UNIVERSITY OF MICHIGAN

UNDERGRADUATE HONORS THESIS

Meta Learning in Continual Reinforcement Learning: An Investigation

---

Author:
Kellen KANARIOS

Co-Advisors:
Professor Lei YING
Professor Mingyan LIU

A thesis submitted in fulfillment of the requirements for the degree of Bachelors of Science

in the

Ying Lab
Electrical and Computer Engineering

August 26, 2024

Declaration of Authorship

I, Kellen KANARIOS, declare that this thesis titled, "Meta Learning in Continual Reinforcement Learning: An Investigation" and the work presented in it are my own. I confirm that:

- This work was done wholly or mainly while in candidature for a research degree at this University.
- Where any part of this thesis has previously been submitted for a degree or any other qualification at this University or any other institution, this has been clearly stated.
- Where I have consulted the published work of others, this is always clearly attributed.
- Where I have quoted from the work of others, the source is always given. With the exception of such quotations, this thesis is entirely my own work.
- I have acknowledged all main sources of help.
- Where the thesis is based on work done by myself jointly with others, I have made clear exactly what was done by others and what I have contributed myself.

Signed: Kellen Kanarios

Date: 8-20-24

Abstract

Kellen KANARIOS

Meta Learning in Continual Reinforcement Learning: An Investigation

Similar to humans, the goal of artificial agents is to be able to interact and learn from the world continually throughout their lifetime. With the emergence of reinforcement learning, a paradigm for these embodied agents has emerged. However, these agents must be trained online in the environment, requiring far more experience than humans to learn specific tasks. An explanation for this can be that our current agents are *tabula rasa*, meaning that they start from a completely blank slate each time they learn. On the contrary, humans are able to somehow construct informative inductive biases or priors from their lifetime of experience. A central question is if we let an agent continually learn then how can we replicate this behavior: concurrently compressing useful information about the world along with making progress toward the current task. A similar idea is "learning to learn" alternatively known as meta-learning. This similarity gives rise to the idea to combine meta learning and reinforcement learning to somehow meta-learn this information. In this thesis, we consider the notion that humans may represent state as predictions and encode this prior as predictions of answers to generally useful questions. To this end, we will explore an existing algorithm to meta learn these questions and consider many of the limitations along with promising directions for future work as we continue on this journey to a generalist agent.

Acknowledgements

I wanted to give a **BIG** thank you to Professor Ying for his patience with me during this project, and I am looking forward to doing much more in the years to come. I also wanted to thank Professor Singh for agreeing to review this work.

Contents

List of Figures

List of Tables

List of Abbreviations

RL	**R**einforcement **L**earning
CRL	**C**ontinual **R**einforcement **L**earning
DNN	**D**eep **N**eural **N**etwork
CNN	**C**onvolutional **N**eural **N**etwork
SGD	**S**tochastic **G**radient **D**escent
MDP	**M**arkov **D**ecision **P**rocess
DQN	**D**eep **Q** **N**etwork
BMG	**B**ootstrapped **M**eta **G**radient
GVF	**G**eneral **V**alue **F**unction
PPO	**P**roximal **P**olicy **O**ptimization
UVFA	**U**niversal **V**alue **F**unction **A**pproximator
USFA	**U**niversal **S**uccessor **F**eature **A**pproximator
ReLU	**Re**ctified **L**inear **U**nit
CReLU	**C**oncatenated **Re**ctified **L**inear **U**nit

Chapter 1

Introduction

In the rapidly advancing field of artificial intelligence, reinforcement learning (RL) has emerged as a powerful paradigm for training agents to make decisions and solve complex problems through interaction with dynamic environments [57, 75, 14, 44]. Traditional reinforcement learning methods, however, are typically designed with the assumption that the environment remains stationary. This assumption often limits the applicability of RL in real-world scenarios, where environments are inherently non-stationary and agents must adapt continually over time. An alternative, but roughly equivalent perspective, is that the agent is much smaller than the environment itself [41]. This means that the agent cannot learn all the information needed to know about the world but must instead continually track the current task.

Continual Reinforcement Learning (CRL) aims to address these challenges by enabling agents to learn and adapt continuously as they encounter new tasks, changes in the environment, and evolving objectives [67, 47, 50, 2]. Unlike standard RL, which typically resets after completing a task, CRL requires the agent to retain knowledge from previous experiences and leverage it to perform better on future tasks. This ongoing learning process poses several unique challenges, including the need for efficient knowledge transfer, maintaining plasticity [53, 52], the prevention of catastrophic forgetting [31, 4], and persistent exploration [84].

One promising approach to overcoming these challenges in CRL is meta-learning [71, 26]. Meta-learning, often referred to as "learning to learn," aims to develop models that can quickly adapt to new tasks by leveraging knowledge of the underlying structure of a set of tasks. In the context of CRL, meta-learning can be particularly valuable as it enables agents to identify and exploit common patterns across tasks, leading to more efficient learning and better generalization. By learning how to learn, the agent can adapt to new situations with minimal retraining, thus enhancing its ability to operate in dynamic environments.

The integration of meta-learning into CRL frameworks holds significant promise for improving the adaptability and robustness of reinforcement learning agents. Meta-learning techniques can facilitate the transfer of knowledge across tasks [19, 45], reduce the impact of catastrophic forgetting [42, 11], and accelerate the learning process in continually changing environments [27]. This synergy between CRL and meta-learning not only enhances the agent's ability to handle new challenges but also provides a foundation for developing more intelligent and versatile systems capable of operating autonomously in the real world.

This thesis aims to advance the understanding of CRL by exploring the viability of existing meta-learning algorithms for continual reinforcement learning. We first instantiate this notion of useful "structure" as predictions about the world [70, 66, 17]. We then investigate whether these predictions can feasibly be learned in a single lifetime based on existing algorithms [80].

1.1 Summary of Contributions

Survey: In the first part of this thesis, we provide necessary background on reinforcement learning. We then present many of the leading definitions for the continual reinforcement learning problem along with their implications. We then provide an extensive review of meta gradient methods of interest for continual reinforcement learning along. Finally, we introduce many of the relevant challenges facing continual learning.

Investigation: In the second part of this thesis, we investigate an existing meta gradient based approach for learning state representations [80]. Namely, we are interested in the reproducibility and stability of this method for future use in the continual learning setting. We also investigate novel variants of this algorithm and the corresponding differences in performance. We provide an open-sourced JAX-based implementation of each of these algorithms, which was previously unavailable from the original paper.

Road Map: Finally, over the course of this project, I encountered many promising ideas and future avenues of research. Due to the constrained nature of this project, I left many of this ideas unexplored experimentally. However, in this chapter, I address many of the issues that plague continual reinforcement learning along with offering potential solutions to alleviate these problems.

Chapter 2

Background

To properly understand continual reinforcement learning (CRL), one must understand many of the adjacent and preceding fields. In this paper, we will frame lifelong learning as continual reinforcement learning. However, lifelong learning is not limited to reinforcement learning and is still an active research area in just the supervised learning setting.

2.1 Reinforcement Learning

In traditional reinforcement learning, we are given the tuple $(\mathcal{S}, \mathcal{A}, p, r, \gamma)$. This consists of

- $\mathcal{S}$ is a collection of states that the agent may encounter
- $\mathcal{A}$ is a collection of actions the agent may choose from
- $p : \mathcal{S} \times \mathcal{A} \to \mathbb{S}$ is the state distribution as determined by the environment.
- $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ is a reward function defined by the environment
- $\gamma \in [0, 1)$ is a discount factor. This is not needed in the episodic case but required in the infinite horizon case.

The objective of the agent is to find a policy $\pi : \mathcal{S} \to \mathbb{R}^{|\mathcal{A}|}$ that maximizes the expected future reward i.e.

$$\pi \in \arg\max_{\pi} \mathbb{E}_{s_t \sim p, a_t \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s \right]$$

It is common to define

$$V^{\pi}(s) = \mathbb{E}_{s_t \sim p, a_t \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s \right]$$

$$Q^{\pi}(s, a) = \mathbb{E}_{s_t \sim p, a_t \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s, a_0 = a \right]$$

Unlike in traditional control, in reinforcement learning the underlying transition dynamics p are nearly never known. This requires learning through interaction and is often categorized into two approaches.

Model-Based: In model-based reinforcement learning, one tries to learn the transition dynamics p. Given the transition dynamics, if $\mathcal{S}$ and $\mathcal{A}$ are finite, then one can explicitly solve for the optimal policy. Additionally, one can use a learned model of

the world to improve sample efficiency by simulating samples rather than directly interacting with the environment.

Model-Free: In model free reinforcement learning, one directly learns the Q function using the Bellman optimality property from dynamic programming [10]. This is done via bootstrapping with the following update

$$Q_{k+1}(s,a) = Q_k(s_t,a_t) + \alpha\Big(r(s_t,a_t) + \gamma Q_k(s_{t+1},\pi(a_t)) - Q_k(s,a)\Big)$$

If $\pi(s)$ is taken as $a \in \arg\max_a Q(s,a)$, then π will converge to the optimal policy. This is known as Q-learning [82] and is the foundation for many modern methods in RL.

2.2 Continual Learning

Much like the rest of continual reinforcement learning, the problem formulation remains an open question. Prior to covering the continual reinforcement learning problem, we will define learning continually in the easier supervised learning setting.

2.2.1 Supervised Learning

An informal introduction to continual learning, can be thought of as how humans interact with the world. An agent or model continuously interacts with an online stream of data across a wide variety of tasks. Unlike in multi task, the task distribution is not fixed and tasks are encountered sequentially. Continual learning began in the supervised learning setting. This setting is formalized as follows

- The learning algorithm makes updates in a minibatch setting, processing M input-output pairs $\{x_i, y_i\}_{i=1}^{M}$, and updating the parameters θ after each minibatch.
- Unique to the continual setting, there is a periodic change to the distribution generating the input-output pairs. Each distribution is referred to as a task.

Similar to [8], the continual supervised and RL settings can be unified by converting the supervised setting into an RL problem. For more details, see [47].

Objective: There are multiple ways that performance is measured in this setting. In [49], they use the notion of *average online task accuracy*. This can be written as

$$\text{Avg Online Task Accuracy}(T_i) = \frac{1}{M}\sum_{j=t_i}^{t_i+M-1} a_j$$

Here, t_i is starting time of task T_i and a_j is the average accuracy on the jth batch of samples. The idea of this metric is to capture how quickly the agent can adapt to new tasks. However, in [52], they consider the average error across all observations in a task at the end of the task.

