Efficient Learning and Adaptation in Non-Stationary Reinforcement Learning

By

Naicheng He

Thesis

Submitted in partial fulfillment of the requirements for Honors Sc.B. in the Department of Computer Science at Brown University

PROVIDENCE, RHODE ISLAND

April 2026

This dissertation by Naicheng He is accepted in its present form by the Department of Computer Science as satisfying the dissertation requirement for the degree of Honors Bachelor of Science.

Date _____ _____

Amy Greenwald, Advisor

Recommended to the Undergraduate Council

Date _____ _____

Amy Greenwald, Reader

Date _____ _____

George Konidaris, Reader

Approved by the Undergraduate Council

Date _____ _____

(DEAN'S NAME), (DEAN'S POSITION)

Acknowledgments

First I would like to thank my advisors and mentors for their guidance, patience, and encouragement. Their advice has shaped not only the projects in this thesis, but also the way I think about research. They shaped my view on how to ask meaningful questions, how to be precise about problems, and how to keep improving an idea through repeated discussion and revision.

I am also grateful to my collaborators and fellow students. Many of the ideas in this thesis grew out of conversations with them, and I have learned a great deal from their feedback, criticism, and willingness to think through difficult problems with me. These interactions are not only inspiring but also very enjoyable.

I would also like to thank my friends for their support throughout this process. They made the difficult parts of undergraduate life easier and brought happiness into my life. These friendships mean a great deal to me.

Finally, I want to thank my family for their constant love and support. Their belief in me has given me the confidence to pursue the work I love, and I am deeply grateful for everything they have done for me.

Contents

Introduction

Developing autonomous agents that can make intelligent decisions in complex, ever-shifting environments is a central goal of artificial intelligence. Reinforcement learning has been a central framework for the stationary decision-making problem, where an agent interacts with an environment and learns a policy that maximizes cumulative reward. While this framework has led to strong results in stationary settings, real-world environments are rarely fixed. This thesis aims to discuss building blocks that ultimately lead to continual decision making in non-stationary environments.

The first part of this thesis addresses efficient policy optimization in stationary environments. We develop an on-policy optimization method from a game theoretic perspective, casting actor critic algorithms as bi-level optimizations. This perspective leads to an efficient algorithm for policy optimization and demonstrates strong performance in classic control problems. The second part studies loss of plasticity in deep continual learning. Although neural networks are powerful function approximators, they often lose the ability to adapt after training on long sequences of tasks. We investigate this loss of plasticity and show that it is closely associated with a phenomenon we term *spectral collapse*, in which useful curvature structure vanishes at the start of a new task, making gradient-based learning ineffective. Based on this analysis, we develop a method to preserve plasticity and enable continued learning across tasks. This provides both a mechanistic explanation for continual-learning failure and a practical intervention for mitigating it.

This thesis builds on the work introduced in the initial proposal, but sharpens its focus on

two components of scalable continual reinforcement learning: efficient policy optimization and the preservation of plasticity under continual training. The two works presented in this thesis aim to take a step toward agents that can learn efficiently and adapt continually in changing environments.

CHAPTER 1

Non-Stationary Decision Making

1.1 Reinforcement Learning

A reinforcement learning (RL) agent learns to make decisions by interacting with its environment sequentially. At each time step, the agent observes a state, chooses an action, receives a reward, and transitions to a new state. The goal is to learn a decision rule that maximizes long-term cumulative reward Sutton and Barto (2018). Formally, a discrete-time Markov decision process (MDP) is defined by the tuple

$$\mathcal{M} \doteq \langle \mathcal{S}, \mathcal{A}, P, r, \gamma, \iota \rangle.$$

Where $\mathcal{S}$ denotes a possibly continuous state space and $\mathcal{A}$ denotes a possibly continuous action space. The initial state $\boldsymbol{s}_0$ is drawn from an initial state distribution ι over $\mathcal{S}$. The transition dynamics $P : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ are specified by the conditional probability density$P(\boldsymbol{S}_{t+1} \mid \boldsymbol{s}_t, \boldsymbol{a}_t)$, which describes the probability of transitioning to the next state $\boldsymbol{S}_{t+1}$ after taking action $\boldsymbol{a}_t$ in state $\boldsymbol{s}_t$. The reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ specifies the immediate reward obtained by taking action $\boldsymbol{a}_t$ in state $\boldsymbol{s}_t$.

