Continual World, $\alpha = 0.75$ and $\lambda = 1 \times 10^4$, while for the CT-graph and Minigrid experiments, $\alpha = 0.5$ and $\lambda = 1 \times 10^2$.

Hyper-parameter	Value
Num. of workers	64
Batch size	32
Rollout length	20
Entropy	0.01
Learning rate	4×10^{-4}
Optimizer	RMSprop
Gradient clip	40
Discount factor	0.99
Num. of cycles (repeat curriculum)	5
Num. of train step per task per cycle	5M
Num. of eval episodes	10
Eval. interval	0.25M train steps

Table 5.15: Hyper-parameters for the curriculum in the ProcGen environment.

For the ProcGen experiments, the setup reported in [205] was followed, with each lifelong RL agent built on top of the IMPALA algorithm. The hyper-parameters for the baselines (IMPALA, Progress & Compress (P&C), ONLINE EWC, and CLEAR) were kept the same as in [205] for the experiments are presented in Table 5.15. ONLINE EWC contains additional hyper-parameter

Layer	Input units	Output units
Linear 1 (shared)	-	200
Linear 2 (shared)	200	200
Linear 3 (shared)	200	200
Linear (actor output)	200	3
Linear (value output)	200	1

Table 5.16: Specification of policy network across all methods for CT-graph and Minigrid curricula. Note, for multi head EWC network, the are multiple Linear (actor output) corresponding to the number of tasks.

Layer	Input units	Output units
Linear (actor body 1)	-	128
Linear (actor body 2)	128	128
Linear (actor output, mean)	128	3
Linear (actor output, log std)	128	3
Linear (value body 1)	-	128
Linear (value body 2)	128	128
Linear (value output)	128	1

Table 5.17: Network specification of policy network across all methods for Continual World curriculum. Note, for multi head EWC network, the are multiple Linear (actor output) corresponding to the number of tasks.

such as $\lambda = 175$ and replay_buffer_size $= 1 \times 10^6$. For P&C, $\lambda = 3000$, replay_buffer_size $= 1 \times 10^5$, and num_train_steps_of_progress $= 3906$. For CLEAR, replay_buffer_size $= 5 \times 10^6$.

5.9.3 Network Specifications

The policy network specification for the CT-graph (i.e., *CT8, CT12, and CT8 multi depth*) and Minigrid (i.e., *MG10*) curricula is presented in Table 5.16, with ReLU activation function employed. The output of the actor layer produces logits of a categorical distribution.

For the Continual World (i.e, *CW10*) curriculum, the policy network specification is presented in Table 5.17, with Tanh activation function employed. The output of the actor layer produces the mean and standard deviation of a gaussian distribution.The output of the standard deviation actor output layer is clipped within the range [-0.6931, 0.4055].

For the ProcGen environment, the input observation is an RGB image with shape $3 \times 64 \times 64$ and 15 discrete actions. ReLU activation was employed in the network. The policy specification for the network across all methods is presented in Table 5.18

Layer	Input	Output	Kernel	Stride	Pad
Conv1 (S)	3	32	[8, 8]	4	0
Conv2 (S)	32	64	[4, 4]	2	0
Conv3 (S)	64	64	[3, 3]	1	0
Flatten	$4 \times 4 \times 64$	1024	-	-	-
Linear1 (S)	1024	512	-	-	-
Linear (AO)	528	15	-	-	-
Linear (VO)	528	1	-	-	-

Table 5.18: Specification of policy network for the ProcGen experiments. Note, ConvX denotes convolutional layer x, with S, AO, and VO denoting shared layer (shared by actor/critic), actor output and value output. Input/Output refer to the number of input/output channels for Conv layers, and the number of input/output units for Linear layers. The number of input units for the actor and value output heads changes to 528 because the one-hot action vector (i.e., size 15) and reward scalar (i.e., size 1) from the previous time step is concatenated to the output of Linear 1.

Note that across all multi-head EWC experiments, the policy network contains multiple actor output layer corresponding to the number of tasks.

Backbone Network Initialization for Modulatory Masking Methods

Across all experiments, the weights of the backbone network for the modulatory masking methods were initialized using the signed Kaiming constant method, introduced in [210]. The constant $\pm c$ is the standard deviation of the Kaiming normal (distribution) initialization method, and could vary from layer to layer in the network. Furthermore, the bias parameters were disabled for the backbone networks in the masking methods, following the setup in [287].

5.9.4 Environments

CT-graph

The configurable tree graph (CT-graph) [254] is a sparse reward, discrete action space environment. The environment is represented as a graph, where each node is a state represented as a 12×12 gray scale image. There exist a number of state/node types in the environment, which are start (H), wait (W), decision (D), end/leaf (E), and fail (F) state. Each environment instance contains one home state, one fail states, and a number of wait, decision and end states. The goal of an agent is to navigate from the home state to one of the end states designated as the goal — the agent receives a reward of 1 when it enters the goal state, but 0 at every other time step. If the agent takes an incorrect action in any state, the agent transitions to the fail state, after which an environment reset takes it back to the home state.

The size and complexity (search space) of each environment (graph) instance is

determined by a set of configuration parameters — hence the term "configurable in the name". Two majors parameters in the CT-graph are the branch b and depth d that defines the branch (i.e., the width or number of decision actions at a decision state) and depth (i.e., the length) of the instantiated graph. The combination of the b and d determine how many end states exist in a each environment instance. Also, b determines the action space of an instance — defined as $b + 1$.

A task is defined by setting one of the leaf states as a desired goal state that can be reached only via one trajectory.

For the *CT8* curriculum, a graph instance with parameter $b = 2$ and $d = 3$ was employed — 2^3 end states. The 8 tasks comprise of each end state designated as the goal/reward location per task. For the *CT12* curriculum, two graph instances with 4 ($b = 2$ and $d = 2$) and 8 ($b = 2$ and $d = 2$) different end/reward states were combined. Additionally, the 8-task graph has a longer path to the reward that introduces variations in both the transition and reward functions. The *CT12* curriculum was based on an interleave of the tasks from both graph instances (i.e., task 1 in 4-tasks, task 1 in 8-tasks, task 2 in 4-tasks, task 2 in 8-tasks, task 3 in 4-tasks, and so on). See Figure 5.11 for a graphical representation of the 8-tasks and 4-tasks CT-graph. Lastly, the *CT8 multi depth* curriculum was composed of the first two end/goal states in each of the following graph instance: (i) $b = 2$ and $d = 2$, (ii) $b = 2$ and $d = 3$, (iii) $b = 2, d = 4$, (iv) $b = 2, d = 5$.

With a branching factor (breadth) b of 2 across all CT-graph curricula, the action space was defined as 3 (i.e., $b + 1$).

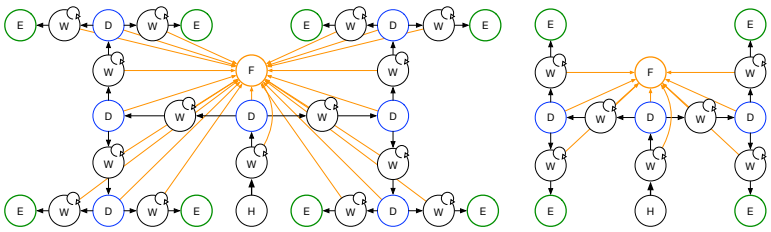


Figure 5.11: CT-graph environments. States are: home (H), wait (W), decision (D), end (E), fail (F). Three actions at W and D nodes determine the next state. (Left) CT8: a depth-3 graph with three sequential decision states (D). Reward probability $1/3^7 = 1/2187$ reward/episodes. (Right) A depth-2 graph with 4 leaf states (CT4) that combined with CT8 results in the CT12 curriculum.

Minigrid

Similar to the CT-graph, the Minigrid [38] is a sparse reward, discrete action navigation environment. The environment is setup as a grid world (with fast execution) where an agent is required to navigate to a goal location. It consist of a number of pre-defined grid worlds with several sub-variants defined by changing the random number generator seed. For all Minigrid experiments in this work, the

default grid encoding was employed, with each state represented using a tensor of shape $7 \times 7 \times 3$. The agent only get a reward slightly under 1 (depending on the number of steps taken as defined in Equation 5.4) when it arrives at the goal location, and a reward of 0 at every other state/time step:

$$goal\_reward = 1 - 0.9 \times \frac{es}{ms} \tag{5.4}$$

where es defines the number of steps taken to navigate to the goal (a green colour square), ms is the maximum number of steps the agent is allowed to take in an episode. For *MG10* curriculum, five pre-defined grid-worlds with two seed instances/variants (seed 860 and 861 was employed) per environment (hence 10 tasks) was employed. They are: SimpleCrossingS9N1, SimpleCrossingS9N2, SimpleCrossingS9N3, LavaCrossingS9N1, LavaCrossingS9N2. Figure 5.12 presents a visual illustration of the 5 grid worlds from which the tasks are derived. Note that when an agent steps a on lava (depicted in orange in the figure), the episode is terminated.

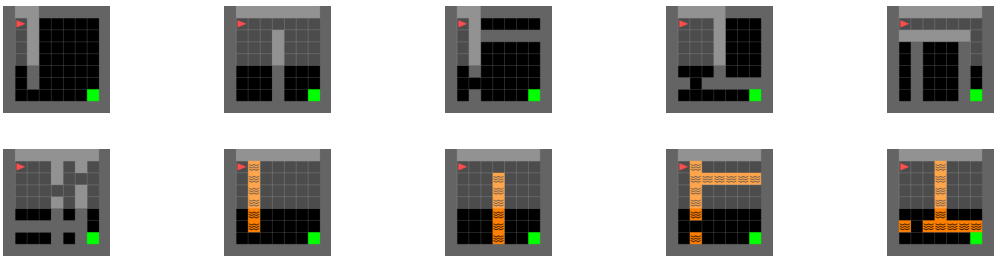


Figure 5.12: Visual representation of the 10 tasks in the *MG10* curriculum. From left to right, two variants of each class: SimpleCrossingS9N1, SimpleCrossingS9N2, SimpleCrossingS9N3, LavaCrossingS9N1, LavaCrossingS9N2.

Although the default action space in Minigrid is 7, the action space was set to 3 (turn left, turn right, and move forward) in this work as only navigational capabilities were required by the agents across all tasks in the *MG10* curriculum (i.e., other actions such as pick object, drop object, toggle and done actions were not necessary). Furthermore, the reduced action space eased the exploration demands across all methods when learning each task.

Continual World

The Continual World [286] is a benchmark for lifelong/continual RL derived from the Meta-World environment [299] — a benchmark consisting of 50 distinct simulated robotic tasks developed using the MuJoCo physics simulator [272]. The *CW10* curriculum in the benchmark comes from 10 tasks selected from the Meta World, with the goal of having a high variance in forward transfer across tasks. The 10 tasks (see Figure 5.13) are: hammer-v2, push-wall-v2, faucet-

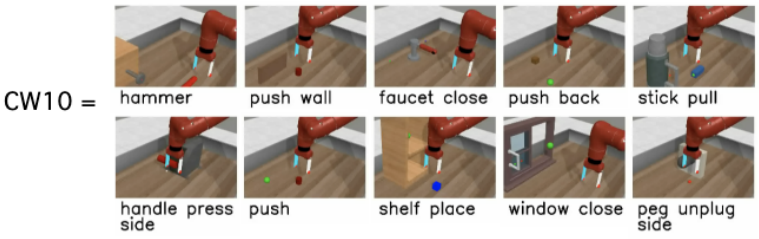


Figure 5.13: Visual representation of the 10 tasks *CW10* [286]

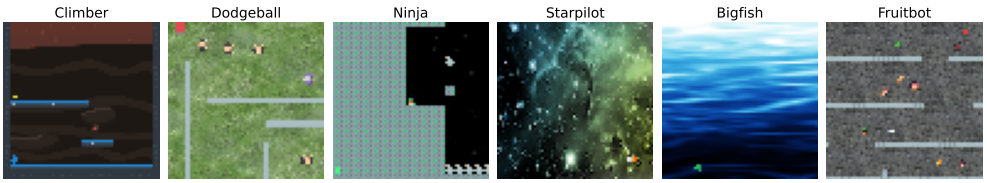


Figure 5.14: A snapshot of the tasks in the ProcGen curriculum. The texture, objects, RGB colour, structure are procedurally generated.

close-v2, push-back-v2, stick-pull-v2, handle-press-side-v2, push-v2, shelf- place-v2, window-close-v2, peg-unplug-side-v2. The input/state space of each task is a 39 dimension vector representation (consisting of proprioceptor information of the robotic arm as well as position of the objects and goal location in the environment), with an action space of 4 that defines the movement of the robotic arm. The reward function is defined based on a multi-component structure where the agent is reward for achieve sub-goals (i.e., reaching objects, gripping objects, and placing objects or a subset of these) within each task. In addition to the reward, another metric called *success metric* is used to measure performance — where the agent gets a 1 if it solves the overall task or 0 otherwise.

Note that when the Continual World benchmark was released, the authors used what is now termed version 1 (v1) environments in the Meta-World. However, the Meta-World v1 environments contained some issues in the reward function [1] which was fixed in the updated v2 environments. Therefore, the experiments in this investigation employed the use of the v2 environment for each task in the Continual World.