2.2.2 Reinforcement Learning

Continual reinforcement learning was first proposed by Ring in his thesis [67]. However, there has since been many alternative definitions of what makes a problem a

continual reinforcement learning problem. Generally, CRL is the natural analog to continual supervised learning. The environment is still a sequential arrival of tasks. However, the tasks are now no longer supervised learning tasks but instead distinct MDPs.

Constrained Reinforcement Learning: In [50], they propose continual learning as constrained reinforcement learning. In their formulation, they introduce the average reward objective. Formally, this is maximizing

$$\bar{r}_\pi = \liminf_{T\to\infty} \mathbb{E}_\pi \left[\frac{1}{T} \sum_{t=0}^{T-1} R_{t+1} \right]$$

This is the natural extension of average online task accuracy to the reinforcement learning setting. However, much like humans we have finite compute capacity both in terms of memory and computation. This leads to the objective

$$\begin{aligned} \max_{\pi} \quad & \bar{r}_\pi \\ \text{s.t.} \quad & \text{computational constraint} \end{aligned}$$

For memory constraint, they utilize information theory to constrain the information content of the agent state. The agent state is some summarization of the history. Notably, the agent state cannot be the entire history due to the memory constraints. Computational constraints then refer to FLOPs or other compute metrics. An interesting result of this formulation is the natural decomposition of information error into forgetting and plasticity.

$$\underbrace{\mathbb{E}\left[\mathbf{d}_{\mathrm{KL}}\left(P_t^* \| \tilde{P}_t\right)\right]}_{\text{error}} = \sum_{k=0}^{t} \Big(\underbrace{\mathbb{I}\left(Y_{t+1}; U_{t-k-1} \mid U_{t-k}, H_{t-k:t}\right)}_{\text{forgetting at lag } k} + \underbrace{\mathbb{I}\left(Y_{t+1}; O_{t-k} \mid U_{t-k}, H_{t-k+1:t}\right)}_{\text{implasticity at lag } k} \Big).$$

Here, we have P_t^* is the optimal prediction based on the history and $\tilde{P}_t$ is the best prediction given the agent state. Note that these cannot be the same unless we have unconstrained resources. Therefore, we have an intuitive upper bound on performance decomposed into the familiar issues of catastrophic forgetting and plasticity.

Non-Stationarity Formulation: In [46], they re-formulate the CRL problem as an RL problem with non-stationarity. In its most general form, they allow for each of the components of the RL problem to be non-stationary. However, it is most common that the reward model and transition dynamics are the only sources of non-stationarity. They provide many standard conditions in order to obtain theoretical results, such as Lipschitz non-stationarity. This means that the non-stationarity cannot be too drastic. Additionally, they reframe this non-stationarity as a POMDP problem. To do this, one simply needs to view the non-stationarity as a bunch of stationary MDPs. The current MDP is then a latent random variable. With this, one can make additional assumptions, such as a factored MDP assumption see [46] for details.

New General Formulation: There has been recent work [2] to provide a general formulation for the CRL problem. At a high level, they claim that an RL problem is an instance of a CRL problem if the optimal agent is one that never stops learning. To this end, a problem consists of a basis of policies i.e. every possible policy is some combination of this basis as in the linear algebraic sense. Then the problem is a CRL problem if the optimal agent does not converge to one of these policies in the span of

the basis. It is crucially important to note that the CRL problem is defined in terms of the basis. This means that traditionally non-CRL problems can be an instance of a CRL problem with respect to a certain basis.

2.2.3 Problems

Catastrophic Forgetting: One of the key issues that arises in continual learning is retaining reasonable performance on previously seen tasks [54]. Even in the simplest of supervised learning settings, it has been observed that DNNs struggle to retain prior performance or even comparable performance. This phenomenon can be explained as drastically different tasks "interfering" with previously learned information and overwriting it. One prevailing theory is that the assumptions of traditional gradient based optimization algorithms, such as SGD are violated. Namely, SGD implicitly assumes the distribution of data is stationary and i.i.d. There has been work to try to preserve previously learned skills and knowledge by explicitly preserving the "useful" neurons for those tasks [48]. In [42], they try to meta-learn representations to re-parametrize manifolds in a way that learning new tasks will minimally interfere with previously learned ones.

Loss of Plasticity: An equally important and prevalent problem, is neural networks maintaining plasticity. By this, we mean the ability to continually learn new tasks. The most common approach to mitigate plasticity loss is through regularization [49]. In regularization based approaches, they often regularize to encourage staying close to some random iniitalization. Another approach is to manually inject randomness [21, 58]. In these approaches, they define a utility metric to retain useful learned information while overcoming some of the limitations discussed below. The causes for plasticity are still not completely understood. Existing explanations consist of

- *Decreasing gradient norm* [1]: causes gradient descent to make really small updates and get stuck. All updates will be essentially negligible, making it so that no learning happens,
- *Dormant neurons* [53]: this is when things like ReLU units become and remain zero making them useless in future computation. This effectively kills them for the remainder of the learning process when using gradient based optimization because their gradient will always be zero,
- *Decreasing representation rank* [49]: as the network accumulates dead units the matrix rank of the features decreasing making the representable space literally smaller,
- *Increased parameter norms* [21]: it is hypothesized that the uniform distribution of weights of similar norm somehow allows for fast adaptation, whereas when learning occurs it causes a disparity between norms of weights, making the learner get stuck in its current skill
- *Optimization landscape* [53]: here they claim that the inductive bias of gradient based optimization pushes the optimization into a landscape that is less amenable to continued learning. They show that gradient descent causes far more drastic outliers than random perturbation over time.

Ultimately, standard random initialization seems to offer some improved ability for adaptation, where the inductive biases of gradient based optimization seem to pollute this initialization, causing the learner to get stuck for numerous reasons. Current methods still seem naive in the sense that they just try to preserve the random

initialization for as long as possible rather than truly understanding what makes it beneficial.

Non-Stationarity in CRL: In traditional reinforcement learning, it is assumed that each of $(\mathcal{S}, \mathcal{R}, p, r)$ are stationary. This means that they are independent of the current timestep. In the continual reinforcement learning setting, this is never the case. There is always some sort of non-stationary as the task or even the environment is continually changing. If we suppose that p changes at time T from p_1 to p_2 then we have

$$V^{\pi}(s) = \mathbb{E}_{s_t \sim p_1, a_t \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t) | s_0 = s\right] + \mathbb{E}_{s_t \sim p_2, a_t \sim \pi}\left[\sum_{t=T}^{\infty} \gamma^t r(s_t, a_t) | s_0 = s\right]$$

Note that traditional reinforcement learning optimizes only for the current transition dynamics. This means that we will no longer get an unbiased estimate of $V^{\pi}(S)$ by doing regular RL. This leads to an algorithmic inevitability of catastrophic forgetting without the need for neural network approximation. Intuitively, RL algorithms will always optimize to improve the current dynamics with a disregard for the dynamics in previous tasks. Futhermore, algorithms like DQN [57] are non-stationary supervised learning, making them suffer from loss of plasticity even in traditional reinforcement learning.

Ray Interference: Another challenge unique to reinforcement learning proposed in [68] is what they call ray interference. This interference results in learning plateaus defined as an inflection point from convex to concave. They characterized the cause of ray interference by two components: A learning system suffers from plateaus if it has

(a) negative interference between the components of its objective,

(b) coupled performance and learning progress.

To understand the notion of interference, first suppose that the objective $J(\theta)$ can be decomposed as $J(\theta) = \sum_{k=0}^{N} J_k(\theta)$. Then the interference can be defined as the cosine similarity between the gradients of components, or

$$\rho_{k,k'}(\theta) := \frac{\langle \nabla J_k(\theta), \nabla J_{k'}(\theta) \rangle}{\|\nabla J_k(\theta)\| \, \|\nabla J_{k'}(\theta)\|}.$$

As you might imagine, this is especially prevalent in the continual learning or multi task setting, where there the objective is literally composed of multiple different tasks that may interfere with one another. However, unlike catastrophic interference more difficulties arise due to the couple performance and learning progress present in RL. Due to on policy methods directly interacting with the environment, they will continue to encounter states that their policy performs well on. This can be problematic in a *Winner-Takes-All* scenario. In this scenario, the improvement of the policy on one task results in worse performance on all other tasks. This often leads to being dragged into plateaus severely slowing performance. We see in Figure 2.1, that in the case of RL with function approximation the agent often learns one task completely prior to starting the new task, leading to slower learning.

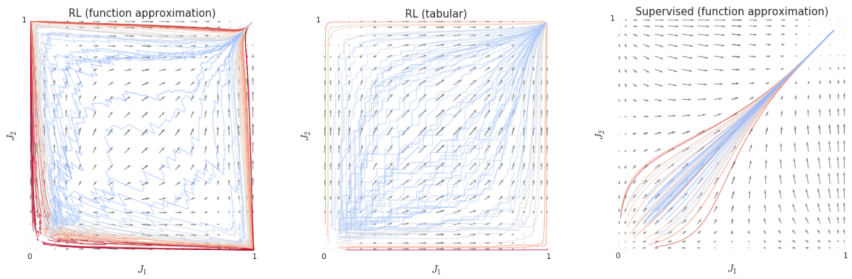

FIGURE 2.1: Illustration from [68], showing how RL with function approximation often takes the slowest path to master both tasks. Each line is a learning trajectory, with darker lines meaning more trajectories.

2.3 Meta Learning

The objective of meta-learning is rather than learning to optimize for the current task you instead want to learn a procedure to solve tasks. This is typically done in a phasic manner, where we have a *meta-train* and *meta-test* phase. Formally, we are given a distribution of tasks $p(\mathcal{T})$, where each task $\mathcal{T} = \{\mathcal{L}, \mathcal{D}\}$ consisting of a dataset and loss function. We then want to find the parameter θ such that

$$\theta = \arg\min_{\theta} \mathbb{E}_{\mathcal{T}\sim p(\mathcal{T})} \left[\mathcal{L}(\mathcal{D}; f_\theta)\right] \tag{2.1}$$

There is an important distinction here that f_θ is the function f trained with parameters θ. Essentially, θ are not the final parameters but the meta-initialization shared across tasks. In meta learning, we are given meta-train tasks and meta-test tasks, consisting of datasets i.e. $\mathcal{D}_{\text{meta-train}} = \{(\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{val}})^{(i)}\}_{i=1}^{M}$. The meta-train phase can then be formalized as

$$\theta^* = \arg\max_{\theta} \log p(\theta|\mathcal{D}_{\text{meta-train}}) \tag{2.2}$$

Intuitively, you want to find the parameterization that has the highest probability of being (2.1) based on the meta training tasks. In the meta-test phase, we then use the learned representation from (2.2) to learn tasks in the meta test set i.e.

$$\theta_i^* = \arg\max_{\theta} \log p(\theta|\mathcal{D}^{(i)}_{\text{meta-test}})$$

See [39] for more details.