A rollout trajectory is a sequence $\tau = (\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \ldots)$. The discounted return of a

trajectory is $R_\tau \doteq \sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)$, where $\gamma \in [0, 1]$ is the discount factor. The discount factor controls how much the agent values future rewards relative to immediate rewards.

Given an MDP $\mathcal{M}$, a policy $\boldsymbol{\pi} : \mathcal{S} \to \mathcal{P}(\mathcal{A})$ maps each state to a probability distribution over actions. Together, the initial state distribution ι, the transition dynamics P, and the policy $\boldsymbol{\pi}$ induce a distribution over trajectories. We write this trajectory distribution as $\rho_\iota^{\boldsymbol{\pi}}$. They also induce a discounted occupancy distribution $\nu_\iota^{\boldsymbol{\pi}}$ over states. When the initial state distribution is a Dirac delta at a state $\boldsymbol{s}$, we abuse notation and write $\rho_{\boldsymbol{s}}^{\boldsymbol{\pi}}$ and $\nu_{\boldsymbol{s}}^{\boldsymbol{\pi}}$.

The standard objective in reinforcement learning is to find a policy that maximizes expected return:

$$\boldsymbol{\pi}^* \in \arg\max_{\boldsymbol{\pi}} J(\boldsymbol{\pi}), \qquad J(\boldsymbol{\pi}) \doteq \mathop{\mathbb{E}}_{\tau \sim \rho_\iota^{\boldsymbol{\pi}}} [R_\tau] .$$

In practice, the policy is often represented by a neural network with parameters $\boldsymbol{\theta} \in \mathbb{R}^P$, and we write the policy as $\boldsymbol{\pi}_{\boldsymbol{\theta}}$. Learning then becomes an optimization problem over the parameter vector $\boldsymbol{\theta}$. Different RL algorithms instantiate this optimization problem in different ways. Policy-gradient methods directly optimize a parameterized policy; value-based methods learn value functions that estimate long-term return; and actor-critic methods combine both perspectives by learning a policy together with a value estimator. We defer a more detailed discussion of actor-critic algorithms to Chapter 2.

1.2 From Stationary to Non-Stationary Decision Making

The classic MDP formulation assumes that the environment is stationary where the state space, action space, transition dynamics, reward function, and initial state distribution remain fixed throughout training and evaluation. Under this assumption, the objective does not change(i.e. remain stationary) and the policy only needs to be optimized once.

Many realistic decision-making problems violate this assumption. Non-stationarity may come in many different ways. An autonomous agent may be deployed in environments whose dynamics change over time, may face a sequence of related but distinct tasks, or may need to

adapt to new reward functions, state distributions, or interaction patterns. In these settings, the agent cannot simply learn one policy for one fixed MDP. Instead, it must keep adapting as the learning problem changes.

This thesis is concerned with this broader setting of non-stationary decision making. The central difficulty of this task is that learning different tasks with a same neural network can interfere with the agent's ability to perform well on both learned and unseen tasks. Thus, beyond optimizing performance on a single MDP, we care about whether the agent remains adaptable over a long sequence of learning problems.

1.3 Continual Learning

The non-stationary setting naturally leads to the problem of continual learning. In continual learning, an agent or model is not trained on a single fixed task. Instead, it encounters a sequence of tasks over time and must continue updating its parameters as new learning problems arrive. Unlike standard supervised learning or reinforcement learning, where the training objective is usually fixed, continual learning studies what happens when the objective itself changes across training.

At a high level, continual learning asks whether a model can reuse knowledge from previous tasks while still remaining capable of adapting to future tasks. This creates a fundamental tension. On one hand, reusing parameters across tasks can improve efficiency because the model does not need to relearn useful representations from scratch. On the other hand, training on one task may interfere with performance on previous tasks or may damage the model's ability to learn future tasks.

The Chapter 3 of this thesis is mainly concerned with the second issue: whether a neural network agent remains trainable after learning previous tasks. However, the broader continual learning problem is usually framed around two related phenomena. The first is *catastrophic forgetting*, where learning a new task causes the model to lose performance on earlier tasks.

The second is *loss of plasticity*, where learning earlier tasks makes the model less able to adapt to new tasks. Thus, one way of framing continual learning is as the study of how to preserve both memory of the past and plasticity for the future.