ProcGen

The ProcGen [41] is an benchmark that consists of 16 visual diverse video game tasks that are procedurally generated and computationally fast to run (in comparison to the Atari games), with the aim of evaluating generalization ability of RL agents. The benchmark was adapted for lifelong RL by [205] which introduced a lifelong RL curriculum based on a subset of the ProcGen games. The

[1] as discussed in https://github.com/rlworkgroup/metaworld/issues/226 and https://github.com/awarelab/continual_world/issues/2

selected games are Climber, Dodgeball, Ninja, Starpilot, Bigfish, and Fruitbot as shown in Figure 5.14. The input observation are RGB images of dimension $64 \times 64 \times 3$, along with 15 possible discrete actions. Also, the reward function and scales (range of values) are different for each task in the curriculum. Due to the procedural nature of the environment, each game contains several levels and the properties of each game instance (such as objects, texture maps, layout, enemies etc) can be procedurally generated, thus ensuring high variance within each game.

5.9.5 Additional Analysis

Modulatory Mask Similarities

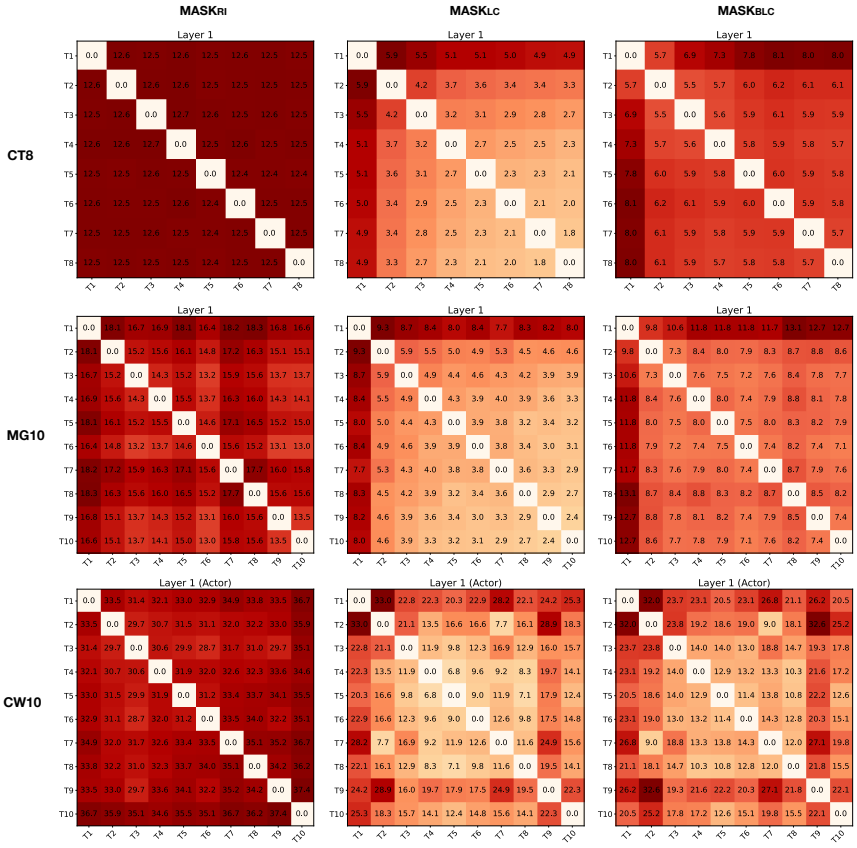


Figure 5.15: Pairwise mask distances between tasks (i.e., each plot computed as the L2 norm of the difference between task masks) in the first layer of the policy network for each modulatory masking method. Despite tasks having similarities in the CT-graph, Mingrid and Continual World curricula, in MASK_{RI} (Left), the learned masks across tasks show no correlation, MASK_{LC} (Middle) and MASK_{LC} (Right) show mask correlation across tasks (benefiting from knowledge re-use).

If similarities in tasks allow for our approach to exploit a linear combination of masks, it is reasonable to ask whether masks do reflect such similarity. We consider the two cases of: (1) Mask_{RI} where each mask is initialized randomly and (2) Mask_{LC} where each mask is a combination of a random mask and known masks. The analysis was conducted on masks learned in the *CT8* curriculum.

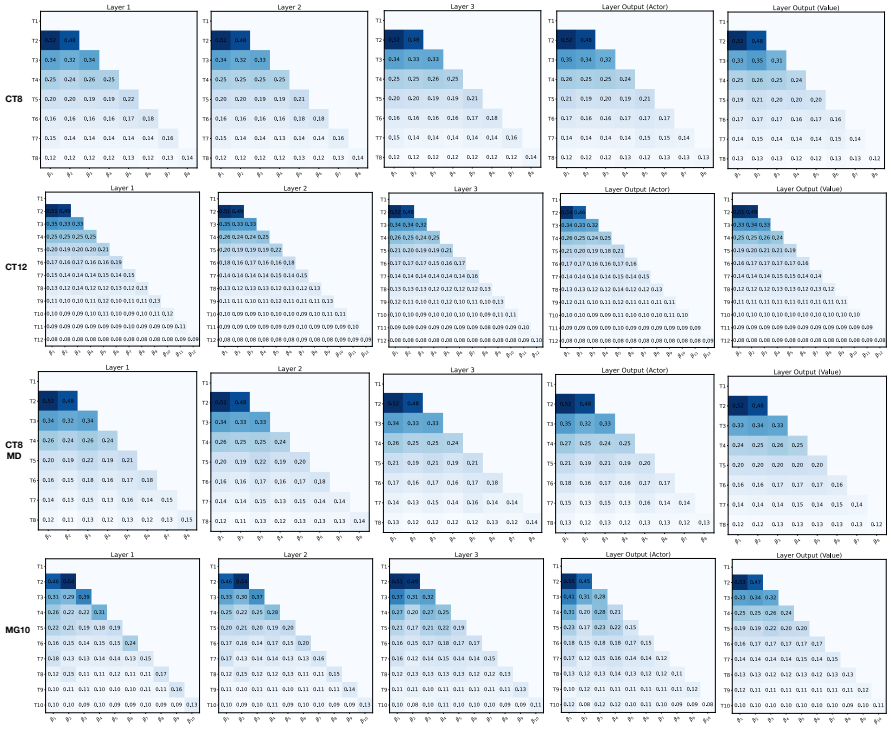


Figure 5.16: Per layer coefficients $\bar{\beta}$ in Mask_{LC} after training on the *CT8*, *CT12*, *CT8 multi depth*, and *MG10* curricula.

Figure 5.15 shows that, despite task similarities, random initialization of masks results in dissimilar masks. This result is expected as independent gradient optimizations will lead generally to different solutions. However, the linear combination of previously known masks is exploited in the tuning of new masks as we observed that the last two mask are significantly more similar to each other than the first two.

Linear Combination Coefficients

In Section 5.6.1, Figure 5.7 showed a summary of the linear combination coefficients of the input and output layers of the MASK_{LC} network after training. For completeness, this section presents the co-efficients for all layers in the network, across the *CT8*, *CT12*, *CT8 multi depth*, *MG10*, *CW10* curricula. The co-efficients are presented in Figures 5.16 and 5.17.

5.9.6 Additional Results

EWC Single versus Multi-Head Policy Network

As highlighted in Section 5.5, the results for the EWC lifelong RL agents presented were based on multi-head (multiple output layers) policy networks, while other methods employed a single head policy network. This is because the EWC single head network EWC_{SH} performed sub-optimally. In the CT-graph *CT8* curriculum, Figure 5.18 presents the continual evaluation comparison between the EWC_{SH} and

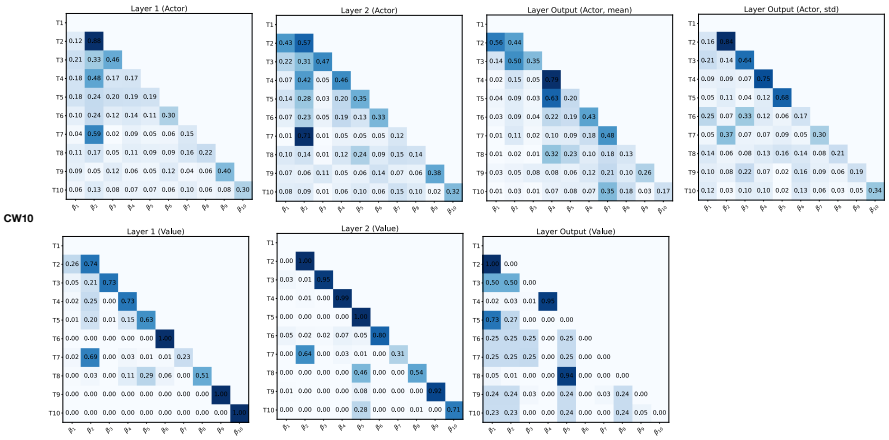


Figure 5.17: Per layer coefficients $\bar{\beta}$ in Mask_{LC} after training on the *CW10* curriculum.

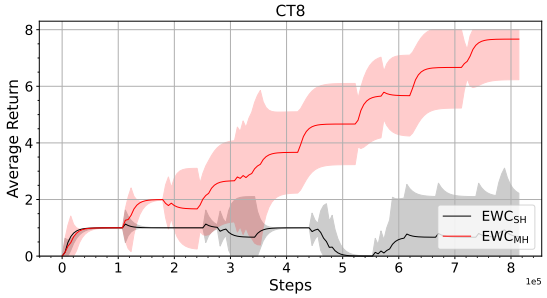


Figure 5.18: Continual evaluation comparison of EWC single head and multi-head policy networks in the *CT8* curriculum.

EWC_{MH}.

Train plot for all methods

In the lifelong training plots reported in Section 5.5, only the masking methods were presented for the sake of clarity and readability. The plots in Figure 5.19 present the lifelong training plots containing all methods across the CT-graph, Minigrid and Continual World curricula.

Per Task Forward Transfer

In the main text, the forward transfer metric was reported as the averaged across seed runs and tasks in the CT-graph, Minigrid and Continual World curricula. The reported information is expanded in this section to show the forward transfer metric per task (averaged across seed runs only), and reported in Tables 5.19, 5.20, 5.21, 5.22, and 5.23. The average across tasks is reported in the last column of each table. As noted in the main text, the tasks are learned independently of other tasks in Mask_{RI}, thus they are omitted in the tables.

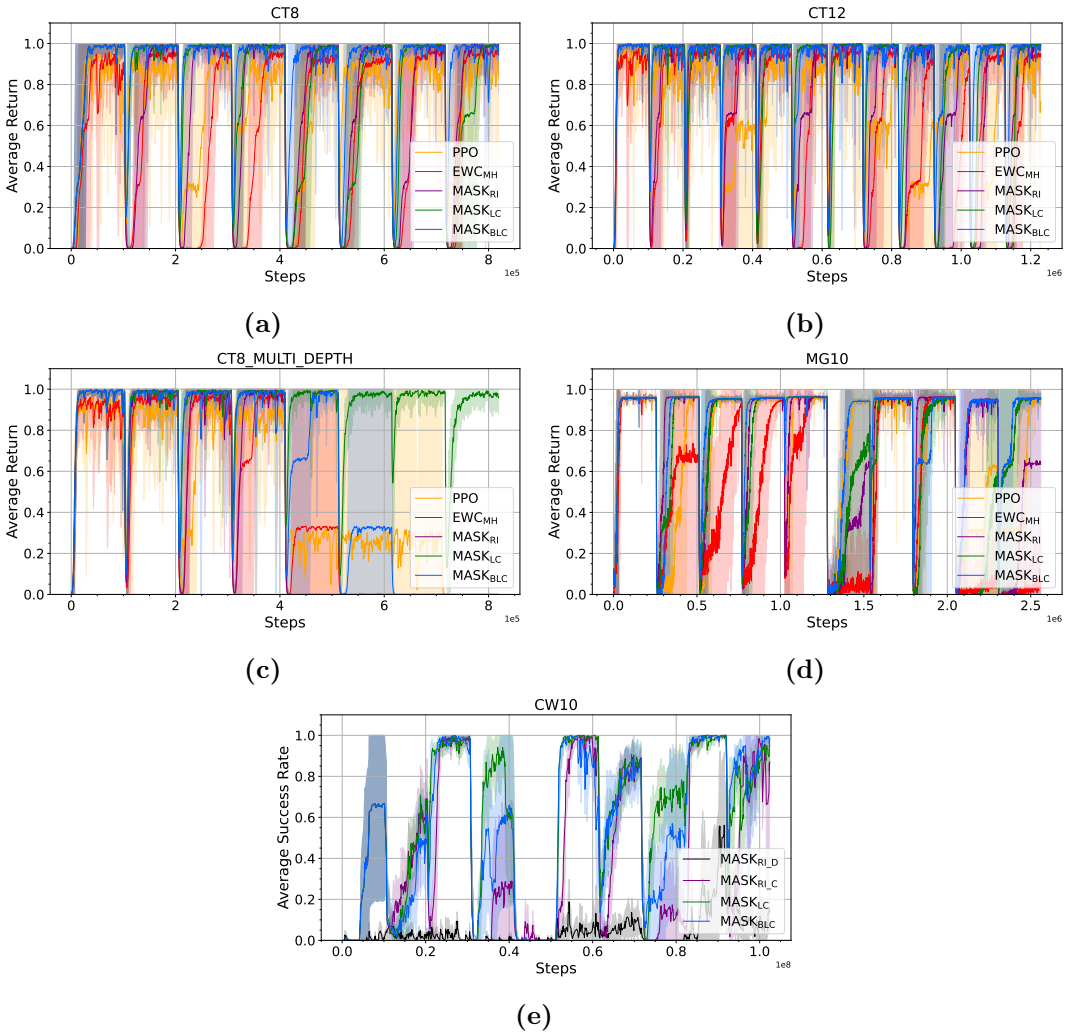


Figure 5.19: Lifelong training plots for all methods and baselines in CTgraph (CT), Minigrid (MG) and Continual World (CW) curricula: (a) *CT8*, (b) *CT12*, (c) *CT8 multi depth*, (d) *MG10*, and (e) *CW10*.