MAML: The most prevalant meta-learning algorithm is MAML [26]. Given a loss function $\mathcal{L}$, traditional gradient descent gives us the following update rule

$$\theta' = \theta - \alpha\nabla_\theta\mathcal{L}(f_\theta)$$

MAML then proposes the following higher order update rule

$$\theta = \theta - \alpha\nabla_\theta\mathcal{L}(f_{\theta'})$$

This can be interpreted as finding the parameters that would minimize the loss function after we were to take one step of gradient descent. Equating gradient descent to learning, this would be learning the parameters that learn the best, encapsulating the idea of meta-learning.

Task Distribution: In practice, the MAML-based approach typically requires a task distribution $\mathcal{T}$. You would then find the parameters θ, such that

$$\theta = \arg\max_{\theta} \mathbb{E}_{t \in \mathcal{T}} \left[\mathcal{L}_t(f_\theta) \right]$$

where the task here defines a specific loss. Knowledge of a change in task has been a common assumption in continual learning work [48, 27, 42]. However, humans have the innate ability to distinguish tasks. This has been required in many previous continuous learning works [64, 30]. In [64], they take a sliding window approach, where they train on the previous N examples as new examples come in regardless of task.

2.3.1 Online Meta Learning

The naive approach is presented in [27]. Here they formalize the online meta learning setting. This problem follows standard online learning conventions, where the agent makes decisions sequentially in rounds. However, in this setting they allow for the agent to make *task-specific* update prior to being deployed in each round. This update is denoted $\mathbf{U}_t(\mathbf{w}_t)$. The problem can be summarized as follows

1. At round t, the agent chooses a model defined by $\boldsymbol{w}_t$.
2. The world simultaneously chooses task defined by f_t.
3. The agent obtains access to the update procedure $\boldsymbol{U}_t$, and uses it to update parameters as $\widetilde{\boldsymbol{w}}_t = \boldsymbol{U}_t(\boldsymbol{w}_t)$.
4. The agent incurs loss $f_t(\widetilde{\boldsymbol{w}}_t)$.

The aim is then to minimize the regret defined as

$$\text{Regret}_T = \sum_{t=1}^{T} f_t(\boldsymbol{U}_t(\boldsymbol{w}_t)) - \min_{\boldsymbol{w}} \sum_{t=1}^{T} f_t(\boldsymbol{U}_t(\boldsymbol{w}))$$

To better understand this distinction from traditional online learning, we will consider the case where $\mathbf{U}_t(\mathbf{w}) = \mathbf{w} - \alpha \nabla \hat{f}_t(\mathbf{w})$. As the learner, we want to find parameters $\mathbf{w}$, so that $\mathbf{U}_t(\mathbf{w})$ performs well. Equivalently, we want to find the parameters that best minimize the loss after one step of gradient descent, not just directly minimize the loss as in traditional online learning. This draws strong similarities to few shot learning. The reason we are now able to do this is because $\mathbf{U}_t$ is allowed access to some samples of the current task prior to evaluation. They consider the task aware setting, where each round defines a new task. Ultimately, their proposed algorithm is to run MAML across all of the tasks seen so far. Initially, they considered the task aware setting, requiring task boundaries to identify which task they are in. They later present a task oblivious algorithm in [64]. However, as previously mentioned, this just makes use of a sliding window approach of new data. The outer loop update still utilizes all the previous tasks, remaining computationally infeasible for a lifelong agent. An alternative model based approach is proposed in [45]. Here, they only allow for non-stationary transition dynamics and aim to exploit structural similarity across tasks. They meta learn a regularization parameter for their model. Then perform planning with this semi-meta-learned model.

Continual Online Meta-Learning: As alluded to above, traditional online meta-learning makes use of a limitless memory, retraining their model to perform well

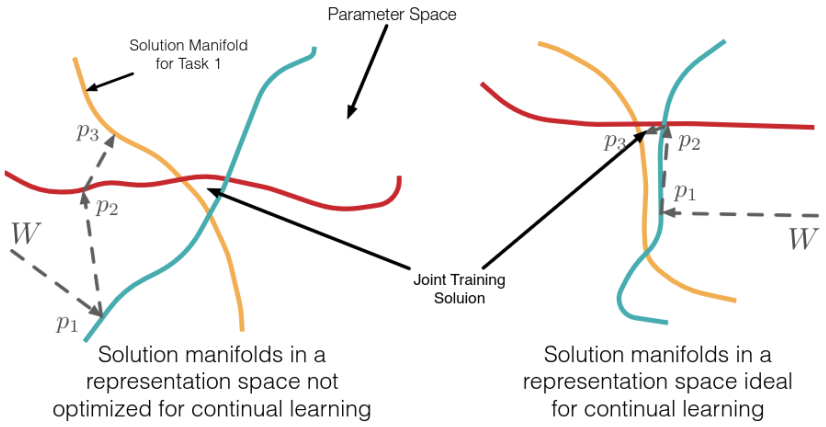

FIGURE 2.2: The difference in solutions for three different tasks with initial representation (left) and meta-learn optimized representation (right) [42]

on every tasks seen so far. In the continual learning setting, this quickly becomes infeasible as the number of tasks grows without bound. This limitation has resulted in a series of works on continual meta-learning [36, 18, 15]. The upshot of these ideas is to incorporate additional terms to the meta-loss function to ensure performance on an algorithm specific *continual objective*. Primarily these objectives consist of the following:

(1) *Mitigate Interference:* In just the multitask setting, it has been shown that tasks can have interfering gradients [32]. As it sounds, this means that optimizing towards the alternative objective results in optimizing in opposing gradient directions. As expected, this results in forgetting of previous tasks and even not learning new tasks. In [42, 29], they meta learn representations to explicitly deal with this issue. In Figure 2.2, we see the resulting solution manifolds. Namely, the dotted arrows represent the learning trajectory in an online continual setting. Without the meta learned representation, we see that solving for p_2 results in interference with p_1. However, after reparameterization, we have orthogonal and parallel manifolds. This mitigates interference and allows quick recall because the solutions learned are the closest to previous solutions. In [29], rather than meta learning a representation that is then passed to the fast adaptation network, they interleave the meta-parameters between layers. The difference in architectures can be seen in Figure 2.3. Additionally, WarpGrad is a more general meta optimization scheme, where backpropagation shows that the meta parameters are actually learned pre-conditioning matrices for gradient descent. This means given suitably well formed learned meta layers (form a Riemman metric), we inherit all of the theoretical guarantees of gradient descent. They also show that the optimal warp layers under their objective should facilitate task adaptation. It is an important note that they are not backpropagating through updates to the meta parameters themselves unlike in MAML [26]. They adapt WarpGrad for the continual learning setting by defining the meta loss to be the average loss over all seen tasks. Intuitively, the meta parameters would then be learned to also parametrize the solution space to prevent interference. An alternative approach to also mitigate catastrophic interference is proposed in [65]. Instead of learning a representation

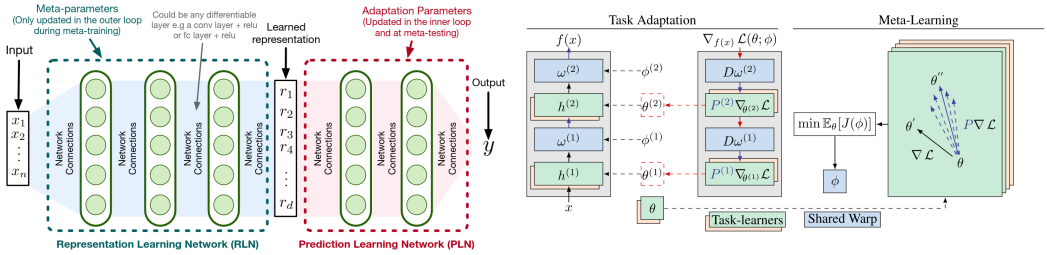

FIGURE 2.3: Illustration of OML [42] (left) and Warpgrad [29] (right) architectures

they learn a sampling scheme from the replay buffer to schedule tasks that do not interfere.

(2) *Faster Remembering:* An alternative objective to trying to retain expertise on previously seen tasks is instead to try to be able to quickly recover previous expertise. This is the approach taken in [38]. This is crucially important because it does not assume task labels, meaning that not only must it be able to quickly recover performance but also quickly identify the current task. Here they provide a framework of two modules: an encoder that encodes the context representation and a decoder that takes the context representation and maps it to a task specific model. Unlike the other works covered, this is entirely context detection. The hope is just to quickly identify which task you are in and then have a specialized model to quickly adapt to this task. To do this in the continual learning setting, they say that you can utilize the identification module with existing continual learning knowledge retention approaches [48].

2.3.2 Meta Reinforcement Learning

A relevant alternative problem is the meta reinforcement learning problem. In this problem, the goal is to meta learn parts or the entirety of the RL algorithm. What this means is that rather than trying to learn a policy directly from experience as in traditional RL, we instead want to learn an algorithm that will produce a policy given experience. The problem formulation also has flexibility. Some formulations allow for a warm-up phase to allow for MAML-like algorithms to have time to adapt. A formal definition that encapsulates most formulations is as follows. We want to maximize

$$\mathcal{J}(\theta) = \mathbb{E}_{M^i \sim p(\mathcal{M})} \left[\mathbb{E}_{\mathcal{D}} \left[\sum_{\tau \in \mathcal{D}_{K:H}} G(\tau) \middle| f_\theta, \mathcal{M}^i \right] \right]$$

Here, $G(\tau)$ is the discounted return in MDP $\mathcal{M}^i$, where H is the task horizon and K is the length of the warm up period. An MDP in this case is interchangeable with the previous use of tasks. Intuitively, this translates to finding an algorithm f that can quickly perform well across a distribution of tasks. A severe limitation of this is that we need this distribution of tasks a priori. To circumvent this requirement [35] proposed utilizing an information-theoretic based unsupervised RL approach to generate tasks. Then one can do meta RL on these unsupervised tasks.