1.4 A Mathematical Formulation of Continual Learning

We define the continual learning problem over a sequence of T tasks and a learner represented by a neural network with parameters $\boldsymbol{\theta} \in \mathbb{R}^P$. Each task $\tau \in \{0, \dots, T-1\}$ comes equipped with an evaluation metric $f_\tau : \mathbb{R}^P \to \mathbb{R}$, such as classification accuracy in supervised learning or expected return in reinforcement learning. If each task were solved independently, the goal for task τ would be to find parameters

$$\boldsymbol{\theta}_\tau^* \in \arg\max_{\boldsymbol{\theta}_\tau} f_\tau(\boldsymbol{\theta}_\tau).$$

Equivalently, the ideal continual learning objective is to find a sequence of task parameters $\{\boldsymbol{\theta}_\tau^*\}_{\tau=0}^{T-1}$ such that $\boldsymbol{\theta}_\tau^*$ maximizes the evaluation metric f_τ for every task τ.

In practice, the evaluation metric may not be directly optimized. For example, accuracy is non-differentiable, and expected return in reinforcement learning is usually optimized through a surrogate objective. We therefore assume that each task τ has a possibly regularized surrogate loss function

$$\mathcal{L}_\tau : \mathbb{R}^P \to \mathbb{R},$$

which is intended to improve the corresponding evaluation metric f_τ, while being more suitable for optimization. Furthermore, we constrain a continual learner with a finite learning budget. We assume that the learner has K update steps per task. Let $\boldsymbol{\theta}_\tau^{(k)}$ denote the parameters after k updates on task τ. Unlike independent task learning, the initialization for task τ depends on the result of previous learning. A common protocol is to initialize each task from the final parameters of the previous task, denoted by $\boldsymbol{\theta}_\tau^{(0)} \doteq \boldsymbol{\theta}_{\tau-1}^{(K)}$. The learning algorithm produces $\boldsymbol{\theta}_\tau^{(K)}$ by optimizing the surrogate objective $\mathcal{L}_\tau$ starting from $\boldsymbol{\theta}_\tau^{(0)}$.

More generally, we can view a continual learning algorithm as a mapping

$$\mathcal{A} : \mathbb{R}^P \times \mathcal{L}_\tau \to \mathbb{R}^P,$$

which takes the current parameters and the task objective as input, and outputs the parameters obtained after training on the current task. which gives $\boldsymbol{\theta}_\tau^{(K)} = \mathcal{A}(\boldsymbol{\theta}_\tau^{(0)}, \mathcal{L}_\tau)$. We develop this formulation to emphasize that continual learning is not a simple combination of independent optimization problems. The solution found for one task changes the starting point, representation, and optimization geometry for later tasks.

1.5 Continual Reinforcement Learning

Continual reinforcement learning is the special case of continual learning in which each task is a reinforcement learning problem. Formally, we define a continual reinforcement learning problem as a sequence of MDPs

$$\mathcal{M}_0, \mathcal{M}_1, \ldots, \mathcal{M}_{T-1},$$

where each task $\mathcal{M}_\tau$ is given by $\mathcal{M}_\tau \doteq \langle \mathcal{S}, \mathcal{A}, P, r, \gamma, \iota \rangle_\tau$. Across tasks, any component of the MDP may change, i.e. the initial state distribution, the transition dynamics, the reward function, or the state and action spaces themselves. In this thesis, we are primarily interested in the setting where the agent uses a shared neural network parameterization across tasks. That is, the agent carries forward its parameters from one task to the next, rather than being reinitialized from scratch.

Let $\boldsymbol{\pi}_{\boldsymbol{\theta}}$ denote a policy parameterized by $\boldsymbol{\theta}$. For task τ, a natural evaluation metric is the expected return of this policy:

$$f_\tau(\boldsymbol{\theta}) \doteq J_\tau(\boldsymbol{\pi}_{\boldsymbol{\theta}}) = \mathop{\mathbb{E}}_{\tau \sim \rho_{\iota_\tau}^{\boldsymbol{\pi}_{\boldsymbol{\theta}}}} \left[\sum_{t=0}^{\infty} \gamma_\tau^t r_\tau(\boldsymbol{s}_t, \boldsymbol{a}_t) \right].$$

The corresponding surrogate objective $\mathcal{L}_\tau$ may be a policy-gradient objective, a value-function loss, an entropy-regularized objective, or an actor-critic objective. The precise form of this objective depends on the RL algorithm used.