	Tasks								
Method	1	2	3	4	5	6	7	8	Avg
PPO	0.03	0.62	-0.50	0.33	-0.04	0.51	0.49	0.07	0.19
EWC_{MH}	-0.07	0.30	-1.17	-0.47	0.11	0.04	-0.09	0.17	-0.15
MASK_{LC}	0.38	0.82	0.69	0.66	0.15	0.14	0.57	-0.18	0.40
MASK_{BLC}	0.38	0.83	0.74	0.76	0.73	0.61	0.66	0.73	0.68

Table 5.19: Forward transfer per task in the *CT8* curriculum, averaged across seed runs.

	Tasks												
Method	1	2	3	4	5	6	7	8	9	10	11	12	Avg
PPO	-0.00	0.30	-0.08	-0.42	-0.38	0.01	-0.03	-0.29	-0.50	0.43	-0.72	0.48	-0.10
EWC_{MH}	-0.07	-0.01	0.06	0.34	-0.10	-0.79	0.30	-0.45	-0.14	-1.08	0.02	0.20	-0.14
$MASK_{LC}$	0.51	0.78	0.36	0.86	0.03	0.72	0.08	0.70	0.66	0.29	0.44	0.46	0.49
$MASK_{BLC}$	0.51	0.75	0.52	0.82	0.51	0.53	0.37	0.71	0.78	0.59	0.71	0.80	0.63

Table 5.20: Forward transfer per task in the *CT12* curriculum, averaged across seed runs.

	Tasks								
Method	1	2	3	4	5	6	7	8	Avg
PPO	-0.08	-0.27	-0.01	0.42	0.25	0.25	0.23	0.00	0.10
EWC_{MH}	0.01	0.04	0.31	0.27	0.28	0.00	0.00	0.00	0.11
$MASK_{LC}$	0.47	0.46	0.76	0.86	0.94	0.89	0.96	0.87	0.77
$MASK_{BLC}$	0.47	0.47	0.73	0.76	0.78	0.26	0.00	0.00	0.44

Table 5.21: Forward transfer per task in the *CT8 multi depth* curriculum, averaged across seed runs.

	Tasks										
Method	1	2	3	4	5	6	7	8	9	10	Avg
PPO	-0.02	-2.04	-0.23	0.47	-0.26	0.41	0.01	-0.69	-1.58	-0.11	-0.40
EWC_{MH}	-0.14	-2.11	-1.73	-1.10	-1.01	-0.47	-0.26	-0.44	-2.25	-1.37	-1.09
$MASK_{LC}$	0.05	-0.64	-0.03	0.04	0.65	-0.03	0.19	-0.83	-1.85	0.50	-0.19
$MASK_{BLC}$	0.05	-0.40	0.34	0.49	0.55	0.39	0.45	-0.52	0.22	0.58	0.22

Table 5.22: Forward transfer per task in the *MG10* curriculum, averaged across seed runs.

	Tasks										
Method	1	2	3	4	5	6	7	8	9	10	Avg
PPO	-0.87	-1.27	-1.06	0.25	-0.00	-13.79	-2.07	-1.42	-12.70	-7.68	-4.06
EWC_{MH}	-3.01	-1.44	-9.47	0.00	-0.00	-17.60	-2.22	-1.42	-30.07	-8.65	-7.39
$MASK_{LC}$	-1.91	-0.68	0.12	0.63	-0.00	0.54	-0.08	-0.10	-0.44	-1.36	-0.33
$MASK_{BLC}$	-1.91	-0.89	-0.23	0.40	-0.00	-1.84	-0.15	-0.63	-0.34	-0.51	-0.61

Table 5.23: Forward transfer per task in the *CW10* curriculum, averaged across seed runs.

ProcGen: Per Task Forward Transfer Metric

The per task forward transfer metric for all methods except MASK_{RI} in the ProcGen curriculum is presented in Table 5.24. Note that MASK_{RI} was omitted because the method does not inherently foster forward transfer as each task is learned independently of other tasks (i.e., for each task, a separate modulatory mask is independently initialized and optimized for the task).

	0-Climb..	1-Dodge..	2-Ninja	3-Starp..	4-Bigfi..	5-Fruit..	Avg
0-Climb..	–	–	–	–	–	–	–
1-Dodge..	-0.9	–	–	–	–	–	-0.9
2-Ninja	2.6	-3.3	–	–	–	–	-0.3
3-Starp..	-0.7	1.5	0.1	–	–	–	0.3
4-Bigfi..	1.7	-3.6	-1.3	-0.4	–	–	-0.9
5-Fruit..	3.1	-0.9	-1.0	0.5	-0.2	–	0.3
Avg	1.2	-1.6	-0.7	0.0	-0.2	–	-0.2

(a) IMPALA

	0-Climb..	1-Dodge..	2-Ninja	3-Starp..	4-Bigfi..	5-Fruit..	Avg
0-Climb..	–	–	–	–	–	–	–
1-Dodge..	0.0	–	–	–	–	–	0.0
2-Ninja	1.6	-2.2	–	–	–	–	-0.3
3-Starp..	-0.8	2.7	-0.2	–	–	–	0.6
4-Bigfi..	1.7	-3.4	0.1	-0.6	–	–	-0.5
5-Fruit..	0.6	-2.2	0.0	0.5	0.7	–	-0.1
Avg	0.6	-1.3	-0.0	-0.0	0.7	–	-0.1

(b) ONLINE EWC

	0-Climb..	1-Dodge..	2-Ninja	3-Starp..	4-Bigfi..	5-Fruit..	Avg
0-Climb..	–	–	–	–	–	–	–
1-Dodge..	0.0	–	–	–	–	–	0.0
2-Ninja	3.2	0.1	–	–	–	–	1.6
3-Starp..	-4.0	1.1	2.3	–	–	–	-0.2
4-Bigfi..	1.5	0.4	-0.2	0.3	–	–	0.5
5-Fruit..	-0.8	1.2	-1.9	0.3	0.9	–	-0.0
Avg	-0.0	0.7	0.1	0.3	0.9	–	0.3

(c) P&C

	0-Climb..	1-Dodge..	2-Ninja	3-Starp..	4-Bigfi..	5-Fruit..	Avg
0-Climb..	–	–	–	–	–	–	–
1-Dodge..	0.0	–	–	–	–	–	0.0
2-Ninja	0.5	-4.9	–	–	–	–	-2.2
3-Starp..	0.0	1.4	-0.6	–	–	–	0.3
4-Bigfi..	-0.8	-1.5	0.2	-0.2	–	–	-0.6
5-Fruit..	1.0	-1.1	-0.9	0.0	-0.4	–	-0.3
Avg	0.1	-1.5	-0.4	-0.1	-0.4	–	-0.5

(d) CLEAR

	0-Climb..	1-Dodge..	2-Ninja	3-Starp..	4-Bigfi..	5-Fruit..	Avg
0-Climb..	–	–	–	–	–	–	–
1-Dodge..	-0.8	–	–	–	–	–	-0.8
2-Ninja	2.1	-2.0	–	–	–	–	0.1
3-Starp..	-2.2	3.0	-4.8	–	–	–	-1.3
4-Bigfi..	2.3	-2.1	2.0	-2.3	–	–	0.0
5-Fruit..	-1.9	-0.9	2.6	-1.8	-0.8	–	-0.5
Avg	-0.1	-0.5	-0.0	-2.0	-0.8	–	-0.5

(e) MASK LC

	0-Climb..	1-Dodge..	2-Ninja	3-Starp..	4-Bigfi..	5-Fruit..	Avg
0-Climb..	–	–	–	–	–	–	–
1-Dodge..	-0.3	–	–	–	–	–	-0.3
2-Ninja	2.2	-2.1	–	–	–	–	0.0
3-Starp..	-1.4	4.4	-5.0	–	–	–	-0.6
4-Bigfi..	0.1	0.0	2.6	-2.8	–	–	-0.0
5-Fruit..	1.2	-0.0	1.6	-1.9	0.3	–	0.3
Avg	0.4	0.6	-0.2	-2.3	0.3	–	-0.1

(f) MASK BLC

Table 5.24: ProcGen transfer metrics.

Chapter 6

Conclusion

In order for the deployment of intelligent systems to gain more adoption in the real world, it is necessary to endow the systems with the ability to continually learn and adapt over the course of their lifetime. Several challenges plague the neural lifelong learner, such as efficient knowledge reuse, rapid task learning, adaptation struggle in diverse tasks, and so on. Several approaches exist to tackle these problems. Although existing solutions employ extra computation steps or storage requirement in addition to learning, they continue the use of the standard (classical) neural architecture that stemmed from scenarios where only one task is learned. The standard architecture of an ANN is composed of a sequence of connected layers of *homogeneous neurons*, where a neuron's output before non-linearity is the weighted sum neuronal output from the previous layer. However, biological neural networks contain heterogeneous neurons, where different types of neurons exist and with distinct functions. The modulatory neurons called *neuromodulators* are example of such heterogeneous neurons, and alter higher level cognitive behaviour by regulating other neurons. Loosely drawing inspiration from biology, neuromodulators introduced into ANNs produce networks with heterogeneous neurons. The inclusion of the neuromodulatory process in the neural architecture introduces emergent properties into the lifelong learning system, offering another vantage point from which to address the lifelong learning and adaptation challenges.

This thesis investigated and addressed specific learning and adaptation challenges facing the lifelong neural learners via the use of neuromodulators. The neuromodulators regulate the plasticity, weights or the neural activities in the network. The neuromodulated network consists of two neurons types, *standard* and *neuromodulatory*. The investigation showed that the incorporation neuromodulators into neural networks offered advantages in addressing diverse adaptation tasks, eliminating forgetting in lifelong RL, rapid task learning via knowledge re-use.

The contribution in Chapter 3 demonstrated that scaling and evolved neuromodulated Hebbian-based network to complex high dimensional partially ob-

servable environments offers an advantage in tackling the lifelong adaptation in such domains. The scaling was achieved through the use of an autoencoder network that mapped high dimensional input to latent features that was fed into the control network, where neuromodulation regulated the Hebbian-based plasticity. The system, characterized by learning across multiple time scales, demonstrated that knowledge encoded into the network by the slow evolutionary process is leveraged by the fast neuromodulated Hebbian-based plasticity to facilitate adaptation. From the empirical findings, knowledge encoded in the system included *reward neurons* that activate at the presence of reward cues, and *location neurons* that loosely act like biological place cells to help the system identify its location in partially observable environments.

In Chapter 4, a neuromodulatory computational framework was introduced and employed to augment the policy networks in meta-RL algorithms, tackling the lifelong adaptation problem across navigational and robotics domains. Neuromodulation regulated the neural activities in the policy network by flipping the activation sign. It was demonstrated that such network dynamics enabled the network to encode diverse input/output mappings required by the optimal policy for rapid task adaptation, where the task distribution contained increasingly complex tasks. Under this condition, the tasks required distinct neural activities (representations) in the optimal policy, and thus caused the standard networks without neuromodulation to struggle. Nevertheless, the policy network with neuromodulation was able to succeed in the tasks due to the effect of the neuromodulators in regulating and rapidly switching the neural activities. Furthermore, the experimental evidence validated this claim across both discrete and continuous action space environments, using two SoTA meta-RL algorithms.

In lifelong learning setting, Chapter 5 presented neuromodulation in the form of a mask that is used to gate the weights of a fixed backbone network. Through gating, the mask activates sub-regions in the network. A mask encodes knowledge for solving a specific task, and thus moves learning from the parameters of the network to the neuromodulators. Demonstrated in a lifelong RL setting, the system showcased an advantage in rapidly learning a new task when masks for previously learned tasks are linear combined with the new mask being learned for the task. The linear combination of masks addressed the lack of knowledge reuse (forward transfer) in existing masking methods, and it is beneficial in conditions where knowledge re-use is crucial to solve new complex tasks.

6.1 Research Contributions

The highlighted research contributions collectively addressed the research questions stated in Chapter 1 as discussed below.

6.1.1 Neuromodulated Hebbian-based Networks for Lifelong Adaptation

How can neuromodulated Hebbian-based networks be extended to solve lifelong adaptation problems in partially observable environments with complex high dimensional visual observations? In this contribution, it was shown that neuromodulated Hebbian networks can be extended to partially observable environment with high dimensional environments. This was made possible through the use of an autoencoder network that served as a feature extractor to produce latent features that fed into neuromodulated network. Consequently, overall system learned rapid adaptation skills from raw pixels in navigation domains. Despite having a separate objective and optimization approach from the neuromodulated Hebbian-based network (controller), the autoencoder was able to learn useful features that facilitated lifelong adaptation in the controller. This also demonstrated the robustness of the controller to change in the feature space.