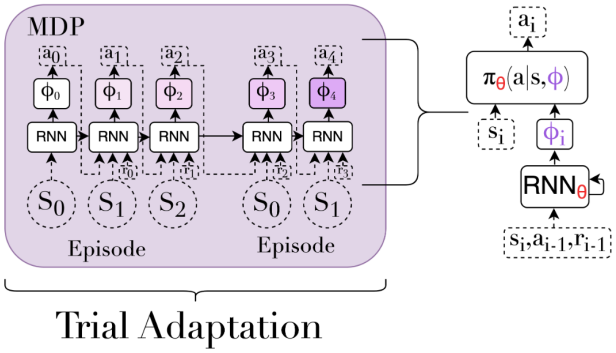

FIGURE 2.4: Graphic of RL^2 from [9]

MAML: This approach is very similar to traditional MAML[26] but in the reinforcement learning setting. In [83], they have an inner loop and outer loop. However, the inner loop is instead the n-step return of an actor critic agent in a reinforcement learning environment. The meta optimization acts as a sort of hyperparameter tuning, where they aim to meta-learn the discount factor γ and TD-λ hyperparameter λ based on the return in the inner loop. The inner loop can also be taken over a distribution of tasks to find "generally" good hyperparameters across the entire distribution as in the meta reinforcement learning formulation.

RL^2: An alternative approach is taken in [22]. Here, they instead try to meta learn the entire RL algorithm itself. This paper has a very similar motivation to ours. They too make the observation that humans are able to more quickly learn tasks by applying their lifetime of experience. To emulate this, they use a general RNN to model their policy. The RNN then has meta parameters trained in the outer loop for "slow" adaptation and parameters trained in the inner loop for fast task adaptation. The meta learned parameters are the hidden weights of the RNN. The RNN takes the entire history as input, meaning that it can dynamically change its policy based on the history. Ideally, the meta learned weights would learn some sort of context detection, which can then identify the current task and adjust the representation accordingly.

Option Discovery: Similar to [83] in [79] they consider non-myopic inner loop rollouts but rather than update hyperparameters they aim to find generally useful options. It has been shown that learning options or auxiliary tasks can greatly increase learning especially in sparse reward environments. Similar to the other meta RL algorithms, this suffers the drawback that you must have some distribution of tasks. In their paper, they define each task to be a different location in a grid world.

Chapter 3

Investigation

In this chapter, we explore the work of [80]. In addition to a open-source JAX-based implementation, we investigate the viability of the approach for continual reinforcement learning. We also propose a potential improvement for this setting via [28]. Finally, we discuss some of the failure modes and limitation of this algorithm. For potential solutions, we differ to the next chapter.

3.1 Background

Single Lifetime Meta RL: From the bitter lesson, we aspire for our algorithms to learn as much as possible on their own without having to explicitly encode any existing knowledge. To this effect, hyperparameters instigate unwanted human intervention. Optimal hyperparameters can take hundreds or thousands of trials to identify. Alternatively, there has been work that meta-learns a large majority of the hyperparameters of a reinforcement learning algorithm [86, 83]. In addition, there is work meta-learning supplementary components of an RL algorith, such as intrinsic rewards [87] or options [79]. Unlike traditional meta-reinforcement learning, these methods do not meta learn across a task distribution. Instead they perform an inner loop, where each iteration is a standard policy gradient update. They then in an outer loop update the meta parameters based on these inner loop updates in a single "lifetime". This is in contrast to meta reinforcement learning that learns over a task distribution. Objectives for the outer loop typically sum over some subset of RL losses from the inner loops.

Bootstrapped Meta Gradients: Traditional meta-learning involves updating the meta-parameters based on how they perform on the true loss function. Namely in the MAML objective, θ is updated based on the evaluation of θ' through the original loss $\mathcal{L}$. However, many times the loss landscape for $\mathcal{L}$ is undesirable. As alluded to above, the meta learning approach taken in [86, 83] uses the RL loss to update the meta parameters. However, it is very often the case that we are dealing with undesirable environments with sparse rewards. Ideally, we can decouple our meta updates from the current task to learn more general information about things, such as transition dynamics. In MAML [26], it was observed that longer rollouts can improve performance. However, this requires the computation of even higher order derivatives becoming computationally expensive. In BMG, they are able to both decouple the meta objective and make use of the information from longer rollouts without having to explicitly differentiate them. Namely, they decompose their rollout into two components: a parametrized component and a target component. The parametrized component acts as the function of our meta parameters and the target acts as a form of optimism to improve these parameters. Explicitly, we make K updates in our inner loop to our policy parameters θ. These updates depend on η

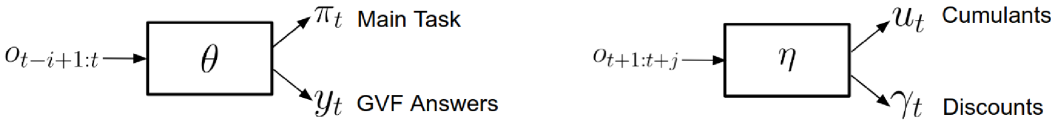

FIGURE 3.1: Overview of network architecture. Adapted from [80].

and give us $\theta^{(K)}(\eta)$. We then make another L updates that still depend on η, but we apply a stop gradient to these updates giving us $\theta^{(K+L)}$. We then update η via a "matching function" μ over these two rollouts i.e.

$$\min_{\eta} \mu(\theta^{(K)}(\eta), \theta^{(K+L)}) \implies \eta \leftarrow \eta - \nabla_{\eta}\mu(\theta^{(K)}(\eta), \theta^{(K+L)})$$

It is most common to take

$$\mu(\theta^{(K)}(\eta), \theta^{(K+L)}) = \mathrm{KL}(\pi_{\theta^{(K)}(\eta)} || \pi_{\theta^{(K+L)}})$$

General Value Functions (GVF): First introduced in [77], a General Value Function (GVF) is a straightforward generalization of the standard value function. A GVF $V^{c,\gamma}$ is the Bellman Optimal solution to the Bellman equation defined by the tuple (c, γ):

$$V^{c,\gamma}(s_t) = c(s_t) + \mathbb{E}\left[\gamma(s_{t+1})V^{c,\gamma}(s_{t+1})\right],$$

where $c, \gamma : \mathcal{S} \to \mathbb{R}$. An important note is that the discount factor γ can depend on the state. This implicitly allows for the determination of the end goal when $\gamma(s) = 0$.

Auxiliary Tasks: Humans, in addition to learning a main task, often learn intermediate or additional tasks that can help to aid the completion of the main task. The first question to arise when attempting to replicate such behavior is how to represent these tasks. The idea of auxiliary tasks was first introduced in [40]. Here they used handcrafted auxiliary tasks, where they added additional heads to the agent network to attempt to predict the next reward or pixels. More generally, these tasks can be represented as GVFs. In [80], they propose to meta learn the cumulant and discount factor in the GVF formulation via its contribution to learning the main task. It remains an open question as to when/why/what auxiliary tasks are useful [81].

3.2 Methods

Discovery of Useful Questions as Auxiliary Tasks [80]: The objective of this algorithm is to meta learn the cumulants and discount factors that define a GVF. To do this, they introduce an additional neural network known as the "question network". The question network generates both the cumulants and the discounts. Notably, the question network takes future observations as input. This can be easily seen in Figure 3.1. In their work they explore the benefits of these learned auxiliary tasks as (i) supplemental to the main task learning and (ii) to learn the representation in its entirety. We will explore the second case, which will be referred to as the representation learning setting. The agent has an encoder layer that learns a state representation that is passed to each of the actor, critic, and answer head. Formally, the agent network contains the parameters $\theta_{\text{agent}} = \{\theta_{\text{actor}}, \theta_{\text{critic}}, \theta_{\text{answer}}, \theta_{\text{rep}}\}$. For the

representation learning setting we define the loss functions as follows:

$$\mathcal{L}_{\text{inner}}(\theta_{\text{agent}}) = \mathcal{L}_{\text{PG}}(\theta_{\text{actor}}) + \mathcal{L}_{\text{V}}(\theta_{\text{critic}}) + \mathcal{L}_{\text{ENT}}(\theta_{\text{actor,critic}}) + \mathcal{L}_{\text{ANS}}(\theta_{\text{rep,answer}})$$
$$\mathcal{L}_{\pi} = \mathcal{L}_{\text{PG}} + \mathcal{L}_{\text{ENT}}$$
$$\mathcal{L}_{\text{outer}} = \sum_{k=1}^{K} \mathcal{L}_{\pi}(\theta_{t,k}) + \mathcal{L}_{\text{KL}}$$

Namely, θ_{rep} is omitted from each of the non-answer losses to ensure that the question network is driving representation learning. In the outer loop update, we add an additionally KL stability term not present in [80] but shown to improve performance in [86]. The term serves as a meta trust region regularization term.

Bootstrapped Meta Gradient Objective: For this algorithm we replace the meta loss shown above with the analagous bootstrapped meta loss. For the matching function μ, we use KL-divergence. The inner loss remains the same as above, but the outer loss becomes

$$\mathcal{L}_{\text{outer}} = \text{KL}(\pi_{\theta_{t+K}(\eta)} || \pi_{\theta_{t+K+L}}),$$

where $\theta_{t+K}(\eta)$ is the parameters of the agent after applying K inner loop updates to the agent and θ_{t+K+L} indicates another L updates. We omit η in the second set of parameters to indicate that we apply a stop gradient on these parameters.

Concatenated ReLU: Dying neurons are currently the leading suspect for most of the instances of plasticity loss. When the activation function is ReLU this is often due to negative outputs at each layer. To remedy this, in [1], they propose to utilize *Concatenated Rectified Linear Units* [74] to help mitigate this loss of plasticity. Rather than the traditional ReLU: $\text{ReLU}(x) = \max(0, x)$, CReLU considers both the positive and negative outputs by concatenating the respective ReLU outputs: $\text{CReLU}(x) = [\text{ReLU}(x),\ \text{ReLU}(-x)]$.

3.3 Experiments

Environment: We limit ourselves to the Mini-Atari domain, where we use the fully JAX-based implementations provided in [51]. The only working implementation was Breakout, so this will be our toy test bed.

Architecture: We do not flatten observations and instead use a CNN-based architecture for the representation network. As in the original paper, both the agent and question network have identical representation network architectures. The question networks uses the next ten observations as input and the agent network uses the previous four. The representation network is three identical CNN layers followed by an MLP. The exact hyperparameters and architecture we used will be provided in Appendix A.

In this work, we aim to investigate and address a few of the shortcomings of the algorithm that are particularly important as we try to extend to

1. *Non-Stationary:* The answer network target undergoes severe non-stationary. This means that even in the standard reinforcement learning setting, we face many of the similar challenges as in the continual setting. Can explicitly addressing this non-stationary improve stability?