This setup captures the basic difficulty of continual reinforcement learning. The agent must not only learn a good policy for the current MDP, but must do so in a way that preserves its ability to learn future MDPs. Since reinforcement learning already involves noisy gradients, distribution shift induced by the policy, and long-horizon credit assignment, these continual learning difficulties can be especially severe in RL. A method that is effective for optimizing a single task may still be insufficient if it leaves the agent less trainable after repeated updates.

1.6 Optimization, Adaptation, and Plasticity

A central theme of this thesis is that continual reinforcement learning depends on two different aspects of this continual learning objective. The first is single-task optimization, where given a fixed task, the agent needs to efficiently improve its policy in the fixed MDP. The second is continued adaptability, where the same agent needs to still learn new tasks after learning previous ones.

The first question motivates Chapter 2, where we study improved optimization methods for actor-critic methods in reinforcement learning. The performance of actor-critic methods depends heavily on the stability and efficiency of the underlying optimization procedure. Improving this single-task learning process is an important step toward agents that can adapt effectively.

The second question motivates Chapter 3 on loss of plasticity in continual reinforcement learning. Even when an agent performs well on early tasks, it may gradually lose the ability to learn new tasks. In neural networks, plasticity refers to the ability of the model to change its behavior in response to new data or new objectives. In continual RL, plasticity is essential because the agent must repeatedly adapt to new decision-making problems.

Definition 1.6.1 (Successful Training)**.** In a continual learning problem, given a finite update budget K and an error tolerance υ, we say that task τ is successfully trained if

$$\ell_\tau(\boldsymbol{\theta}_\tau^{(K)}) \leq \upsilon.$$

Equivalently, we write

$$\mathbb{I}\left(\ell_\tau(\boldsymbol{\theta}_\tau^{(K)}) \leq \upsilon\right)$$

as the indicator of successful training on task τ.

Definition 1.6.2 (Plasticity)**.** A neural network agent is *plastic* over a sequence of T tasks if it successfully trains on every task in the sequence:

$$\ell_\tau(\boldsymbol{\theta}_\tau^{(K)}) \leq \upsilon, \qquad \forall \tau \in \{0, \ldots, T-1\}.$$

With these definitions in hand, we will proceed to first investigate learning and optimization in single-MDP tasks.

CHAPTER 2

Bilevel Policy Optimization with Nyström Hypergradients

2.1 Introduction

In Chapter 1, we framed continual reinforcement learning as a problem of learning and adaptation across a sequence of decision-making tasks. Before studying how learning dynamics vary across tasks, we first study a more basic question, namely how an agent can efficiently optimize its policy within a single MDP. Single-task optimization is an important building block of continual reinforcement learning. An agent that cannot reliably improve its policy in one environment is unlikely to adapt well across many environments.

In this chapter, we focus on actor-critic methods, a classic reinforcement learning architecture in which a policy, called the actor, is trained together with a value function, called the critic. As introduced in Chapter 1, actor-critic methods combine policy optimization with value estimation. Many state-of-the-art reinforcement learning algorithms are built on actor-critic-like structures, such as Trust Region Policy Optimization Schulman et al. (2015), Proximal Policy Optimization Schulman et al. (2017), Deep Deterministic Policy Gradient Lillicrap et al. (2019), and Soft Actor-Critic Haarnoja et al. (2018). Their practical success makes them a

natural starting point for studying efficient learning in reinforcement learning.

The actor and critic, however, are not independent optimization problems. The actor uses the critic's value estimates to update the policy, while the critic's learning objective depends on the state distribution induced by the actor's policy. Standard actor-critic algorithms typically update both networks simultaneously: the actor updates using the current critic, and the critic updates using data generated by the current actor. This simultaneous update rule is simple and effective, but it does not explicitly account for the way that changing the actor changes the critic's future response.

This dependency suggests that actor-critic learning has an inherently hierarchical structure. The actor's objective depends on the critic's response, while the critic exists primarily to aid the actor's policy search. We make this structure explicit by characterizing actor-critic reinforcement learning as a bilevel optimization problem. In this formulation, the actor is the outer player, or Stackelberg leader, and the critic is the inner player, or Stackelberg follower. The critic learns a best-response value function for the actor's current policy, and the actor updates its policy while accounting for how the critic would respond.