What type of knowledge in such neuromodulated Hebbian-based networks is useful to facilitate rapid adaptation? During the training of the system, properties emerged in the evolved neuromodulated Hebbian-based network that was encoded in form of in-born knowledge. The network properties include the emergence of *location neurons* that helped the system identify its location in the partially observable environment, and *reward neurons* that helped to identify the presence of reward cues from raw pixels in the environment. Both in-born knowledge was then exploited by the neuromodulated Hebbian-based plasticity in the network to foster rapid adaptation. Through the identification of reward cues pixel, the system was able to explore when faced with a task change in order to rapidly discover the new optimal policy. Once the optimal policy is learned, the system then exploit continually the reward cues.

6.1.2 Neuromodulated Meta-RL for Lifelong Adaptation in Diverse Tasks

How do we design and develop a simple yet modular computational framework that can support several training setup The neuromodulated networks in the previous question focused on the development of evolved unstructured networks. A natural question arise about how the benefits of neuromodulation can be leveraged in structured multi-layer networks that can be optimized using any plasticity algorithm (i.e., gradient descent, neuroevolution, Hebbian-based plasticity and so on). This question was addressed through the development of a novel framework of a neuromodulated layer in PyTorch. The framework was designed to be modular, and can be plugged into any existing network architecture to incorporate the use of neuromodulation. Also, the framework can be extended with the inclusion

of new implementation or specification of the functional role of neuromodulation in the system (e.g., regulating neural activities, regulating network weights and so on).

What are the limitations of current methods as the task/problem complexity increases and tasks become increasingly dissimilar? In Chapter 4, this question was addressed. In scenarios where the tasks are similar, the policy networks without neuromodulation in meta-RL algorithms are able to achieve optimal adaptation performance. However, as the tasks become increasingly diverse and dissimilar, these networks struggle to achieve optimal performance. The struggle stemmed from the inability of the networks to capture diverse neural representations required by the optimal policy for each task. A few steps of fine tuning the policy is not sufficient to achieve optimal performance, and thus the goal of rapid task adaptation is not achieved.

What properties emerge from using neuromodulation to regulate neural activities in a meta-RL agent operating in complex task settings, and consequently does it lead to efficient adaptation in such settings? Building from the answer to the research question, the hypothesis of incorporating neuromodulation into the policy networks in meta-RL algorithms was investigated. The policy networks were augmented with neuromodulators, using modular architectural extension introduced. The neuromodulators employed targeted neural activities by regulating the numeric sign of the *standard neurons* in the network. This enabled the SoTA meta-RL algorithms to maintain rapid task adaptation as the task complexity increased. The regulation of neural activities enabled the networks to capture diverse neural representations with a few steps of fine tuning, thus facilitating the rapid task adaptation.

6.1.3 Neuromodulation in Lifelong Reinforcement Learning

How can the learned neuromodulatory masking method be extended to reinforcement learning domain for lifelong learning? The neuromodulatory masking framework was extended to the lifelong RL domain by reformulating the learning problem to satisfy the RL objective, using PPO and IMPALA RL algorithms. A mask is learned for each RL task, and it is applied on a fixed backbone network to select sub-region of the network that produces the optimal policy for the task. The mask is learned by randomly initialising mask parameters which are optimized by gradient descent.

What are the limitation and challenges of such extension to RL? While the use of neuromodulatory masks in lifelong RL was beneficial, the main limitation was the lack of the ability to reuse knowledge (forward transfer), since the mask for each task is learned independently. In RL, tasks share similarities in different

ways, either in the input/state space, reward function, transition function, or a combination of these. Therefore, it is beneficial for a lifelong learner to possess the ability to reuse knowledge.

What mechanisms are necessary for the incorporation of forward transfer property in neuromodulatory mask? The forward transfer property was introduced into the neuromodulatory masking method as a novel feature. In the investigation conducted, it was defined as the linear combination of masks from previously learned tasks and the new mask for the current task. The new mask is randomly initialised at the start of the task learning, and as such gives the agent an explorative ability, while the previous task masks enable the agent exploit existing knowledge. Also, the combination of masks is weighted by co-efficient parameters that are learnable by the agent. The learnable co-efficients enable the agent to self-discover which masks are useful for solving the current tasks, thus increasing the co-efficient value of those masks, while decreasing others.

Are there any tangible benefits gained from the combination of neuromodulatory masks to enable the forward transfer property in the learning system? The investigation carried to answer this question showed that there benefits in using the combination of masks to enable forward transfer (knowledge reuse). When learning a new task, the combination of masks facilitate the rapid task learning. The benefits are particularly evident in a scenario where the current task being learned is composed of knowledge encoded in previous task masks. Even when the task being learned is complex and cannot be solved from scratch, the combination of masks enable an agent to leverage on existing knowledge and discover the optimal policy. This benefit is however conditioned on the similarities that the complex task share with the previously learned task.

6.2 Limitations

Across all research contributions, the introduction of neuromodulation to the neural architecture of the lifelong learner facilitated rapid adaptation and learning in both partially and fully observable environments. The benefits are particular evident in settings where the tasks are increasingly complex or diverse, and share some similarities. In conditions where tasks are completely distinct and share no similarity, then the benefits could be absent and this opens a possible research direction in the future. Also, gains from the use of neuromodulators incurs an extra computation and storage cost as the networks that incorporates neuromodulation would contain more parameters than networks without.

In the first contribution, the system contained two neural components, an autoeconder feature extractor network and a neuromodulated Hebbian-based network. Each component had separate objectives being optimized. The multiple

neural components and multi-objective nature of the system could be seen as a bottleneck from the lens of robustness and generality, as distributional drift in the latent features of the autoencoder impacts the quality of decisions taken by the system downstream. Also, the lack of a single objective introduces difficulties with respect to how the system is setup and optimized. One solution would be to train the neural components one after the other, starting with the autoencoder. In such training setup, the system lack full online learning capabilities, and the input images required to train the autoencoder would need to be collected using another optimal policy or a random policy. In the investigation conducted for this thesis, both components were trained simultaneously, and the distributional drifts from the autoencoder were handled through using scaling and transformation operations.

For the second contribution, the policy network with neuromodulation contained more parameters than the policy network without. The gating mechanism by the neuromodulation bears similarity with gating in recurrent networks. The policy network with neuromodulation benefits from the neuromodulatory gating and the computational efficiency of feed forward network (faster training and better parallelization in comparison to recurrent networks). However, in task conditions where memory is required (e.g., POMDPs and sequential data processing), recurrent networks with gating would be advantageous over feed forward gated networks.

In the lifelong learning neuromodulatory framework, the main concern is the need to store masks for each task learned, thus the storage requirement grows linearly with tasks. This makes the approach not particularly scalable to large numbers of tasks. The binarization of mask parameters was key to reduce the memory footprint of the system, but the memory still grows linearly with tasks, albeit at a smaller scale. In the introduction of the linear combination of masks, real value mask parameters (instead of binary masks) are stored instead to enable the linear combination operation. Therefore, the benefits of knowledge reuse come at the cost of increased storage. However, the benefits gained in the system hints that an upper bound could be imposed on the number of stored masks. When the upper bound is reached, new tasks can be learned solely as linear combinations of known masks, significantly reducing memory requirements.

6.3 Future Work

The incorporation of neuromodulation in ANNs introduced heterogeneity into the neural architecture, which was beneficial for lifelong learning and adaptation. For the investigations conducted in this thesis, only one type of neuromodulator was employed per neural architecture. The demonstrated benefits show the promise of

neuromodulation and thus encourage the investigation of increased heterogeneity in neural architecture. For example, a neural system could be explored where two or more types of neuromodulators exist in a neural architecture. Each neuromodulator would perform different roles with the goal of encoding features beneficial for lifelong learning.

The architectural extension introduced in Chapter 4, showcased the use of neuromodulation in meta-RL algorithms using feed forward networks. A direction worth exploring is the use of plasticity gating neuromodulators in meta-RL algorithms that employs recurrent memory policy networks. This could the enhance the memory dynamics in the system. Studies from neuro-cognitive science already suggests that biological neuromodulators play a role in working memory and memory consolidation in the brain [11]. Therefore, future investigation could draw inspiration from dynamics and operations of such biological neuromodulators. From the perspective of lifelong learning and adaptation, such neuromodulators could help in memory consolidation, reconsolidation and selective forgetting in the neural systems.

While the empirical results showcasing the benefits of neuromodulation for plasticity regulation in Hebbian-based networks is useful, a detailed mathematical analysis of the formulation could be investigated in the future. Insights from such study could help provide better understanding of the effect of the neuromodulator on the overall behaviour of the system during lifelong adaptation and other related problems.

In the neuromodulatory framework containing the linear combination of masks for knowledge reuse, several solutions could be explored to combat the linear growth in memory requirement as the number of tasks increase. One approach could be to impose an upper bound on the number of masks that can be learned and stored. When the limit has been reached, then new tasks could be learned based on linear combination of known masks. Therefore, only the learned co-efficient parameters are stored for each subsequent task. For each mask in the linear combination operation, the investigation employed the use of one co-efficient value per layer in the network. One additional way to further impose restrictions on memory could be the use of a single scalar value for each mask in the linear combination operation across all network layers.

A different way to approach the reduction in memory usage in the linear combination of masks could be the exploitation of equal representations of known masks. If the advantage of previous knowledge can be represented as an average of previous masks, it is possible to modify the algorithm to store only a moving average of all previous masks. Therefore, the algorithm will combine a new mask with the average of all previous masks. Leveraging on this idea, a limited number

of *template* masks can be used instead of a single average mask. Each template could be a running average of a cluster of tasks, simply determined by L2 distances of masks, that will ensure good forward transfer while maintaining scalability.

Finally, another direction worth exploring in the linear combination of masks is this the identification of task specific and general knowledge in the system. This would enable the encoding of general knowledge into masks separate from task specific ones. The combination of masks could then be performed on the general knowledge masks, thus accurately eliminating unsuitable or potentially adversarial knowledge from the combination operation.

6.4 Broader Impact Statement

Although the current AI systems do not offer a human-level general purpose intelligence, it is important to consider the implications of the use of such systems as they gain wide spread adoption.

The research outcomes discussed in this thesis advance the development of intelligent systems that can learn multiple tasks over a lifetime. Such a system has the potential for real-world deployment, especially in robotics applications and the automation of manufacturing systems. Demand for human labour is likely to be impacted in some industries due to the deployment of autonomous systems. It is also important to consider how to ensure a smooth human-machine collaboration in the work force to improve productivity. A detailed risk assessment should be conducted and procedures should be put in place to mitigate against human injuries that could stem from human-machine collaboration.

In order to seek economic prosperity and benefit for all, a robust ethical framework should be designed to ensure that when autonomous systems are deployed, biases are eliminated or considerably reduced. The use of such systems should be fair, equitable and transparent. Legislators across the world should seek to understand the use of intelligent systems technology and design of policies to regulate the use of the technology as it gains wide spread adoption across industries and government. Military applications should be painstakingly reviewed and discouraged in situations where it can be avoided.

Bibliography

[1] Karim Abbasi, Parvin Razzaghi, Antti Poso, Saber Ghanbari-Ara, and Ali Masoudi-Nejad. Deep learning in drug target interaction prediction: current and future perspectives. *Current Medicinal Chemistry*, 28(11):2100–2113, 2021.

[2] LF Abbott and Wade G Regehr. Synaptic computation. *Nature*, 431(7010):796–803, 2004.

[3] Milad Taleby Ahvanooey, Qianmu Li, Ming Wu, and Shuo Wang. A survey of genetic programming and its applications. *KSII Transactions on Internet and Information Systems (TIIS)*, 13(4):1765–1794, 2019.

[4] Rahaf Aljundi, Francesca Babiloni, Mohamed Elhoseiny, Marcus Rohrbach, and Tinne Tuytelaars. Memory aware synapses: Learning what (not) to forget. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 139–154, 2018.

[5] David Alvarez-Melis and Nicolo Fusi. Geometric dataset distances via optimal transport. *Advances in Neural Information Processing Systems*, 33:21428–21439, 2020.

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

[7] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. *Advances in neural information processing systems*, 29, 2016.

[8] Sanjeev Arora, Rong Ge, Behnam Neyshabur, and Yi Zhang. Stronger generalization bounds for deep nets via a compression approach. In *International Conference on Machine Learning*, pages 254–263. PMLR, 2018.

[9] Derrik E Asher, Alexis B Craig, Andrew Zaldivar, Alyssa A Brewer, and Jeffrey L Krichmar. A dynamic, embodied paradigm to investigate the role

of serotonin in decision-making. *Frontiers in Integrative Neuroscience*, 7:78, 2013.

[10] Kenneth Atz, Francesca Grisoni, and Gisbert Schneider. Geometric deep learning on molecular representations. *Nature Machine Intelligence*, 3(12):1023–1032, 2021.

[11] Michael C Avery and Jeffrey L Krichmar. Neuromodulatory systems and their interactions: a review of models, theories, and experiments. *Frontiers in neural circuits*, 11:108, 2017.

[12] Kamyar Azizzadenesheli, Brandon Yang, Weitang Liu, Zachary C Lipton, and Animashree Anandkumar. Surprising negative results for generative adversarial tree search. *arXiv preprint arXiv:1806.05780*, 2018.

[13] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[14] Mohammad Babaeizadeh, Iuri Frosio, Stephen Tyree, Jason Clemons, and Jan Kautz. Reinforcement learning through asynchronous advantage actor-critic on a gpu. In *International Conference on Learning Representations*, 2017.

[15] Thomas Back, Ulrich Hammel, and H-P Schwefel. Evolutionary computation: Comments on the history and current state. *IEEE transactions on Evolutionary Computation*, 1(1):3–17, 1997.