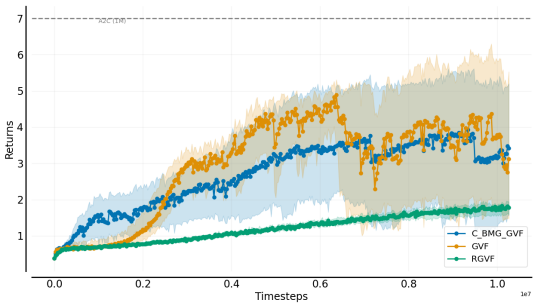

FIGURE 3.2: Average return over 10 seeds of C_BMG, RGVF, and GVF.

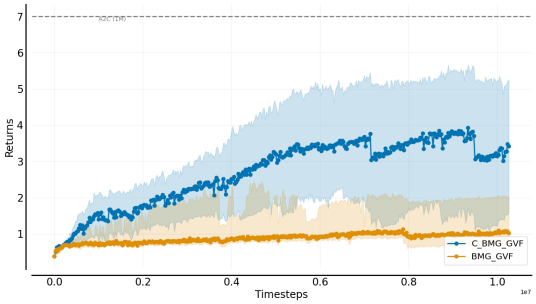

FIGURE 3.3: Average returns over 10 seeds of Continual BMG algorithm with CReLU vs. standard BMG.

2. *Coupled Objective:* Despite being a meta objective, the summed RL loss proposed in [80] is still deeply coupled to the current task. While this results in some ability to learn more general auxiliary tasks, the efficacy of the learned tasks is deeply dependent on the task with which they were discovered. We propose a Bootstrapped Meta Gradient [28] alternative to circumvent this issue.

Non-Stationary: We utilize *Concatenated Relu* (CReLU) units to maintain plasticity in the representation network. Since the CReLU units double the number of parameters in the following layers, we half all of the effected layers' parameters for fair comparison. To see that this is a fair comparison, we show the CReLU version of our A2C baseline performs approximately the same as the non-CReLU version in Appendix A in Figure A.1. In Figure 3.5, we see that this drastically decreases the percentage of dead units. In Figure 3.3, we see that CReLU is necessary to achieve reasonable performance with BMG.

Coupled Objective: Ultimately, we want to see if we can match the performance of the traditional MG algorithm using a BMG objective. This would in theory allow us to learn a decoupled representation for improved generalization. In Figure 3.2, we see that we are roughly able to match the this performance when using CReLU activations. Unfortunately, we did not see a favorable increase in successive meta-gradient cosine similarity as in [28]. An additional benefit to the BMG algorithm is that it is more memory efficient as we only backpropagated through a 10 step rollout as opposed to a twenty step rollout in the original algorithm. Future work will be to see whether the learned representation is indeed more general than the task-coupled original objective.

3.4 Limitations

In the previous section, we see that we are unable to match the performance of an end-to-end agent asymptotically. Here we propose a few reasons why this may be the case.

Training Instability: One implementation subtlety is that of minibatches. Analogous to the supervised learning setting, it has become standard to use PPO [73], sampling a large batch of trajectories and performing a sequence of intermediate gradient updates on minibatch subsets. This is possible due to the stability of the

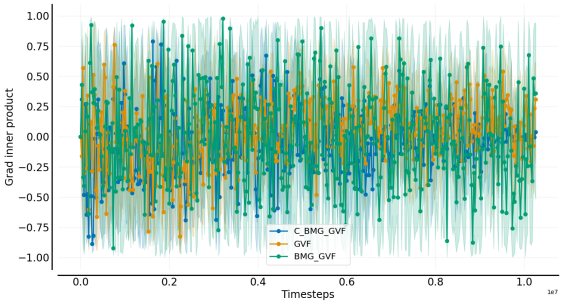

FIGURE 3.4: Normalized inner product between successive meta-gradients.

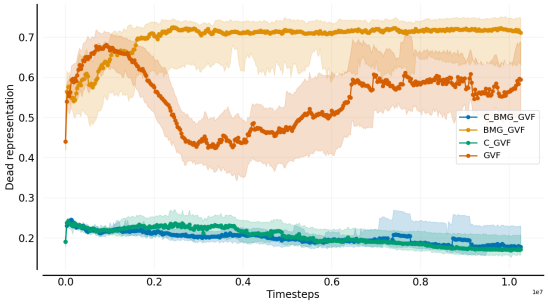

FIGURE 3.5: Percentage of dead representation units for Concatenated ReLU and original versions of MG and BMG.

pseudo-trust-region. The additional gradient updates have been shown to improve convergence [60]. Additionally, the same data can be reused for multiple epochs to further increase sample efficiency [6]. With the introduction of non-myopic meta-gradients, the meta objective requires re-interacting the environment upon each step in the unroll phase to approximate the true on-policy gradient of the updated policies. A somewhat related issue occurs the asynchronous setting with multiple actors [23, 86]. Here the learner updates are on data that was not generated by the current learner policy. Their workaround is to use importance sampling to approximate the true policy gradient. This idea seems promising, but it becomes impossible to compute the meta-gradient in this setting. The reason for this is because the computation of the cumulant target relies on the temporal structure from the trajectories within a batch. When we sample minibatches this temporal structure is broken, meaning that we can perform a k-step off-policy rollout for the inner learner. However, we still cannot perform multiple meta-gradient updates within the batch. Due to the inability to implement these stabilizing optimizations, the algorithm becomes far more susceptible to hyperparameters, such as learning rate and batch size. An even more niche detail is that if we naively use a standard PPO implementation then minibatch updates are treated as standard gradient updates in the inner loop of the meta loss. This means that n minibatches and an unroll length of k is effectively an unroll length of nk in terms of memory usage for the intermediate parameters. This means that the inner learner must also be a less stable algorithm such as A2C.

Hyperparameters: It has recently been shown that reinforcement learning hyperparameters fail to generalize well across environments [61, 62]. While this may seem unimportant for most applications, clearly, it is a substantial limitation of hyperparameter dependent algorithms in a continual reinforcement learning setting. The algorithm proposed in [80] has a dependence on many sensitive hyperparameters i.e. unroll length, number of questions, etc. This makes it extremely unlikely that it can be a viable solution in the continual learning setting. Beyond the continual learning setting, even in a stationary minigrid, it was very difficult to find a set of working hyperparameters. Using CARBS [25], we conducted a 200 run hyperparameter sweep in an empty gridworld [59]. Even in the most simple setting we see significant variation across hyperparameters in Figure 3.6.

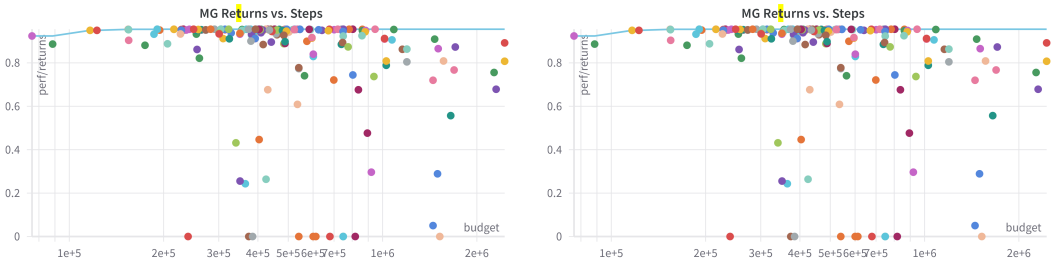

FIGURE 3.6: Final returns of MG (left) and BMG (right) for 200 different hyperparameter configurations.

3.5 Discussion

While meta-gradient methods may seem like a promising avenue for continual learning, it still remains a challenge to get them to work well in practice. Many of the tricks required are not conducive to generalization to new tasks. We aimed to remedy two of these issues via Bootstrapped Meta Gradients and Concatenated ReLU. However, the results leave a lot to be desired. Along with many of the limitations discussed previously, we list a few other possible reasons we were unable to reproduce the results.

Reproducibility: In addition to no reference code, in the initial paper, they provide very limited insight into many of the hyperparameters used to produce their results. A few that we found particularly important were the number of subsequent observations seen by both the agent and the question network. Although, they were briefly mentioned in the paper, there is no reference to them beyond their initial introduction. Additionally, there is no mention of the size of the networks used to conduct their experiments. We were unable to replicate results in the more sparse grid world setting. In the original paper, they alluded to the fact that they did not learn discounts in this setting but instead "handcrafted" discounts. It was unclear what they meant by this, possibly leading to our inability to reproduce their results.

Future Work: In the future, we intend to remedy some of these issues along with introducing the ability to scale across devices. With this, we could then begin to reproduce / compare the algorithm results in the original Atari domain and begin to tackle more challenging continual domains.

In the next chapter, we will elaborate more on some of the challenges that face learning GVFs in the continual setting. We will also begin to formulate some approaches to try to tackle these issues.

Chapter 4

Conclusion

In this work, we aimed to explore the viability of meta learning for continual reinforcement learning. In particular, we hoped to meta learn general useful structure that can transfer across tasks. In the first chapter, we conducted an extensive literature review across continual and meta learning. We provided much of the necessary background to understand the landscape of the field. In the second chapter, we investigated existing work to discover useful questions for General Value Functions [80]. We found that there are many issues preventing the use of this method for the more challenging continual setting. We proposed several modifications to increase stability and hopefully viability for more challenging domains.

Throughout the rest of this chapter, we summarize the unresolved issues for the continual setting, facing the proposed method. Ultimately, I was not able to explore these issues in depth due to the constraints of this project. However, I was able to think about potential starting points for future work and believe this is a good place to document them. Additionally, this is a good avenue to receive feedback and help others get started to work on these problems.

4.1 Background

Successor Representation: As a precursor, the successor representation has been around for decades [20]. The successor representation aims to model the structure of the transition dynamics of the environment. Intuitively, it is a GVF, where the discount factor is the same as the environment and the cumulant is the indicator for state occupancy $\mathbf{1}_{\{s_{t+1}=\tilde{s}\}}(s)$ i.e.

$$M^{\pi}(s,\tilde{s}) = \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t \mathbf{1}_{\{s_{t+1}=\tilde{s}\}} \mid s_0 = s\right]$$

The key idea here is that they decompose the value function into reward and task structure. This is actually easily seen explicitly as

$$\begin{aligned} V(s) &= \sum_{\tilde{s}} M(s,\tilde{s}) R(\tilde{s}) \\ &= \boldsymbol{M}(s)^T \boldsymbol{R}(s), \end{aligned}$$

where the last equality vectorizes the successor representation across all states and is of dimension $\mathcal{S}$.