Bilevel optimization (BLO) is a class of hierarchical optimization problems with two objectives, an *outer* one and an *inner* one. Crucially, the objective and the variables of the outer problem depend on the solution to the inner problem. Given functions $f : \mathbb{R}^n \times \mathbb{R}^m \to \mathbb{R}$ and $\mathcal{L} : \mathbb{R}^m \to \mathbb{R}$, an unconstrained bilevel optimization problem can be formulated as follows:

$$\min_{\boldsymbol{\phi} \in \mathbb{R}^n} \Phi(\boldsymbol{\phi}) \doteq f(\boldsymbol{\phi}, \boldsymbol{\theta}^*(\boldsymbol{\phi})) \quad \text{s.t.} \quad \boldsymbol{\theta}^*(\boldsymbol{\phi}) \in \mathcal{Y}^*_{\boldsymbol{\phi}} \doteq \arg\min_{\boldsymbol{\theta} \in \mathbb{R}^m} \mathcal{L}_{\boldsymbol{\phi}}(\boldsymbol{\theta}). \tag{2.1}$$

A solution to a BLO comprises a pair $(\boldsymbol{\phi}^*, \boldsymbol{\theta}^*) \in (\mathbb{R}^n, \mathbb{R}^m)$ such that $\boldsymbol{\phi}$ optimizes $\Phi(\boldsymbol{\phi})$ subject to the constraint that $\boldsymbol{\theta}^*$ optimizes $\mathcal{L}_{\boldsymbol{\phi}}(\boldsymbol{\theta})$. The notation $\mathcal{L}_{\boldsymbol{\phi}}$ allows the optimal value of $\mathcal{L}$ and the corresponding solution set $\mathcal{Y}^*_{\boldsymbol{\phi}}$ to vary with $\boldsymbol{\phi}$.

Bilevel optimization problems are sometimes referred to as two-player general-sum Stackelberg games Dempe (2018), with the outer player as the Stackelberg leader and the inner

player as the Stackelberg follower. This language is useful for actor-critic learning because it makes the asymmetry between actor and critic explicit: the actor's policy search is the primary objective, while the critic's value function representation is useful insofar as it supports better actor updates.

Solving a BLO naturally leads to nested gradient descent. For each outer value ϕ, the inner optimization is solved, for example by gradient descent, to find $\boldsymbol{\theta}^*(\phi)$. The outer variable is then updated by following the *hypergradient* of f at $\boldsymbol{\theta}^*(\phi)$. The hypergradient contains a direct gradient term, which captures the direct dependence of f on ϕ, and an implicit gradient term, which accounts for how changes in ϕ affect the inner solution $\boldsymbol{\theta}^*$.

The implicit gradient is the main computational challenge. A common approach is to use the implicit function theorem, which expresses the implicit gradient through an inverse Hessian vector product (IHVP) Lorraine et al. (2020). Directly computing this quantity is infeasible for large neural networks, and iterative approximations such as conjugate gradient can be unstable when Hessians are ill-conditioned. This motivates the central algorithmic choice in this chapter: using the Nyström method Drineas et al. (2005); Hataya and Yamada (2023) to compute a low-rank approximation of the IHVP.

We call the resulting algorithm *Bilevel Policy Optimization* (BLPO). BLPO modifies actor-critic learning in two ways: it nests the critic update so that the critic better approximates a response to the actor's policy, and it updates the actor using a Nyström hypergradient that accounts for this critic response. The rest of this chapter explains the algorithm, describes the Nyström hypergradient computation, and evaluates BLPO on discrete and continuous control tasks.

2.2 Contributions

We propose a new policy-gradient based RL algorithm, Bilevel Policy Optimization with Nyström Hypergradient (BLPO), inspired by our AC-BLO formulation. In this formulation, the

actor plays the role of the Stackelberg leader, while the critic plays the role of the follower. This motivates a nested algorithm that alternates between taking one step along the hypergradient of the actor's objective and taking multiple steps along the gradient of the critic's objective.