[16] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[17] Yanis Bahroun, Eugénie Hunsicker, and Andrea Soltoggio. Building efficient deep hebbian networks for image classification tasks. In *Artificial Neural Networks and Machine Learning–ICANN 2017: 26th International Conference on Artificial Neural Networks, Alghero, Italy, September 11-14, 2017, Proceedings, Part I 26*, pages 364–372. Springer, 2017.

[18] Yanis Bahroun and Andrea Soltoggio. Online representation learning with single and multi-layer hebbian networks for image classification. In *Artificial Neural Networks and Machine Learning–ICANN 2017: 26th International Conference on Artificial Neural Networks, Alghero, Italy, September 11-14, 2017, Proceedings, Part I 26*, pages 354–363. Springer, 2017.

[19] Sungyong Baik, Janghoon Choi, Heewon Kim, Dohee Cho, Jaesik Min, and Kyoung Mu Lee. Meta-learning with task-adaptive loss function for few-shot learning. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9465–9474, 2021.

[20] Megan M. Baker, Alexander New, Mario Aguilar-Simon, Ziad Al-Halah, Sébastien M.R. Arnold, Ese Ben-Iwhiwhu, Andrew P. Brna, Ethan Brooks, Ryan C. Brown, Zachary Daniels, Anurag Daram, Fabien Delattre, Ryan Dellana, Eric Eaton, Haotian Fu, Kristen Grauman, Jesse Hostetler, Shariq Iqbal, David Kent, Nicholas Ketz, Soheil Kolouri, George Konidaris, Dhireesha Kudithipudi, Erik Learned-Miller, Seungwon Lee, Michael L. Littman, Sandeep Madireddy, Jorge A. Mendez, Eric Q. Nguyen, Christine Piatko, Praveen K. Pilly, Aswin Raghavan, Abrar Rahman, Santhosh Kumar Ramakrishnan, Neale Ratzlaff, Andrea Soltoggio, Peter Stone, Indranil Sur, Zhipeng Tang, Saket Tiwari, Kyle Vedder, Felix Wang, Zifan Xu, Angel Yanguas-Gil, Harel Yedidsion, Shangqun Yu, and Gautam K. Vallabha. A domain-agnostic approach for characterization of lifelong learning systems. *Neural Networks*, 2023.

[21] DAVID L BARKER, PERRY B MOLINOFF, and EDWARD A KRAVITZ. Octopamine in the lobster nervous system. *Nature New Biology*, 236(63):61–63, 1972.

[22] Mark Bear, Barry Connors, and Michael A Paradiso. *Neuroscience: Exploring the brain*. Jones & Bartlett Learning, LLC, 2020.

[23] Shawn Beaulieu, Lapo Frati, Thomas Miconi, Joel Lehman, Kenneth O Stanley, Jeff Clune, and Nick Cheney. Learning to continually learn. *arXiv preprint arXiv:2002.09571*, 2020.

[24] Sarah Bechtle, Artem Molchanov, Yevgen Chebotar, Edward Grefenstette, Ludovic Righetti, Gaurav Sukhatme, and Franziska Meier. Meta learning via learned loss. In *2020 25th International Conference on Pattern Recognition (ICPR)*, pages 4161–4168. IEEE, 2021.

[25] Eseoghene Ben-Iwhiwhu, Jeffery Dick, Nicholas A Ketz, Praveen K Pilly, and Andrea Soltoggio. Context meta-reinforcement learning via neuromodulation. *Neural Networks*, 152:70–79, 2022.

[26] Eseoghene Ben-Iwhiwhu, Pawel Ladosz, Jeffery Dick, Wen-Hua Chen, Praveen Pilly, and Andrea Soltoggio. Evolving inborn knowledge for fast adaptation in dynamic POMDP problems. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 280–288, 2020.

[27] Samy Bengio, Yoshua Bengio, Jocelyn Cloutier, and Jan Gecsei. On the optimization of a synaptic learning rule. In *Preprints Conf. Optimality in Artificial and Biological Neural Networks*, pages 6–8. Univ. of Texas, 1992.

[28] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432*, 2013.

[29] Marcus K Benna and Stefano Fusi. Computational principles of synaptic memory consolidation. *Nature neuroscience*, 19(12):1697–1706, 2016.

[30] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Dębiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. Dota 2 with large scale deep reinforcement learning. *arXiv preprint arXiv:1912.06680*, 2019.

[31] Jesper Blynel and Dario Floreano. Levels of dynamics and adaptive behavior in evolutionary neural controllers. In *Proceedings of the seventh international conference on simulation of adaptive behavior on From animals to animats*, pages 272–281. MIT Press, 2002.

[32] Hans J Bremermann et al. Optimization through evolution and recombination. *Self-organizing systems*, 93:106, 1962.

[33] Sahan Bulathwela, María Pérez-Ortiz, Aldo Lipani, Emine Yilmaz, and John Shawe-Taylor. Predicting engagement in video lectures. *arXiv preprint arXiv:2006.00592*, 2020.

[34] Sahan Bulathwela, María Pérez-Ortiz, Emine Yilmaz, and John Shawe-Taylor. Towards an integrative educational recommender for lifelong learners (student abstract). In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 13759–13760, 2020.

[35] Rich Caruana. Multitask learning. *Machine Learning*, 28:41–75, 1997.

[36] Arslan Chaudhry, Puneet K Dokania, Thalaiyasingam Ajanthan, and Philip HS Torr. Riemannian walk for incremental learning: Understanding forgetting and intransigence. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 532–547, 2018.

[37] Arslan Chaudhry, Marc'Aurelio Ranzato, Marcus Rohrbach, and Mohamed Elhoseiny. Efficient lifelong learning with a-GEM. In *International Conference on Learning Representations*, 2019.

[38] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. Minimalistic gridworld environment for openai gym. https://github.com/maximecb/gym-minigrid, 2018.

[39] Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

[40] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[41] Karl Cobbe, Chris Hesse, Jacob Hilton, and John Schulman. Leveraging procedural generation to benchmark reinforcement learning. In *International conference on machine learning*, pages 2048–2056. PMLR, 2020.

[42] Cédric Colas, Olivier Sigaud, and Pierre-Yves Oudeyer. How many random seeds? statistical power analysis in deep reinforcement learning experiments. *arXiv preprint arXiv:1806.08295*, 2018.

[43] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015.

[44] Michael Crawshaw. Multi-task learning with deep neural networks: A survey. *arXiv preprint arXiv:2009.09796*, 2020.

[45] Molly J Crockett, Luke Clark, Annemieke M Apergis-Schoute, Sharon Morein-Zamir, and Trevor W Robbins. Serotonin modulates the effects of pavlovian aversive predictions on response vigor. *Neuropsychopharmacology*, 37(10):2244–2252, 2012.

[46] Antoine Cully, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret. Robots that can adapt like animals. *Nature*, 521(7553):503–507, 2015.

[47] Ashok Cutkosky and Harsh Mehta. Momentum improves normalized sgd. In *International conference on machine learning*, pages 2260–2268. PMLR, 2020.

[48] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3150–3158, 2016.

[49] Henry Dale. Pharmacology and nerve-endings. *Proceedings of the Royal Society of Medicine*, 28(3):319–332, 1935.

[50] David B D'Ambrosio and Kenneth O Stanley. A novel generative encoding for exploiting neural network sensor and output geometry. In *Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 974–981, 2007.

[51] Kenneth Alan De Jong. *An analysis of the behavior of a class of genetic adaptive systems.* University of Michigan, 1975.

[52] Matthias De Lange, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Aleš Leonardis, Gregory Slabaugh, and Tinne Tuytelaars. A continual learning survey: Defying forgetting in classification tasks. *IEEE transactions on pattern analysis and machine intelligence*, 44(7):3366–3385, 2021.

[53] Georgios Detorakis, Travis Bartley, and Emre Neftci. Contrastive hebbian learning with random feedback weights. *Neural Networks*, 114:1–14, 2019.

[54] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. Learning modular neural network policies for multi-task and multi-robot transfer. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 2169–2176. IEEE, 2017.

[55] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics.

[56] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021.

[57] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[58] Kenji Doya. Metalearning and neuromodulation. *Neural networks*, 15(4-6):495–506, 2002.

[59] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *International conference on machine learning*, pages 1329–1338. PMLR, 2016.

[60] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[61] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7), 2011.

[62] Felix Eckstein, Akshay S Chaudhari, David Fuerst, Martin Gaisberger, Jana Kemnitz, Christian F Baumgartner, Ender Konukoglu, David J Hunter, and Wolfgang Wirth. Detection of differences in longitudinal cartilage thickness loss using a deep-learning automated segmentation algorithm: Data from the foundation for the national institutes of health biomarkers study of the osteoarthritis initiative. *Arthritis Care & Research*, 74(6):929–936, 2022.

[63] Harrison Edwards and Amos Storkey. Towards a neural statistician. In *International Conference on Learning Representations*, 2017.

[64] L Eshelman. On crossover as an evolutionarily viable strategy. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 61–68. Morgan Kaufmann Publishers San Francisco, 1991.

[65] Lasse Espeholt, Hubert Soyer, Remi Munos, Karen Simonyan, Vlad Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International conference on machine learning*, pages 1407–1416. PMLR, 2018.

[66] Mehrdad Farajtabar, Navid Azizan, Alex Mott, and Ang Li. Orthogonal gradient descent for continual learning. In *International Conference on Artificial Intelligence and Statistics*, pages 3762–3773. PMLR, 2020.

[67] Jean-Marc Fellous and Christiane Linster. Computational models of neuromodulation. *Neural computation*, 10(4):771–805, 1998.

[68] Chrisantha Fernando, Jakub Sygnowski, Simon Osindero, Jane Wang, Tom Schaul, Denis Teplyashin, Pablo Sprechmann, Alexander Pritzel, and Andrei Rusu. Meta-learning by the baldwin effect. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, pages 1313–1320, 2018.

[69] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1126–1135. JMLR. org, 2017.

[70] Dario Floreano and Laurent Keller. Evolution of adaptive behaviour in robots by means of darwinian selection. *PLoS Biol*, 8(1):e1000292, 2010.

[71] Dario Floreano and Francesco Mondada. Evolution of plastic neurocontrollers for situated agents. In *Proc. of The Fourth International Conference on Simulation of Adaptive Behavior (SAB), From Animals to Animats*. ETH Zürich, 1996.

[72] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. *arXiv preprint arXiv:1803.03635*, 2018.

[73] Richard M Friedberg. A learning machine: Part i. *IBM Journal of Research and Development*, 2(1):2–13, 1958.

[74] Justin Fu and Irving Hsu. Model-based reinforcement learning for playing atari games. *Accessed December*, 7, 2016.

[75] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International conference on machine learning*, pages 1587–1596. PMLR, 2018.

[76] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202, 1980.

[77] Adam Gaier and David Ha. Weight agnostic neural networks. *Advances in neural information processing systems*, 32, 2019.

[78] Yuan Gao, Jiayi Ma, Mingbo Zhao, Wei Liu, and Alan L Yuille. Nddr-cnn: Layerwise feature fusing in multi-task cnns by neural discriminative dimensionality reduction. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3205–3214, 2019.

[79] Jason Gauci, Kenneth O Stanley, et al. A case study on the critical role of geometric regularity in machine learning. In *AAAI*, pages 628–633, 2008.

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

[81] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256. JMLR Workshop and Conference Proceedings, 2010.

[82] Thomas Goerttler and Klaus Obermayer. Exploring the similarity of representations in model-agnostic meta-learning. In *Learning to Learn-Workshop at ICLR 2021*, 2021.

[83] David E Goldberg, Kalyanmoy Deb, Hillol Kargupta, and Georges Harik. Rapid, accurate optimization of difficult problems using messy genetic algorithms. In *Proceedings of the Fifth International Conference on Genetic Algorithms,(Urbana, USA)*, pages 59–64. Proceedings of the Fifth International Conference on Genetic Algorithms . . . , 1993.

[84] Ian J Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio. An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*, 2013.

[85] Arthur Gretton, Olivier Bousquet, Alex Smola, and Bernhard Schölkopf. Measuring statistical dependence with hilbert-schmidt norms. In *International conference on algorithmic learning theory*, pages 63–77. Springer, 2005.

[86] Yunhui Guo, Mingrui Liu, Tianbao Yang, and Tajana Rosing. Improved schemes for episodic memory-based lifelong learning. *Advances in Neural Information Processing Systems*, 33:1023–1035, 2020.

[87] Abhishek Gupta, Russell Mendonca, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Meta-reinforcement learning of structured exploration strategies. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 5307–5316, 2018.

[88] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *Advances in Neural Information Processing Systems*, pages 2450–2462, 2018.

[89] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1861–1870. PMLR, 2018.

[90] Raia Hadsell, Dushyant Rao, Andrei A Rusu, and Razvan Pascanu. Embracing change: Continual learning in deep neural networks. *Trends in cognitive sciences*, 24(12):1028–1040, 2020.

[91] Danijar Hafner, Timothy Lillicrap, Jimmy Ba, and Mohammad Norouzi. Dream to control: Learning behaviors by latent imagination. In *International Conference on Learning Representations*, 2020.

[92] Danijar Hafner, Timothy Lillicrap, Ian Fischer, Ruben Villegas, David Ha, Honglak Lee, and James Davidson. Learning latent dynamics for planning from pixels. In *International conference on machine learning*, pages 2555–2565. PMLR, 2019.