Successor Features: One quickly realizes as the state space becomes increasingly large it is not feasible for your representation to have the same dimension as the number of states. Instead, we want to compress this representation to just focus on

the features that are shared across all states. This is the idea of successor features [7]. We replace the indicator for the state occupancy with a general cumulant $\boldsymbol{\phi} \in \mathbb{R}^d$ for $d \leq |\mathcal{S}|$. We then have the corresponding definition

$$\boldsymbol{\psi}^{\pi}(s) = \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \boldsymbol{\phi}\left(s_{t+1}\right) \mid s_0 = s\right]$$

This is a strict generalization of the successor representation as we can take $\boldsymbol{\phi}$ as the indicator above. The real power of successor features is encapsulated in their transfer ability Namely, if we have a low rank linear MDP i.e. $\langle \boldsymbol{\phi}(s), w \rangle = r(s)$ for some $\boldsymbol{w} \in \mathbb{R}^d$ with $d \ll |\mathcal{S}|$. If this is the case then we inherit the exact desirable properties from the successor representation:

$$\begin{aligned} V^{\pi}(s) &= \mathbb{E}\left[\sum_{t=0}^{\infty} \gamma^t \langle \boldsymbol{\phi}\left(s_{t+1}\right), \mathbf{w} \rangle \mid s_0 = s\right] \\ &= \langle \boldsymbol{\psi}^{\pi}(s), \mathbf{w} \rangle \end{aligned}$$

Intuitively, this allows us to learn a "sufficient" representation of the transition dynamics of the environment, which we can then reuse across tasks. Note that successor features also have a Bellman equation

$$\boldsymbol{\psi}^{\pi}(s_t, a_t) = \boldsymbol{\phi}(s_t) + \gamma \mathbb{E}_{d_{\pi}}\left[\boldsymbol{\psi}(s_{t+1})\right]$$

We also have the analagous control version

$$\boldsymbol{\psi}^{\pi}(s_t, a_t) = \boldsymbol{\phi}(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim d, a_{t+1} \sim \pi}\left[\boldsymbol{\psi}(s_{t+1}, a_{t+1})\right]$$

Universal Successor Feature Approximator: An important caveat to the above discussion is that successor features are policy dependent. Since they are update using TD-learning, the next state distribution depends on the current policy. One can use the successor feature equivalent of Q-learning for learning the successor features of the optimal policy for a given task $\boldsymbol{w}$ as follows

$$\boldsymbol{\psi}^{w}(s_t, a_t) = \boldsymbol{\psi}^{w}(s_t, a_t) + \alpha\left[\boldsymbol{\phi}(s_t) + \gamma \boldsymbol{\psi}^{w}(s_{t+1},) - \boldsymbol{\psi}^{w}(s_t, a_t)\right]$$

However, this completely defeates the purpose for forward transfer discussed above. Due to this, there is the Universal Successor Feature Approximator [12] (USFA). Similar to its Universal Value Function Approximator [69] (UVFA) counterpart that takes a goal, the USFA takes a task as input. The resulting successor feature's Bellman equation can be written as

$$\boldsymbol{\psi}(s_t, a_t, \boldsymbol{w}) = \boldsymbol{\phi}(s_t) + \gamma \boldsymbol{\psi}(s_{t+1}, \arg\max_{a} \langle \boldsymbol{\psi}(s_{t+1}, a, \boldsymbol{w}), \boldsymbol{w} \rangle, \boldsymbol{w})$$

This now allows us to immediately compute the new Q function for any new task $\boldsymbol{w}$. Recently, there has been an extensive review on successor features and we differ you to their paper for more details [17].

4.2 Future Work

In this section, we consider many of the limitations encountered while conducting this thesis. In doing so, we provide extensive coverage of what the problems are and

how they arise in ours setting of continual RL. We then propose a starting point for potential remedies to each of these problems and highlight existing promising work towards these goals.

4.2.1 Continual Meta-Learning in RL:

Problem: Earlier in this thesis, we introduced and discussed some of the approaches for continual meta learning [18, 36]. However, these methods are not easily transferrable to the reinforcement learning setting. An underlying assumption made by each of these methods is access to the sequence of loss functions $(\ell_1, \ldots, \ell_n)$ that make up each task seen so far. Explicitly, these methods provide alternative objectives to optimize the following continual learning objective provided in [65]:

$$\min_{\theta_0^j} \left(\sum_{i=1}^{t} \left(\ell_i \left(\theta_0^j \right) \right) - \alpha \sum_{p,q \leq t} \left(\frac{\partial \ell_p \left(\theta_0^j \right)}{\partial \theta_0^j} \cdot \frac{\partial \ell_q \left(\theta_0^j \right)}{\partial \theta_0^j} \right) \right)$$

Here each ℓ_i is the a loss function for task i. Intuitively, this utilizes the cosine similarity of the gradients as a regularization term to minimize catastrophic interference. This regularization term also enables forward transfer In the supervised learning setting this is not too restrictive, one can just store previous input-output pairs $\{(X_i, Y_i)\}_{i=1}^{N}$ seen in previous tasks. In the reinforcement learning setting, access to a tasks loss function is access to the task environment itself. As in [56], in off-policy value based algorithms, we can inherit supervised learning techniques. In [65], they do this to meta learn an experience replay sampling scheme that aims to minimize interference and maximize transfer between samples. However, this does not transfer to the policy gradient setting, where we need on-policy data.

Future Work: In [11], they perform continual meta learning in the RL setting. To do so, they utilize ideas from behavior cloning. Namely, they store a policy and trajectory corresponding to the best performance on each of the previous tasks. They then aim to optimize the behavior cloning objective for a policy that minimizes the loss after one gradient update. Explicitly,

$$\min_{\theta} \sum_{\mathcal{T}_i} \sum_{\mathcal{T}_i^v \sim \mathcal{D}_{0:i}^*} \mathbb{E}_{\mathcal{T}_i} \left[\mathcal{L}_{BC} \left(\theta + \alpha \nabla_\theta J_i(\theta), \mathcal{D}_i^v \right) \right], \mathcal{L}_{BC} \left(\theta_i, \mathcal{D}_i \right) = - \sum_{(s_t, a_t) \in \mathcal{D}_i} \log \pi \left(a_t \mid s_t, \theta_i \right).$$

More intuitively, they want to learn a policy that most quickly can recover on all of the previously seen tasks. This idea does not directly transfer to the more general setting, where we want to learn about things like cumulants that do not directly parametrize the policy. However, I think the idea of storing a trajectory as a "task representative" and then adapting the ideas from the continual supervised learning setting can be a promising avenue of future work. Additionally, we can employ this objective as a regularization term to a BMG type bootstrapped objective to incorporate the cumulants into the computation graph and encourage meta updates to find solutions that minimize interference.

4.2.2 Continual Auxiliary Task Learning:

Problem: In the previous chapter, we investigated discovering the cumulants in the auxiliary task problem. An important point is that, given we are able to continually learn new cumulants for new tasks, we still need to be able to continually track these changing cumulants when learning the GVFs. A solution to this was proposed

in [5]. Here they decompose a value function into a permanent and a transient component. The permanent component aims to capture general structure that can be shared across different tasks. The transient component aims to correct the permanent component for a given task. A key limitation of this work is that the weights of the two components are disjoint. This makes it unclear how to apply this idea into the representation learning setting. It is of independent interest whether learning a "fast-slow" or "long-term-short-term" representation is a promising avenue.

Future Work: Successor features are a very promising future direction for this. The idea has already been proposed in [55]. However, the successor feature representation aims to decouple the task from the underlying learned features. Assuming an underlying good learned successor feature representation, then under a task change, one can immediately compute the new GVF by taking the dot product with the existing representation. It was recently shown that the use of many structured random successor features drastically improves performance [24]. Given the same ability to scale, one could instead learn USFA's [12] and learn the tasks w via regression as in [55] to hopefully replicate the results in the continual setting. However, in [55] they only consider the case where the cumulants are given and the task is changing. However, we have to track these non-stationary cumulants when learning our successor features. In [16], they show that this is actually possible by applying the PMF trick from [72] to learn the successor features.

4.2.3 Auxiliary Tasks as Structured Representations

Problem: Previously, we explored the algorithm proposed in [80]. Namely, they learn "useful" GVFs through meta-learning the cumulants and the discounts with respect to the meta objective of the summed RL loss. A major limitation of the algorithm in the continual RL settings is that these GVFs are learned on policy. This means that the "answers" learned to these questions only answer questions about the current policy. For example, in Figure 4.1 suppose that the current policy is to walk forward. In this case, even if we perfectly learn the cumulants to indicate whether we see each color then our GVFs would be zero for every GVF except the color of the wall directly in front of us. It is clear that this is not a very useful representation. To remedy this, it requires that we are able to learn off policy

Future Work: The first decision we must make is what policy this off policy learning system should follow. The natural choice is to learn the value function according to the optimal policy for a given task. Going back to Figure 4.1, this would encode information like "If I were to follow the shortest path to the yellow wall how long would it take to reach it?". Already this is much more useful information than before. In the deep learning case where we are making use of severe function approximation learning off policy TD-estimates is far less stable [37, 76]. Naively applying importance sampling drastically increases variance and can be a new cause of instability. In [76], they propose the idea of emphatic weighting, where they reduce the variance by essentially truncating the importance sampling to the start of an "excursion". This was shown to work in the deep reinforcement learning setting [43, 86, 23]. Later works have extended this by also employing an "interest" function that weighs how much interest we have in ensuring that we have a good approximation at a given state [55, 34].