In practice, the main obstacle is computing the actor's hypergradient, which requires an IHVP. Previous bilevel actor-critic methods approximate this quantity using conjugate gradient, but CG can be unstable when the Hessian is ill-conditioned. We instead compute a low-rank approximation of the IHVP via the Nyström method. This modification characterizes BLPO and makes the hypergradient computation more stable in deep RL experiments.

Empirically, we show that BLPO outperforms PPO in a variety of discrete and continuous control tasks. We also present ablations showing that both nesting and the Nyström hypergradient are essential for these performance gains. Related, we show that using CG to approximate the IHVP can lead to severe performance degradation.

The original paper also provides a convergence analysis: under a linear parameterization of the critic's value function, BLPO converges in polynomial time to a point satisfying the necessary conditions of a local strong Stackelberg equilibrium, with high probability. Since this chapter focuses on the algorithm and its empirical behavior, we refer to the original paper for the full proof Prakash et al. (2025).

2.3 Related Work

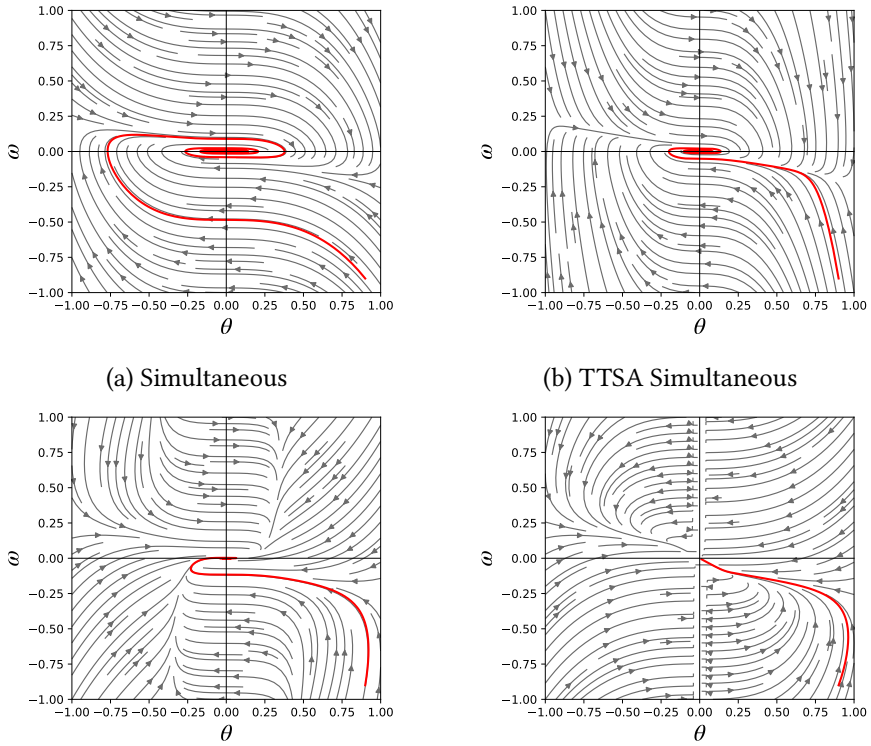
Bilevel optimization has applications in many areas of machine learning and beyond, including reinforcement learning Chakraborty et al. (2024); Zheng et al. (2022), hyperparameter optimization Lorraine et al. (2020), meta-learning Rajeswaran et al. (2019), energy markets Afcsar et al. (2016), agriculture Bard et al. (1998), and security Sinha et al. (2018). In machine learning, bilevel methods are especially useful when one optimization problem depends on the solution of another. This is exactly the structure that appears in actor-critic learning: the actor's update depends on the critic's value estimate, while the critic's training objective

depends on the actor's policy-induced data distribution.

Actor-critic as a BLO, or equivalently as a Stackelberg game, has been considered previously by the reinforcement learning community. Zheng et al. (2022); Wen et al. (2021) attempted to solve various MDPs using actor-critic variants that incorporate hypergradients. These methods are closely related to ours in motivation, but in practice they are largely unstable because they compute the hypergradient using conjugate gradient. When the Hessian is ill-conditioned, conjugate gradient can produce poor approximations of the IHVP, leading to unreliable actor updates. Chakraborty et al. (2024) developed PARL, a hypergradient-based method for RLHF, which also uses conjugate gradient and can suffer from inaccurate hypergradient estimates Ding et al. (2024).