[93] Danijar Hafner, Timothy P Lillicrap, Mohammad Norouzi, and Jimmy Ba. Mastering atari with discrete world models. In *International Conference on Learning Representations*, 2021.

[94] Jessica B Hamrick, Abram L. Friesen, Feryal Behbahani, Arthur Guez, Fabio Viola, Sims Witherspoon, Thomas Anthony, Lars Holger Buesing, Petar Veličković, and Theophane Weber. On the role of planning in model-based deep reinforcement learning. In *International Conference on Learning Representations*, 2021.

[95] Nikolaus Hansen, Sibylle D Müller, and Petros Koumoutsakos. Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es). *Evolutionary computation*, 11(1):1–18, 2003.

[96] Michael E Hasselmo and Jill McGaughy. High acetylcholine levels set circuit dynamics for attention and encoding and low acetylcholine levels set dynamics for consolidation. *Progress in brain research*, 145:207–231, 2004.

[97] Jeffrey Hawke, Vijay Badrinarayanan, Alex Kendall, et al. Reimagining an autonomous vehicle. *arXiv preprint arXiv:2108.05805*, 2021.

[98] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034, 2015.

[99] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[100] DO Hebb. The organization of behavior; a neuropsychological theory. 1949.

[101] Nicolas Heess, Greg Wayne, Yuval Tassa, Timothy Lillicrap, Martin Riedmiller, and David Silver. Learning and transfer of modulated locomotor controllers. *arXiv preprint arXiv:1610.05182*, 2016.

[102] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

[103] Matteo Hessel, Joseph Modayil, Hado Van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Azar, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *Thirty-second AAAI conference on artificial intelligence*, 2018.

[104] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent. *Cited on*, 14(8):2, 2012.

[105] Geoffrey E Hinton. Learning translation invariant recognition in a massively parallel networks. In *International Conference on Parallel Architectures and Languages Europe*, pages 1–13. Springer, 1987.

[106] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[107] Sepp Hochreiter, A Steven Younger, and Peter R Conwell. Learning to learn using gradient descent. In *International Conference on Artificial Neural Networks*, pages 87–94. Springer, 2001.

[108] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. *The Journal of Machine Learning Research*, 22(1):10882–11005, 2021.

[109] Sebastian Hofstätter, Aldo Lipani, Markus Zlabinger, and Allan Hanbury. Learning to re-rank with contextualized stopwords. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2057–2060, 2020.

[110] G Zacharias Holland, Erin J Talvitie, and Michael Bowling. The effect of planning shape on dyna-style planning in high-dimensional state spaces. *arXiv preprint arXiv:1806.01825*, 2018.

[111] John H Holland. Outline for a logical theory of adaptive systems. *Journal of the ACM (JACM)*, 9(3):297–314, 1962.

[112] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9):5149–5169, 2021.

[113] Kyle Hsu, Sergey Levine, and Chelsea Finn. Unsupervised learning via meta-learning. In *International Conference on Learning Representations*, 2019.

[114] Shih-Cheng Huang, Anuj Pareek, Saeed Seyyedi, Imon Banerjee, and Matthew P Lungren. Fusion of medical imaging and electronic health records using deep learning: a systematic review and implementation guidelines. *NPJ digital medicine*, 3(1):136, 2020.

[115] Jan Humplik, Alexandre Galashov, Leonard Hasenclever, Pedro A Ortega, Yee Whye Teh, and Nicolas Heess. Meta reinforcement learning as task inference. *arXiv preprint arXiv:1905.06424*, 2019.

[116] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015.

[117] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z Leibo, David Silver, and Koray Kavukcuoglu. Reinforcement learning with unsupervised auxiliary tasks. In *International Conference on Learning Representations*, 2017.

[118] Miguel Jaques, Michael Burke, and Timothy Hospedales. Physics-as-inverse-graphics: Unsupervised physical parameter estimation from video. In *Eighth International Conference on Learning Representations*, pages 1–16, 2020.

[119] Matthew Johnson, Katja Hofmann, Tim Hutton, and David Bignell. The malmo platform for artificial intelligence experimentation. In *IJCAI*, pages 4246–4247, 2016.

[120] John Jumper, Richard Evans, Alexander Pritzel, Tim Green, Michael Figurnov, Olaf Ronneberger, Kathryn Tunyasuvunakool, Russ Bates, Augustin Žídek, Anna Potapenko, et al. Highly accurate protein structure prediction with alphafold. *Nature*, 596(7873):583–589, 2021.

[121] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Continual reinforcement learning with complex synapses. In *International Conference on Machine Learning*, pages 2497–2506. PMLR, 2018.

[122] Christos Kaplanis, Murray Shanahan, and Claudia Clopath. Policy consolidation for continual reinforcement learning. *arXiv preprint arXiv:1902.00255*, 2019.

[123] Michał Kempka, Marek Wydmuch, Grzegorz Runc, Jakub Toczek, and Wojciech Jaśkowski. Vizdoom: A doom-based ai research platform for visual reinforcement learning. In *2016 IEEE conference on computational intelligence and games (CIG)*, pages 1–8. IEEE, 2016.

[124] Alex Kendall, Jeffrey Hawke, David Janz, Przemyslaw Mazur, Daniele Reda, John-Mark Allen, Vinh-Dieu Lam, Alex Bewley, and Amar Shah. Learning to drive in a day. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 8248–8254. IEEE, 2019.

[125] GA Kerkut and RJ Walker. The effect of l-glutamate, acetylcholine and gamma-aminobutyric acid on the miniature end-plate potentials and contractures of the coxal muscles of the cockroach, periplaneta americana. *Comparative Biochemistry and Physiology*, 17(2):435–454, 1966.

[126] Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J Roberts. Unclear: A straightforward method for continual reinforcement learning. In *Proceedings of the 37th International Conference on Machine Learning*, 2020.

[127] Samuel Kessler, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J Roberts. Same state, different task: Continual reinforcement learning without interference. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 7143–7151, 2022.

[128] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. Towards continual reinforcement learning: A review and perspectives. *arXiv preprint arXiv:2012.13490*, 2020.

[129] Siavash Khodadadeh, Ladislau Boloni, and Mubarak Shah. Unsupervised meta-learning for few-shot image classification. *Advances in neural information processing systems*, 32, 2019.

[130] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2015.

[131] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[132] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al. Overcoming catastrophic forgetting in neural networks. *Proceedings of the national academy of sciences*, 114(13):3521–3526, 2017.

[133] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

[134] Gregory Koch, Richard Zemel, Ruslan Salakhutdinov, et al. Siamese neural networks for one-shot image recognition. In *ICML deep learning workshop*, volume 2, page 0. Lille, 2015.

[135] Feliks Kogan, Evan Levine, Akshay S Chaudhari, Uchechukwuka D Monu, Kevin Epperson, Edwin HG Oei, Garry E Gold, and Brian A Hargreaves. Simultaneous bilateral-knee mr imaging. *Magnetic resonance in medicine*, 80(2):529–537, 2018.

[136] Soheil Kolouri, Nicholas A Ketz, Andrea Soltoggio, and Praveen K Pilly. Sliced cramer synaptic consolidation for preserving deeply learned representations. In *International Conference on Learning Representations*, 2019.

[137] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *International Conference on Machine Learning*, pages 3519–3529. PMLR, 2019.

[138] Nils Koster, Oliver Grothe, and Achim Rettinger. Signing the supermask: Keep, hide, invert. *arXiv preprint arXiv:2201.13361*, 2022.

[139] John R Koza et al. Hierarchical genetic algorithms operating on populations of computer programs. In *IJCAI*, volume 89, pages 768–774, 1989.

[140] Nikolaus Kriegeskorte, Marieke Mur, and Peter A Bandettini. Representational similarity analysis-connecting the branches of systems neuroscience. *Frontiers in systems neuroscience*, 2:4, 2008.

[141] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25: 26th Annual Conference on Neural Information Processing Systems 2012. Proceedings of a meeting held December 3-6, 2012, Lake Tahoe, Nevada, United States*, pages 1106–1114, 2012.

[142] Anders Krogh and John Hertz. A simple weight decay can improve generalization. *Advances in neural information processing systems*, 4, 1991.

[143] Dhireesha Kudithipudi, Mario Aguilar-Simon, Jonathan Babb, Maxim Bazhenov, Douglas Blackiston, Josh Bongard, Andrew P Brna, Suraj Chakravarthi Raja, Nick Cheney, Jeff Clune, et al. Biological underpinnings for lifelong learning machines. *Nature Machine Intelligence*, 4(3):196–210, 2022.

[144] Pawel Ladosz, Eseoghene Ben-Iwhiwhu, Jeffery Dick, Nicholas Ketz, Soheil Kolouri, Jeffrey L Krichmar, Praveen K Pilly, and Andrea Soltoggio. Deep reinforcement learning with modulated hebbian plus q-network architecture. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[145] William B Langdon. Genetic programming—computers using "natural selection" to generate programs. In *Genetic programming and data structures*, pages 9–42. Springer, 1998.

[146] Yann LeCun. Generalization and network design strategies. *Connectionism in perspective*, 19(143-155):18, 1989.

[147] Yann LeCun, Bernhard Boser, John Denker, Donnie Henderson, Richard Howard, Wayne Hubbard, and Lawrence Jackel. Handwritten digit recognition with a back-propagation network. *Advances in neural information processing systems*, 2, 1989.

[148] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

[149] Hae Beom Lee, Hayeon Lee, JaeWoong Shin, Eunho Yang, Timothy Hospedales, and Sung Ju Hwang. Online hyperparameter meta-learning with hypergradient distillation. In *International Conference on Learning Representations*, 2022.

[150] June-Goo Lee, Sanghoon Jun, Young-Won Cho, Hyunna Lee, Guk Bae Kim, Joon Beom Seo, and Namkug Kim. Deep learning in medical imaging: general overview. *Korean journal of radiology*, 18(4):570–584, 2017.

[151] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *International Conference on Machine Learning*, pages 2927–2936. PMLR, 2018.

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

[153] Joel Lehman and Kenneth O Stanley. Novelty search and the problem with objectives. In *Genetic programming theory and practice IX*, pages 37–56. Springer, 2011.

[154] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020.

[155] Zhenguo Li, Fengwei Zhou, Fei Chen, and Hang Li. Meta-sgd: Learning to learn quickly for few-shot learning. *arXiv preprint arXiv:1707.09835*, 2017.

[156] Timothy P Lillicrap, Jonathan J Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning. In *International Conference on Learning Representations*, 2016.

[157] Timothy P Lillicrap, Adam Santoro, Luke Marris, Colin J Akerman, and Geoffrey Hinton. Backpropagation and the brain. *Nature Reviews Neuroscience*, 21(6):335–346, 2020.

[158] Sen Lin, Li Yang, Deliang Fan, and Junshan Zhang. TRGP: Trust region gradient projection for continual learning. In *International Conference on Learning Representations*, 2022.

[159] Aldo Lipani, Ben Carterette, and Emine Yilmaz. From a user model for query sessions to session rank biased precision (srbp). In *Proc. of ICTIR*, 2019.

[160] Chaoyue Liu and Mikhail Belkin. Accelerating sgd with momentum for over-parameterized learning. In *International Conference on Learning Representations*, 2020.

[161] Evan Z Liu, Aditi Raghunathan, Percy Liang, and Chelsea Finn. Decoupling exploration and exploitation for meta-reinforcement learning without sacrifices. In *International Conference on Machine Learning*, pages 6925–6935. PMLR, 2021.

[162] Evan Zheran Liu, Moritz Stephan, Allen Nie, Chris Piech, Emma Brunskill, and Chelsea Finn. Giving feedback on interactive student programs with

meta-exploration. In *36th Conference on Neural Information Processing Systems (NeurIPS)*, 2022.

[163] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2019.

[164] Shikun Liu, Edward Johns, and Andrew J Davison. End-to-end multi-task learning with attention. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 1871–1880, 2019.

[165] Xiaodong Liu, Jianfeng Gao, Xiaodong He, Li Deng, Kevin Duh, and Ye-Yi Wang. Representation learning using multi-task deep neural networks for semantic classification and information retrieval. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 912–921, 2015.

[166] Xinran Liu, Yikun Bai, Yuzhe Lu, Andrea Soltoggio, and Soheil Kolouri. Wasserstein task embedding for measuring task similarities. *arXiv preprint arXiv:2208.11726*, 2022.

[167] David Lopez-Paz and Marc'Aurelio Ranzato. Gradient episodic memory for continual learning. *Advances in neural information processing systems*, 30, 2017.

[168] Yiren Lu, Justin Fu, George Tucker, Xinlei Pan, Eli Bronstein, Becca Roelofs, Benjamin Sapp, Brandyn White, Aleksandra Faust, Shimon Whiteson, et al. Imitation is not enough: Robustifying imitation with reinforcement learning for challenging driving scenarios. *arXiv preprint arXiv:2212.11419*, 2022.

[169] Song Luan, Ian Williams, Konstantin Nikolic, and Timothy G Constandinou. Neuromodulation: present and emerging methods. *Frontiers in neuroengineering*, 7:27, 2014.

[170] Wenjie Luo, Cheolho Park, Andre Cornman, Benjamin Sapp, and Dragomir Anguelov. Jfp: Joint future prediction with interactive multi-agent modeling for autonomous driving. *arXiv preprint arXiv:2212.08710*, 2022.