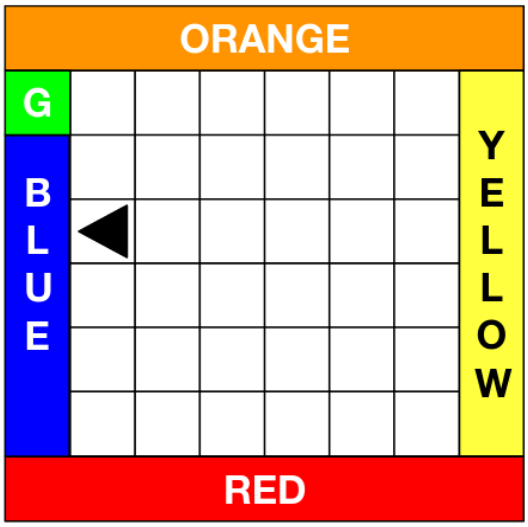

FIGURE 4.1: Adapted from [70]. The Compass World: The agent can only observe a color if it is at the wall and facing outwards. The agent depicted as an arrow would see Blue. In the middle of the world, the agent sees White. Goal: The agent's goal is to make accurate predictions about which direction it is facing

Appendix A

Additional Implementation Details

All of the code will be open-sourced and publicly available. It is entirely implemented in JAX [13] and can serve as a modular framework for future RL developement. I want to thank the following open source projects for providing great inspiration (and sometimes code) to greatly simplify my life throughout this project. In order to provide support for multiple JAX environments, we used the Jumanji wrappers provided by Stoix [78]. As inspiration for the modular codebase structure we used Navix [63]. For statistically correct plots we used RLLiable [3] and some helpful plotting code from [33]. For managing configurations and providing command line support, we used Hydra [85].

A.1 Additional Experiments

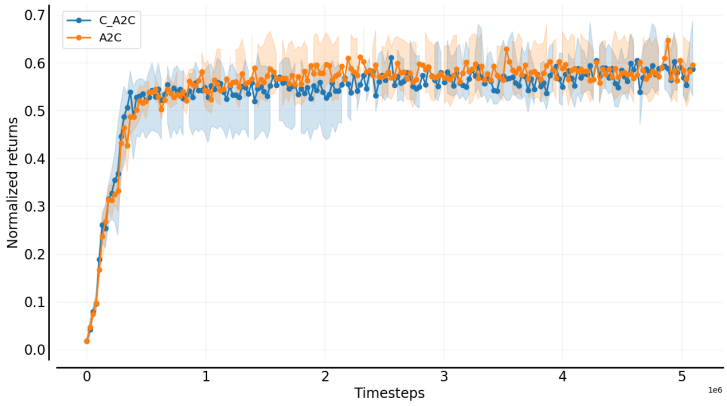

FIGURE A.1: Comparison of base A2C algorithm when using ReLU vs. CReLU

A.2 Hyperparameters

Original Meta Gradient	
Inner Learner	
Optimiser	RMSProp
RMSProp β	0.99
Learning rate	10^{-4}
Num Envs	16
Num Steps	16
γ	0.99
MLP feature size (V, π)	256
Previous Observations	2
Meta-learner	
Optimiser	Adam
ϵ (Adam)	10^{-3}
β_1, β_2	$0.9, 0.99$
Learning rate	7×10^{-3}
MLP feature size (c, γ)	256
Unroll Length	20
Number of Questions	128
Discount Output Activation	Sigmoid
Cumulant Output Activation	Tanh
Future Observations	2

TABLE A.1: Original meta gradient hyperparameters

Bootstrapped Meta Gradient	
Inner Learner	
Optimiser	RMSProp
RMSProp β	0.99
Learning rate	10^{-4}
Num Envs	4
Num Steps	16
γ	0.99
μ	$\mathrm{KL}\left(\pi_{\tilde{x}} \| \pi_{x'}\right)$
MLP feature size (V, π)	256
Previous Observations	2
Meta-learner	
Optimiser	Adam
ϵ (Adam)	10^{-3}
β_1, β_2	$0.9, 0.99$
Learning rate	7×10^{-3}
MLP feature size (c, γ)	256
Unroll Length	10
Bootstrap Length	10
Number of Questions	128
Discount Output Activation	Sigmoid
Cumulant Output Activation	Tanh
Future Observations	2

TABLE A.2: Bootstrapped meta gradient hyperparameters

Parameter	Representation Network	CReLU Representation Network
convolutions in block	1	1
channels	32	16
kernel sizes	(3, 3)	(3,3)
kernel strides	(1,1)	(1, 1)
mlp layers	1	1
head hiddens	256	128
activation	ReLU	CReLU

Bibliography

[1] Zaheer Abbas et al. *Loss of Plasticity in Continual Deep Reinforcement Learning*. 2023. arXiv: 2303.07507 [cs.LG].

[2] David Abel et al. "A definition of continual reinforcement learning". In: *Advances in Neural Information Processing Systems* 36 (2024).

[3] Rishabh Agarwal et al. "Deep Reinforcement Learning at the Edge of the Statistical Precipice". In: *Advances in Neural Information Processing Systems* (2021).

[4] Everton L. Aleixo et al. *Catastrophic Forgetting in Deep Learning: A Comprehensive Taxonomy*. 2023. arXiv: 2312.10549 [cs.LG].

[5] Nishanth Anand and Doina Precup. "Prediction and Control in Continual Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 63779–63817. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/c94bbbef466ab1b2cfa100e41413b3a8-Paper-Conference.pdf.

[6] Marcin Andrychowicz et al. *What Matters In On-Policy Reinforcement Learning? A Large-Scale Empirical Study*. 2020. arXiv: 2006.05990 [cs.LG]. URL: https://arxiv.org/abs/2006.05990.

[7] Andre Barreto et al. "Successor Features for Transfer in Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. Ed. by I. Guyon et al. Vol. 30. Curran Associates, Inc., 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/350db081a661525235354dd3e19b8c05-Paper.pdf.

[8] Andrew G Barto and Thomas G Dietterich. "Reinforcement learning and its relationship to supervised learning". In: *Handbook of learning and approximate dynamic programming* 10 (2004), p. 9780470544785.

[9] Jacob Beck et al. *A Survey of Meta-Reinforcement Learning*. 2023. arXiv: 2301.08028 [cs.LG].

[10] Richard Bellman. "The theory of dynamic programming". In: *Bulletin of the American Mathematical Society* 60.6 (1954), pp. 503–515.

[11] Glen Berseth et al. *CoMPS: Continual Meta Policy Search*. 2021. arXiv: 2112.04467 [cs.LG].

[12] Diana Borsa et al. *Universal Successor Features Approximators*. 2018. arXiv: 1812.07626 [cs.LG]. URL: https://arxiv.org/abs/1812.07626.

[13] James Bradbury et al. *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. 2018. URL: http://github.com/google/jax.

[14] Noam Brown and Tuomas Sandholm. "Superhuman AI for multiplayer poker". In: *Science* 365.6456 (2019), pp. 885–890.

[15] Massimo Caccia et al. "Online Fast Adaptation and Knowledge Accumulation (OSAKA): a New Approach to Continual Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 16532–16545. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/c0a271bc0ecb776a094786474322cb82-Paper.pdf.

[16] Wilka Carvalho et al. *Combining Behaviors with the Successor Features Keyboard*. 2023. arXiv: 2310.15940 [cs.AI]. URL: https://arxiv.org/abs/2310.15940.

[17] Wilka Carvalho et al. *Predictive representations: building blocks of intelligence*. 2024. arXiv: 2402.06590 [cs.AI]. URL: https://arxiv.org/abs/2402.06590.

[18] Arslan Chaudhry et al. "Efficient Lifelong Learning with A-GEM". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=Hkf2_sC5FX.

[19] John D Co-Reyes et al. "Accelerating online reinforcement learning via model-based meta-learning". In: *Learning to Learn-Workshop at ICLR 2021*. 2021.

[20] Peter Dayan. "Improving Generalization for Temporal Difference Learning: The Successor Representation". In: *Neural Computation* 5.4 (1993), pp. 613–624. DOI: 10.1162/neco.1993.5.4.613.

[21] Shibhansh Dohare et al. "Maintaining plasticity in deep continual learning". In: *arXiv preprint arXiv:2306.13812* (2023).

[22] Yan Duan et al. "Rl 2: Fast reinforcement learning via slow reinforcement learning". In: *arXiv preprint arXiv:1611.02779* (2016).

[23] Lasse Espeholt et al. *IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures*. 2018. arXiv: 1802.01561 [cs.LG]. URL: https://arxiv.org/abs/1802.01561.

[24] Jesse Farebrother et al. "Proto-Value Networks: Scaling Representation Learning with Auxiliary Tasks". In: *The Eleventh International Conference on Learning Representations*. 2023. URL: https://openreview.net/forum?id=oGDKSt9JrZi.

[25] Abraham J. Fetterman et al. *Tune As You Scale: Hyperparameter Optimization For Compute Efficient Training*. 2023. arXiv: 2306.08055 [cs.LG]. URL: https://arxiv.org/abs/2306.08055.

[26] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-agnostic meta-learning for fast adaptation of deep networks". In: *International conference on machine learning*. PMLR. 2017, pp. 1126–1135.

[27] Chelsea Finn et al. "Online meta-learning". In: *International conference on machine learning*. PMLR. 2019, pp. 1920–1930.

[28] Sebastian Flennerhag et al. *Bootstrapped Meta-Learning*. 2022. arXiv: 2109.04504 [cs.LG].

[29] Sebastian Flennerhag et al. "Meta-learning with warped gradient descent". In: *arXiv preprint arXiv:1909.00025* (2019).

[30] Sebastian Flennerhag et al. "Temporal difference uncertainties as a signal for exploration". In: *arXiv preprint arXiv:2010.02255* (2020).

[31] Robert French. "Catastrophic interference in connectionist networks: Can It Be predicted, can It be prevented?" In: *Advances in Neural Information Processing Systems*. Ed. by J. Cowan, G. Tesauro, and J. Alspector. Vol. 6. Morgan-Kaufmann, 1993. URL: https://proceedings.neurips.cc/paper_files/paper/1993/file/28267ab848bcf807b2ed53c3a8f8fc8a-Paper.pdf.

[32] Robert M French. "Using semi-distributed representations to overcome catastrophic forgetting in connectionist networks". In: *Proceedings of the 13th annual cognitive science society conference*. Vol. 1. 1991, pp. 173–178.

[33] Rihab Gorsane et al. "Towards a Standardised Performance Evaluation Protocol for Cooperative MARL". In: *arXiv preprint arXiv:2209.10485* (2022).

[34] Eric Graves et al. *Off-Policy Actor-Critic with Emphatic Weightings*. 2023. arXiv: 2111.08172 [cs.LG]. URL: https://arxiv.org/abs/2111.08172.

[35] Abhishek Gupta et al. *Unsupervised Meta-Learning for Reinforcement Learning*. 2020. arXiv: 1806.04640 [cs.LG].

[36] Gunshi Gupta, Karmesh Yadav, and Liam Paull. "Look-ahead Meta Learning for Continual Learning". In: *Advances in Neural Information Processing Systems*. Ed. by H. Larochelle et al. Vol. 33. Curran Associates, Inc., 2020, pp. 11588–11598. URL: https://proceedings.neurips.cc/paper_files/paper/2020/file/85b9a5ac91cd629bd3afe396ec07270a-Paper.pdf.