[171] Jiaqi Ma, Zhe Zhao, Xinyang Yi, Jilin Chen, Lichan Hong, and Ed H Chi. Modeling task relationships in multi-task learning with multi-gate mixture-of-experts. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1930–1939, 2018.

[172] David JC MacKay. A practical bayesian framework for backpropagation networks. *Neural computation*, 4(3):448–472, 1992.

[173] Arun Mallya, Dillon Davis, and Svetlana Lazebnik. Piggyback: Adapting a single network to multiple tasks by learning to mask weights. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 67–82, 2018.

[174] Arun Mallya and Svetlana Lazebnik. Packnet: Adding multiple tasks to a single network by iterative pruning. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7765–7773, 2018.

[175] Eve Marder. Neuromodulation of neuronal circuits: back to the future. *Neuron*, 76(1):1–11, 2012.

[176] Charles E Martin and Praveen K Pilly. Probabilistic program neurogenesis. In *ALIFE 2019: The 2019 Conference on Artificial Life*, pages 440–447. MIT Press, 2019.

[177] Gary McHale and Phil Husbands. Gasnets and other evolvable neural networks applied to bipedal locomotion. *From Animals to Animats*, 8:163–172, 2004.

[178] Jorge A Mendez, Harm van Seijen, and Eric Eaton. Modular lifelong reinforcement learning via neural composition. In *International Conference on Learning Representations*, 2022.

[179] Luke Metz, Niru Maheswaranathan, Brian Cheung, and Jascha Sohl-Dickstein. Meta-learning update rules for unsupervised representation learning. In *International Conference on Learning Representations*, 2019.

[180] Elliot Meyerson and Risto Miikkulainen. Beyond shared hierarchies: Deep multitask learning through soft layer ordering. In *International Conference on Learning Representations*, 2018.

[181] Vincent Micheli, Eloi Alonso, and François Fleuret. Transformers are sample efficient world models. *arXiv preprint arXiv:2209.00588*, 2022.

[182] Thomas Miconi, Aditya Rawal, Jeff Clune, and Kenneth O. Stanley. Backpropamine: training self-modifying neural networks with differentiable neuromodulated plasticity. In *International Conference on Learning Representations*, 2019.

[183] Kieran Milan, Joel Veness, James Kirkpatrick, Michael Bowling, Anna Koop, and Demis Hassabis. The forget-me-not process. *Advances in Neural Information Processing Systems*, 29, 2016.

[184] Azalia Mirhoseini, Anna Goldie, Mustafa Yazgan, Joe Jiang, Ebrahim Songhori, Shen Wang, Young-Joon Lee, Eric Johnson, Omkar Pathak, Sungmin Bae, et al. Chip placement with deep reinforcement learning. *arXiv preprint arXiv:2004.10746*, 2020.

[185] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *International Conference on Learning Representations*, 2018.

[186] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. Cross-stitch networks for multi-task learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3994–4003, 2016.

[187] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International conference on machine learning*, pages 1928–1937. PMLR, 2016.

[188] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[189] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

[190] John E Moody. Note on generalization, regularization and architecture selection in nonlinear learning systems. In *Neural Networks for Signal Processing Proceedings of the 1991 IEEE Workshop*, pages 1–10. IEEE, 1991.

[191] Ari Morcos, Maithra Raghu, and Samy Bengio. Insights on representational similarity in neural networks with canonical correlation. *Advances in Neural Information Processing Systems*, 31, 2018.

[192] Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909*, 2015.

[193] Tsendsuren Munkhdalai and Hong Yu. Meta networks. In *International Conference on Machine Learning*, pages 2554–2563. PMLR, 2017.

[194] Elias Najarro and Sebastian Risi. Meta-learning through hebbian plasticity in random networks. *Advances in Neural Information Processing Systems*, 33:20719–20731, 2020.

[195] Kensuke Nakamura and Byung-Woo Hong. Adaptive weight decay for deep neural networks. *IEEE Access*, 7:118857–118865, 2019.

[196] Yu E Nesterov. A method for solving the convex programming problem with convergence rate $obigl(frac1k^2bigr)$. In *Dokl. Akad. Nauk SSSR,*, volume 269, pages 543–547, 1983.

[197] Alexander New, Megan Baker, Eric Nguyen, and Gautam Vallabha. Lifelong learning metrics. *arXiv preprint arXiv:2201.08278*, 2022.

[198] Ehren L Newman, Kishan Gupta, Jason R Climer, Caitlin K Monaghan, and Michael E Hasselmo. Cholinergic modulation of cognitive processing: insights drawn from computational models. *Frontiers in behavioral neuroscience*, page 24, 2012.

[199] Junhyuk Oh, Satinder Singh, and Honglak Lee. Value prediction network. *Advances in neural information processing systems*, 30, 2017.

[200] Erkki Oja. Neural networks, principal components, and subspaces. *International journal of neural systems*, 1(01):61–68, 1989.

[201] Andreas Ostermeier. An evolution strategy with momentum adaptation of the random number distribution. In *PPSN*, pages 199–208, 1992.

[202] Praveen Palanisamy. Multi-agent connected autonomous driving using deep reinforcement learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7. IEEE, 2020.

[203] German I Parisi, Ronald Kemker, Jose L Part, Christopher Kanan, and Stefan Wermter. Continual lifelong learning with neural networks: A review. *Neural Networks*, 113:54–71, 2019.

[204] Andreas Precht Poulsen, Mark Thorhauge, Mikkel Hvilshj Funch, and Sebastian Risi. Dlne: A hybridization of deep learning and neuroevolution for visual control. In *2017 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 256–263. IEEE, 2017.

[205] Sam Powers, Eliot Xing, Eric Kolve, Roozbeh Mottaghi, and Abhinav Gupta. Cora: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents. In *Conference on Lifelong Learning Agents (CoLLAs)*, 2022.

[206] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[207] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals. Rapid learning or feature reuse? towards understanding the effectiveness of maml. In *International Conference on Learning Representations*, 2020.

[208] Pranav Rajpurkar, Jeremy Irvin, Robyn L Ball, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis P Langlotz, et al. Deep learning for chest radiograph diagnosis: A retrospective comparison of the chexnext algorithm to practicing radiologists. *PLoS medicine*, 15(11):e1002686, 2018.

[209] Kate Rakelly, Aurick Zhou, Chelsea Finn, Sergey Levine, and Deirdre Quillen. Efficient off-policy meta-reinforcement learning via probabilistic context variables. In *International Conference on Machine Learning*, pages 5331–5340, 2019.

[210] Vivek Ramanujan, Mitchell Wortsman, Aniruddha Kembhavi, Ali Farhadi, and Mohammad Rastegari. What's hidden in a randomly weighted neural network? In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11893–11902, 2020.

[211] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International Conference on Learning Representations*, 2017.

[212] Ingo Rechenberg. Evolutionsstrategie. *Optimierung technischer Systeme nach Prinzipien derbiologischen Evolution*, 1973.

[213] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 779–788, 2016.

[214] Danilo Rezende, Ivo Danihelka, Karol Gregor, Daan Wierstra, et al. One-shot generalization in deep generative models. In *International conference on machine learning*, pages 1521–1529. PMLR, 2016.

[215] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In

International conference on machine learning, pages 1278–1286. PMLR, 2014.

[216] Sebastian Risi and Kenneth O Stanley. Deep neuroevolution of recurrent and discrete world models. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 456–462, 2019.

[217] Sebastian Risi and Kenneth O Stanley. Improving deep neuroevolution via deep innovation protection. *arXiv preprint arXiv:2001.01683*, 2019.

[218] Sebastian Risi and Kenneth O Stanley. Deep innovation protection: Confronting the credit assignment problem in training heterogeneous neural architectures. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 35, pages 12391–12399, 2021.

[219] David Rolnick, Arun Ahuja, Jonathan Schwarz, Timothy Lillicrap, and Gregory Wayne. Experience replay for continual learning. *Advances in Neural Information Processing Systems*, 32, 2019.

[220] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18*, pages 234–241. Springer, 2015.

[221] Jonas Rothfuss, Dennis Lee, Ignasi Clavera, Tamim Asfour, and Pieter Abbeel. ProMP: Proximal meta-policy search. In *International Conference on Learning Representations*, 2019.

[222] Sebastian Ruder, Joachim Bingel, Isabelle Augenstein, and Anders Søgaard. Latent multi-task architecture learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4822–4829, 2019.

[223] David E Rumelhart, Geoffrey E Hinton, Ronald J Williams, et al. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1, 1988.

[224] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. Progressive neural networks. *arXiv preprint arXiv:1606.04671*, 2016.

[225] Gobinda Saha, Isha Garg, and Kaushik Roy. Gradient projection memory for continual learning. In *International Conference on Learning Representations*, 2021.

[226] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv preprint arXiv:1703.03864*, 2017.

[227] Luis Sanchez, Jiyin He, Jarana Manotumruksa, Dyaa Albakour, Miguel Martinez, and Aldo Lipani. Easing legal news monitoring with learning to rank and bert. In *Advances in Information Retrieval: 42nd European Conference on IR Research, ECIR 2020, Lisbon, Portugal, April 14–17, 2020, Proceedings, Part II 42*, pages 336–343. Springer, 2020.

[228] Terence D Sanger. Optimal unsupervised learning in a single-layer linear feedforward neural network. *Neural networks*, 2(6):459–473, 1989.

[229] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *International conference on machine learning*, pages 1842–1850. PMLR, 2016.

[230] Andrew M Saxe, James L McClelland, and Surya Ganguli. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. *arXiv preprint arXiv:1312.6120*, 2013.

[231] Juergen Schmidhuber, Jieyu Zhao, and MA Wiering. Simple principles of metalearning. *Technical report IDSIA*, 69:1–23, 1996.

[232] Jürgen Schmidhuber. *Evolutionary principles in self-referential learning, or on learning how to learn: the meta-meta-... hook*. PhD thesis, Technische Universität München, 1987.

[233] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International conference on machine learning*, pages 1889–1897. PMLR, 2015.

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

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

[236] Wolfram Schultz, Peter Dayan, and P Read Montague. A neural substrate of prediction and reward. *Science*, 275(5306):1593–1599, 1997.

[237] Wolfram Schultz, Wiliam R Stauffer, and Armin Lak. The phasic dopamine signal maturing: from reward via behavioural activation to formal economic utility. *Current opinion in neurobiology*, 43:139–148, 2017.

[238] Jonathan Schwarz, Wojciech Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell. Progress & compress: A scalable framework for continual learning. In *International Conference on Machine Learning*, pages 4528–4537. PMLR, 2018.

[239] Jonathan Schwarz, Siddhant Jayakumar, Razvan Pascanu, Peter E Latham, and Yee Teh. Powerpropagation: A sparsity inducing weight reparameterisation. *Advances in neural information processing systems*, 34:28889–28903, 2021.

[240] Hans-Paul Schwefel. Evolutionsstrategien für die numerische optimierung. In *Numerische Optimierung von Computer-Modellen mittels der Evolutionsstrategie*, pages 123–176. Springer, 1977.

[241] Nicolas Schweighofer, Mathieu Bertin, Kazuhiro Shishida, Yasumasa Okamoto, Saori C Tanaka, Shigeto Yamawaki, and Kenji Doya. Low-serotonin levels increase delayed reward discounting in humans. *Journal of Neuroscience*, 28(17):4528–4532, 2008.

[242] Nicolas Schweighofer and Kenji Doya. Meta-learning in reinforcement learning. *Neural Networks*, 16(1):5–9, 2003.

[243] Joan Serra, Didac Suris, Marius Miron, and Alexandros Karatzoglou. Overcoming catastrophic forgetting with hard attention to the task. In *International Conference on Machine Learning*, pages 4548–4557. PMLR, 2018.

[244] Ben Seymour, Nathaniel D Daw, Jonathan P Roiser, Peter Dayan, and Ray Dolan. Serotonin selectively modulates reward value in human decision-making. *Journal of Neuroscience*, 32(17):5833–5842, 2012.

[245] Kun Shao, Dongbin Zhao, Nannan Li, and Yuanheng Zhu. Learning battles in vizdoom via deep reinforcement learning. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*, pages 1–4. IEEE, 2018.

[246] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484–489, 2016.

[247] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al. A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. *Science*, 362(6419):1140–1144, 2018.

[248] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30, 2017.

[249] Ghada Sokar, Decebal Constantin Mocanu, and Mykola Pechenizkiy. Spacenet: Make free space for continual learning. *Neurocomputing*, 439:1–11, 2021.

[250] Andrea Soltoggio, Eseoghene Ben-Iwhiwhu, Christos Peridis, Pawel Ladosz, Jeffery Dick, Praveen K Pilly, and Soheil Kolouri. The configurable tree graph (ct-graph): measurable problems in partially observable and distal reward environments for lifelong reinforcement learning. *arXiv preprint arXiv:2302.10887*, 2023.

[251] Andrea Soltoggio, John A Bullinaria, Claudio Mattiussi, Peter Dürr, and Dario Floreano. Evolutionary advantages of neuromodulated plasticity in dynamic, reward-based scenarios. In *Proceedings of the 11th international conference on artificial life (Alife XI)*, pages 569–576. MIT Press, 2008.

[252] Andrea Soltoggio, Peter Durr, Claudio Mattiussi, and Dario Floreano. Evolving neuromodulatory topologies for reinforcement learning-like problems. In *2007 IEEE Congress on Evolutionary Computation*, pages 2471–2478. IEEE, 2007.