[37] Hado van Hasselt et al. *Deep Reinforcement Learning and the Deadly Triad*. 2018. arXiv: 1812.02648 [cs.AI]. URL: https://arxiv.org/abs/1812.02648.

[38] Xu He et al. *Task Agnostic Continual Learning via Meta Learning*. 2019. arXiv: 1906.05201 [stat.ML].

[39] Timothy Hospedales et al. *Meta-Learning in Neural Networks: A Survey*. 2020. arXiv: 2004.05439 [cs.LG].

[40] Max Jaderberg et al. "Reinforcement Learning with Unsupervised Auxiliary Tasks". In: *International Conference on Learning Representations*. 2017. URL: https://openreview.net/forum?id=SJ6yPD5xg.

[41] Khurram Javed and Richard S. Sutton. "The Big World Hypothesis and its Ramifications for Artificial Intelligence". In: *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*. 2024. URL: https://openreview.net/forum?id=Sv7DazuCn8.

[42] Khurram Javed and Martha White. *Meta-Learning Representations for Continual Learning*. 2019. arXiv: 1905.12588 [cs.LG].

[43] Ray Jiang et al. "Emphatic Algorithms for Deep Reinforcement Learning". In: *Proceedings of the 38th International Conference on Machine Learning*. Ed. by Marina Meila and Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 5023–5033. URL: https://proceedings.mlr.press/v139/jiang21j.html.

[44] John Jumper et al. "Highly accurate protein structure prediction with AlphaFold". In: *nature* 596.7873 (2021), pp. 583–589.

[45] Khimya Khetarpal et al. *POMRL: No-Regret Learning-to-Plan with Increasing Horizons*. 2022. arXiv: 2212.14530 [cs.AI].

[46] Khimya Khetarpal et al. *Towards Continual Reinforcement Learning: A Review and Perspectives*. 2022. arXiv: 2012.13490 [cs.LG].

[47] Khimya Khetarpal et al. "Towards continual reinforcement learning: A review and perspectives". In: *Journal of Artificial Intelligence Research* 75 (2022), pp. 1401–1476.

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

[49] Saurabh Kumar, Henrik Marklund, and Benjamin Van Roy. *Maintaining Plasticity in Continual Learning via Regenerative Regularization*. 2023. arXiv: 2308.11958 [cs.LG].

[50] Saurabh Kumar et al. *Continual Learning as Computationally Constrained Reinforcement Learning*. 2023. arXiv: 2307.04345 [cs.LG].

[51] Robert Tjarko Lange. *gymnax: A JAX-based Reinforcement Learning Environment Library*. Version 0.0.4. 2022. URL: http://github.com/RobertTLange/gymnax.

[52] Alex Lewandowski et al. *Directions of Curvature as an Explanation for Loss of Plasticity*. 2024. arXiv: 2312.00246 [cs.LG].

[53] Clare Lyle et al. *Understanding plasticity in neural networks*. 2023. arXiv: 2303.01486 [cs.LG].

[54] Michael McCloskey and Neal J Cohen. "Catastrophic interference in connectionist networks: The sequential learning problem". In: *Psychology of learning and motivation*. Vol. 24. Elsevier, 1989, pp. 109–165.

[55] Matthew McLeod et al. *Continual Auxiliary Task Learning*. 2022. arXiv: 2202.11133 [cs.LG]. URL: https://arxiv.org/abs/2202.11133.

[56] Volodymyr Mnih et al. "Human-level control through deep reinforcement learning". In: *Nature* 518.7540 (Feb. 2015), pp. 529–533. ISSN: 00280836. URL: http://dx.doi.org/10.1038/nature14236.

[57] Volodymyr Mnih et al. *Playing Atari with Deep Reinforcement Learning*. 2013. arXiv: 1312.5602 [cs.LG].

[58] Evgenii Nikishin et al. *Deep Reinforcement Learning with Plasticity Injection*. 2023. arXiv: 2305.15555 [cs.LG].

[59] Alexander Nikulin et al. "XLand-MiniGrid: Scalable Meta-Reinforcement Learning Environments in JAX". In: *Intrinsically-Motivated and Open-Ended Learning Workshop, NeurIPS2023*. 2023. URL: https://openreview.net/forum?id=xALDC4aHGz.

[60] Johan Obando Ceron, Marc Bellemare, and Pablo Samuel Castro. "Small batch deep reinforcement learning". In: *Advances in Neural Information Processing Systems*. Ed. by A. Oh et al. Vol. 36. Curran Associates, Inc., 2023, pp. 26003–26024. URL: https://proceedings.neurips.cc/paper_files/paper/2023/file/528388f1ad3a481249a97cbb698d2fe6-Paper-Conference.pdf.

[61] Johan Obando-Ceron et al. *On the consistency of hyper-parameter selection in value-based deep reinforcement learning*. 2024. arXiv: 2406.17523 [cs.LG]. URL: https://arxiv.org/abs/2406.17523.

[62] Andrew Patterson et al. *The Cross-environment Hyperparameter Setting Benchmark for Reinforcement Learning*. 2024. arXiv: 2407.18840 [cs.LG]. URL: https://arxiv.org/abs/2407.18840.

[63] Eduardo Pignatelli et al. "NAVIX: Scaling MiniGrid Environments with JAX". In: *arXiv preprint arXiv:2407.19396* (2024).

[64] Jathushan Rajasegaran, Chelsea Finn, and Sergey Levine. "Fully online meta-learning without task boundaries". In: *arXiv preprint arXiv:2202.00263* (2022).

[65] Matthew Riemer et al. *Learning to Learn without Forgetting by Maximizing Transfer and Minimizing Interference*. 2019. arXiv: 1810.11910 [cs.LG].

[66] Mark Ring. *Representing Knowledge as Predictions (and State as Knowledge)*. 2021. arXiv: 2112.06336 [cs.AI].

[67] Mark Bishop Ring. *Continual learning in reinforcement environments*. The University of Texas at Austin, 1994.

[68] Tom Schaul et al. *Ray Interference: a Source of Plateaus in Deep Reinforcement Learning*. 2019. arXiv: 1904.11455 [cs.LG].

[69] Tom Schaul et al. "Universal Value Function Approximators". In: *Proceedings of the 32nd International Conference on Machine Learning*. Ed. by Francis Bach and David Blei. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, 2015, pp. 1312–1320. URL: https://proceedings.mlr.press/v37/schaul15.html.

[70] Matthew Schlegel et al. "General Value Function Networks". In: *Journal of Artificial Intelligence Research* 70 (Jan. 2021), 497–543. ISSN: 1076-9757. DOI: 10.1613/jair.1.12105. URL: http://dx.doi.org/10.1613/jair.1.12105.

[71] Juergen Schmidhuber, Jieyu Zhao, and Marco Wiering. *Simple Principles of Metalearning*. Tech. rep. 1996.

[72] Julian Schrittwieser et al. "Mastering Atari, Go, chess and shogi by planning with a learned model". In: *Nature* 588.7839 (Dec. 2020), 604–609. ISSN: 1476-4687. DOI: 10.1038/s41586-020-03051-4. URL: http://dx.doi.org/10.1038/s41586-020-03051-4.

[73] John Schulman et al. *Proximal Policy Optimization Algorithms*. 2017. arXiv: 1707.06347 [cs.LG]. URL: https://arxiv.org/abs/1707.06347.

[74] Wenling Shang et al. *Understanding and Improving Convolutional Neural Networks via Concatenated Rectified Linear Units*. 2016. arXiv: 1603.05201 [cs.LG]. URL: https://arxiv.org/abs/1603.05201.

[75] David Silver et al. "Mastering chess and shogi by self-play with a general reinforcement learning algorithm". In: *arXiv preprint arXiv:1712.01815* (2017).

[76] Richard S. Sutton, A. Rupam Mahmood, and Martha White. *An Emphatic Approach to the Problem of Off-policy Temporal-Difference Learning*. 2015. arXiv: 1503.04269 [cs.LG]. URL: https://arxiv.org/abs/1503.04269.

[77] Richard S. Sutton et al. "Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction". In: *The 10th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*. AAMAS '11. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems, 2011, 761–768. ISBN: 0982657161.

[78] Edan Toledo. *Stoix: Distributed Single-Agent Reinforcement Learning End-to-End in JAX*. Version v0.0.1. Apr. 2024. DOI: 10.5281/zenodo.10916258. URL: https://github.com/EdanToledo/Stoix.

[79] Vivek Veeriah et al. "Discovery of options via meta-learned subgoals". In: *Advances in Neural Information Processing Systems* 34 (2021), pp. 29861–29873.

[80] Vivek Veeriah et al. "Discovery of Useful Questions as Auxiliary Tasks". In: *Advances in Neural Information Processing Systems*. Ed. by H. Wallach et al. Vol. 32. Curran Associates, Inc., 2019. URL: https://proceedings.neurips.cc/paper_files/paper/2019/file/10ff0b5e85e5b85cc3095d431d8c08b4-Paper.pdf.

[81] Claas Voelcker et al. "When does Self-Prediction help? Understanding Auxiliary Tasks in Reinforcement Learning". In: *arXiv preprint arXiv:2406.17718* (2024).

[82] Christopher JCH Watkins and Peter Dayan. "Q-learning". In: *Machine learning* 8 (1992), pp. 279–292.

[83] Zhongwen Xu, Hado P van Hasselt, and David Silver. "Meta-gradient reinforcement learning". In: *Advances in neural information processing systems* 31 (2018).

[84] Ziping Xu, Kelly W. Zhang, and Susan A. Murphy. *The Fallacy of Minimizing Local Regret in the Sequential Task Setting*. 2024. arXiv: 2403.10946 [stat.ML]. URL: https://arxiv.org/abs/2403.10946.

[85] Omry Yadan. *Hydra - A framework for elegantly configuring complex applications*. Github. 2019. URL: https://github.com/facebookresearch/hydra.

[86] Tom Zahavy et al. *A Self-Tuning Actor-Critic Algorithm*. 2021. arXiv: 2002.12928 [stat.ML]. URL: https://arxiv.org/abs/2002.12928.

[87] Zeyu Zheng, Junhyuk Oh, and Satinder Singh. "On Learning Intrinsic Rewards for Policy Gradient Methods". In: *Advances in Neural Information Processing Systems*. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018. URL: https://proceedings.neurips.cc/paper_files/paper/2018/file/51de85ddd068f0bc787691d356176df9-Paper.pdf.