[253] Andrea Soltoggio and Ben Jones. Novelty of behaviour as a basis for the neuro-evolution of operant reward learning. In *Proceedings of the 11th Annual conference on Genetic and evolutionary computation*, pages 169–176. ACM, 2009.

[254] Andrea Soltoggio, Pawel Ladosz, Eseoghene Ben-Iwhiwhu, and Jeff Dick. The CT-graph environments, 2019.

[255] Andrea Soltoggio, Kenneth O Stanley, and Sebastian Risi. Born to learn: the inspiration, progress, and future of evolved plastic artificial neural networks. *Neural Networks*, 108:48–67, 2018.

[256] Xingyou Song, Wenbo Gao, Yuxiang Yang, Krzysztof Choromanski, Aldo Pacchiano, and Yunhao Tang. Es-maml: Simple hessian-free meta learning. *arXiv preprint arXiv:1910.01215*, 2019.

[257] Bradly C Stadie, Ge Yang, Rein Houthooft, Xi Chen, Yan Duan, Yuhuai Wu, Pieter Abbeel, and Ilya Sutskever. Some considerations on learning to explore via meta-reinforcement learning. *arXiv preprint arXiv:1803.01118*, 2018.

[258] Kenneth O Stanley, Jeff Clune, Joel Lehman, and Risto Miikkulainen. Designing neural networks through neuroevolution. *Nature Machine Intelligence*, 1(1):24–35, 2019.

[259] Kenneth O Stanley, David B D'Ambrosio, and Jason Gauci. A hypercube-based encoding for evolving large-scale neural networks. *Artificial life*, 15(2):185–212, 2009.

[260] Kenneth O Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary computation*, 10(2):99–127, 2002.

[261] William R Stauffer, Armin Lak, and Wolfram Schultz. Dopamine reward prediction error responses reflect marginal utility. *Current biology*, 24(21):2491–2500, 2014.

[262] Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. Many task learning with task routing. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1375–1384, 2019.

[263] Gjorgji Strezoski, Nanne van Noord, and Marcel Worring. Learning task relatedness in multi-task learning for images in context. In *Proceedings of the 2019 on International Conference on Multimedia Retrieval*, pages 78–86, 2019.

[264] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. *arXiv preprint arXiv:1712.06567*, 2017.

[265] Tianxiang Sun, Yunfan Shao, Xiaonan Li, Pengfei Liu, Hang Yan, Xipeng Qiu, and Xuanjing Huang. Learning sparse sharing architectures for multiple tasks. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 8936–8943, 2020.

[266] Ximeng Sun, Rameswar Panda, Rogerio Feris, and Kate Saenko. Adashare: Learning what to share for efficient deep multi-task learning. *Advances in Neural Information Processing Systems*, 33:8728–8740, 2020.

[267] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[268] Saori C Tanaka, Kazuhiro Shishida, Nicolas Schweighofer, Yasumasa Okamoto, Shigeto Yamawaki, and Kenji Doya. Serotonin affects association of aversive outcomes to past actions. *Journal of Neuroscience*, 29(50):15669–15674, 2009.

[269] Alexander Thiele and Mark A Bellgrove. Neuromodulation of attention. *Neuron*, 97(4):769–785, 2018.

[270] Sebastian Thrun and Lorien Pratt. Learning to learn: Introduction and overview. In *Learning to learn*, pages 3–17. Springer, 1998.

[271] Yuandong Tian, Jerry Ma, Qucheng Gong, Shubho Sengupta, Zhuoyuan Chen, James Pinkerton, and Larry Zitnick. Elf opengo: An analysis and open reimplementation of alphazero. In *International Conference on Machine Learning*, pages 6244–6253. PMLR, 2019.

[272] Emanuel Todorov, Tom Erez, and Yuval Tassa. Mujoco: A physics engine for model-based control. In *2012 IEEE/RSJ international conference on intelligent robots and systems*, pages 5026–5033. IEEE, 2012.

[273] Gido M Van de Ven and Andreas S Tolias. Three scenarios for continual learning. *arXiv preprint arXiv:1904.07734*, 2019.

[274] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Proceedings of the AAAI conference on artificial intelligence*, volume 30, 2016.

[275] Simon Vandenhende, Stamatios Georgoulis, and Luc Van Gool. Mti-net: Multi-scale task interaction networks for multi-task learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part IV 16*, pages 527–543. Springer, 2020.

[276] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[277] Roby Velez and Jeff Clune. Diffusion-based neuromodulation can eliminate catastrophic forgetting in simple neural networks. *PloS one*, 12(11):e0187736, 2017.

[278] Oriol Vinyals, Charles Blundell, Timothy Lillicrap, Daan Wierstra, et al. Matching networks for one shot learning. *Advances in neural information processing systems*, 29, 2016.

[279] Johannes von Oswald, Christian Henning, Benjamin F. Grewe, and João Sacramento. Continual learning with hypernetworks. In *International Conference on Learning Representations*, 2020.

[280] Johannes Von Oswald, Dominic Zhao, Seijin Kobayashi, Simon Schug, Massimo Caccia, Nicolas Zucchet, and João Sacramento. Learning where to learn: Gradient sparsity in meta and continual learning. *Advances in Neural Information Processing Systems*, 34:5250–5263, 2021.

[281] Jane X Wang, Michael King, Nicolas Porcel, Zeb Kurth-Nelson, Tina Zhu, Charlie Deck, Peter Choy, Mary Cassin, Malcolm Reynolds, Francis Song, et al. Alchemy: A structured task distribution for meta-reinforcement learning. *arXiv preprint arXiv:2102.02926*, 2021.

[282] Jane X Wang, Zeb Kurth-Nelson, Dhruva Tirumala, Hubert Soyer, Joel Z Leibo, Rémi Munos, Charles Blundell, Dharshan Kumaran, and Matt Botvinick. Learning to reinforcement learn, 2016. *arXiv preprint arXiv:1611.05763*, 2016.

[283] Ming Wen, Zhimin Zhang, Shaoyu Niu, Haozhi Sha, Ruihan Yang, Yonghuan Yun, and Hongmei Lu. Deep-learning-based drug–target interaction prediction. *Journal of proteome research*, 16(4):1401–1409, 2017.

[284] Paul J Werbos. Applications of advances in nonlinear sensitivity analysis. In *System modeling and optimization*, pages 762–770. Springer, 1982.

[285] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

[286] Maciej Wołczyk, Michał Zając, Razvan Pascanu, Łukasz Kuciński, and Piotr Miłoś. Continual world: A robotic benchmark for continual reinforcement learning. *Advances in Neural Information Processing Systems*, 34:28496–28510, 2021.

[287] Mitchell Wortsman, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi. Supermasks in superposition. *Advances in Neural Information Processing Systems*, 33:15173–15184, 2020.

[288] Yuxin Wu and Kaiming He. Group normalization. In *Proceedings of the European conference on computer vision (ECCV)*, pages 3–19, 2018.

[289] Zihao Wu, Huy Tran, Hamed Pirsiavash, and Soheil Kolouri. Is multitask learning an upper bound for continual learning? *arXiv preprint arXiv:2210.14797*, 2022.

[290] Xiaohui Xie and H Sebastian Seung. Equivalence of backpropagation and contrastive hebbian learning in a layered network. *Neural computation*, 15(2):441–454, 2003.

[291] Jinwei Xing, Xinyun Zou, and Jeffrey L Krichmar. Neuromodulated patience for robot and self-driving vehicle navigation. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020.

[292] Jinwei Xing, Xinyun Zou, Praveen K Pilly, Nicholas A Ketz, and Jeffrey L Krichmar. Adapting to environment changes through neuromodulation of reinforcement learning. In *International Conference on Simulation of Adaptive Behavior*, pages 115–126. Springer, 2022.

[293] Dan Xu, Wanli Ouyang, Xiaogang Wang, and Nicu Sebe. Pad-net: Multi-tasks guided prediction-and-distillation network for simultaneous depth estimation and scene parsing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 675–684, 2018.

[294] Yongxin Yang and Timothy M. Hospedales. Deep multi-task representation learning: A tensor factorisation approach. In *International Conference on Learning Representations*, 2017.

[295] Xin Yao. Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9):1423–1447, 1999.

[296] Fanghua Ye, Jarana Manotumruksa, Qiang Zhang, Shenghui Li, and Emine Yilmaz. Slot self-attentive dialogue state tracking. In *Proceedings of the Web Conference 2021*, pages 1598–1608, 2021.

[297] Jaehong Yoon, Eunho Yang, Jeongtae Lee, and Sung Ju Hwang. Lifelong learning with dynamically expandable networks. In *International Conference on Learning Representations*, 2018.

[298] Jaesik Yoon, Taesup Kim, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn. Bayesian model-agnostic meta-learning. *Advances in neural information processing systems*, 31, 2018.

[299] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. Meta-world: A benchmark and evaluation for multi-task and meta reinforcement learning. In *Conference on robot learning*, pages 1094–1100. PMLR, 2020.

[300] Greg Zaharchuk. Fellow in a box: combining ai and domain knowledge with bayesian networks for differential diagnosis in neuroimaging, 2020.

[301] Arber Zela, Thomas Elsken, Tonmoy Saikia, Yassine Marrakchi, Thomas Brox, and Frank Hutter. Understanding and robustifying differentiable architecture search. In *Proceedings of the International Conference on Learning Representations (ICLR)*, 2020.

[302] Guanxiong Zeng, Yang Chen, Bo Cui, and Shan Yu. Continual learning of context-dependent processing in neural networks. *Nature Machine Intelligence*, 1(8):364–372, 2019.

[303] Friedemann Zenke, Ben Poole, and Surya Ganguli. Continual learning through synaptic intelligence. In *International Conference on Machine Learning*, pages 3987–3995. PMLR, 2017.

[304] Xueting Zhang, Debin Meng, Henry Gouk, and Timothy M Hospedales. Shallow bayesian meta learning for real-world few-shot recognition. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 651–660, 2021.

[305] Xueting Zhang, Yuting Qiang, Flood Sung, Yongxin Yang, and Timothy Hospedales. Relationnet2: Deep comparison network for few-shot learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8, 2020.

[306] Zhanpeng Zhang, Ping Luo, Chen Change Loy, and Xiaoou Tang. Facial landmark detection by deep multi-task learning. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pages 94–108. Springer, 2014.

[307] Zhenyu Zhang, Zhen Cui, Chunyan Xu, Yan Yan, Nicu Sebe, and Jian Yang. Pattern-affinitive propagation across depth, surface normal and semantic segmentation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 4106–4115, 2019.

[308] Xiangyun Zhao, Haoxiang Li, Xiaohui Shen, Xiaodan Liang, and Ying Wu. A modulation module for multi-task learning with applications in image retrieval. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 401–416, 2018.

[309] Hao Zheng and Ahmed Louri. An energy-efficient network-on-chip design using reinforcement learning. In *Proceedings of the 56th Annual Design Automation Conference 2019*, pages 1–6, 2019.

[310] Hattie Zhou, Janice Lan, Rosanne Liu, and Jason Yosinski. Deconstructing lottery tickets: Zeros, signs, and the supermask. *Advances in neural information processing systems*, 32, 2019.

[311] Wei Zhou, Yiying Li, Yongxin Yang, Huaimin Wang, and Timothy Hospedales. Online meta-critic learning for off-policy actor-critic methods. *Advances in neural information processing systems*, 33:17662–17673, 2020.

[312] Wenda Zhou, Victor Veitch, Morgane Austern, Ryan P Adams, and Peter Orbanz. Non-vacuous generalization bounds at the imagenet scale: a pac-bayesian compression approach. *arXiv preprint arXiv:1804.05862*, 2018.

[313] Guangxiang Zhu, Minghao Zhang, Honglak Lee, and Chongjie Zhang. Bridging imagination and reality for model-based deep reinforcement learning. *Advances in Neural Information Processing Systems*, 33:8993–9006, 2020.

[314] Luisa Zintgraf, Kyriacos Shiarli, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. Fast context adaptation via meta-learning. In *International Conference on Machine Learning*, pages 7693–7702, 2019.

[315] Luisa Zintgraf, Kyriacos Shiarlis, Maximilian Igl, Sebastian Schulze, Yarin Gal, Katja Hofmann, and Shimon Whiteson. Varibad: A very good method for bayes-adaptive deep rl via meta-learning. In *International Conference on Learning Representations*, 2020.

[316] Yongshuo Zong, Yongxin Yang, and Timothy Hospedales. MEDFAIR: Benchmarking fairness for medical imaging. In *The Eleventh International Conference on Learning Representations*, 2023.

[317] Fangyu Zou, Li Shen, Zequn Jie, Weizhong Zhang, and Wei Liu. A sufficient condition for convergences of adam and rmsprop. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 11127–11135, 2019.

[318] Xinyun Zou, Soheil Kolouri, Praveen K Pilly, and Jeffrey L Krichmar. Neuromodulated attention and goal-driven perception in uncertain domains. *Neural Networks*, 125:56–69, 2020.

[319] Łukasz Kaiser, Mohammad Babaeizadeh, Piotr Miłos, Błażej Osiński, Roy H Campbell, Konrad Czechowski, Dumitru Erhan, Chelsea Finn, Piotr Kozakowski, Sergey Levine, Afroz Mohiuddin, Ryan Sepassi, George Tucker, and Henryk Michalewski. Model based reinforcement learning for atari. In *International Conference on Learning Representations*, 2020.