A related line of work studies actor-critic as a Nash, or simultaneous-move, game. Castro and Meir (2010) analyze two-timescale stochastic approximation actor-critic algorithms with linear function approximation, and show convergence to a neighborhood around a local Nash equilibrium. Heusel et al. (2017) sharpen this type of analysis to show that two-timescale updates with a faster critic can converge to a local Nash equilibrium almost surely. These results justify the common practice of training the critic faster than the actor, but they do not yield the same Stackelberg update as BLPO, because they rely on gradients rather than hypergradients.

Our work also builds on the broader literature on non-convex–strongly-convex bilevel optimization. Ghadimi and Wang (2018) analyze iterative methods for this class of BLO problems using hypergradients and the implicit function theorem. Ji et al. (2021) sharpen this analysis using warm starts in the inner optimization. Hataya and Yamada (2023) introduce the Nyström method to estimate hypergradients using low-rank approximations, addressing some of the computational difficulties associated with iterative methods Lorraine et al. (2020). BLPO combines these ideas with actor-critic reinforcement learning: it uses the Stackelberg formulation of actor-critic, computes the actor update through a hypergradient, and uses the Nyström method to stabilize the IHVP approximation.



(a) Simultaneous

(b) TTSA Simultaneous

(c) Regularized Stackelberg

(d) Stackelberg

Figure 2.1: Simultaneous training dynamics cycle in the one-step MDP from Zheng et al. (2022), even with TTSA. Stackelberg dynamics converge to the equilibrium, $(0, 0)$, even when regularized.

2.4 Using a Nyström Hypergradient

The main computational challenge in BLPO is the IHVP in **??**. Given the computational intractability of directly computing the IHVP, many approaches have been proposed to instead approximate it. One of the most popular is the conjugate gradient (CG) method Hestenes et al. (1952), which iteratively refines the solution to the linear system $\boldsymbol{A}\boldsymbol{x} = \boldsymbol{b}$ by updating in conjugate directions until convergence to $\boldsymbol{x} = \boldsymbol{A}^{-1}\boldsymbol{b}$. In our application, $\boldsymbol{A} = \nabla^2_{\boldsymbol{\omega}} L_{\boldsymbol{\theta}}$ and $\boldsymbol{b} = \nabla_{\boldsymbol{\omega}} J$.

While CG overcomes the memory issue of storing the entire Hessian, it can take an arbitrary number of iterations to converge Frangella et al. (2023). Furthermore, CG requires a well-conditioned matrix to converge. If the matrix is not well-conditioned, numerical errors can accumulate, and the approximation can become arbitrarily bad. As shown in Figure 2.2, ill conditioning can arise from the inherent non-convexity of deep neural networks and from

overparameterization Saarinen et al. (1993); Paige and Saunders (1975), which can cause the IHVP error to spike when using CG.

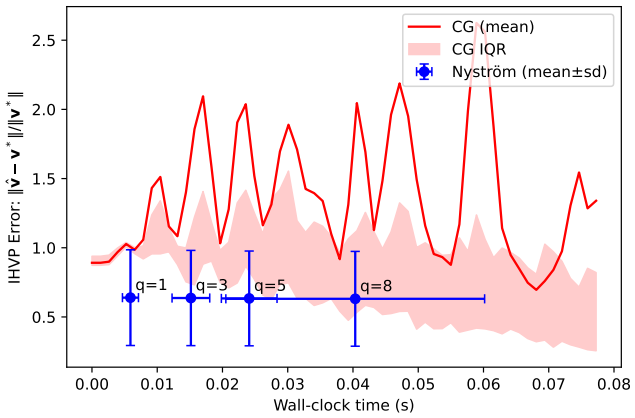


Figure 2.2: We calculate the IHVP error averaged over 100 small neural networks using both the Nyström method and CG. The CG mean is consistently outside the interquartile range (IQR), which suggests at least occasional occurrence of massive errors.

Proposed by Hataya and Yamada (2023), the Nyström method produces a low-rank approximation of the IHVP. Assume a p-dimensional positive semidefinite Hessian denoted by $\mathbf{H}$ and a low-rank approximation of $\mathbf{H}$ denoted by $\mathbf{H}_q$, for $q \ll p$. By selecting a set of Q indices of size q at random, we obtain the Nyström approximation $\mathbf{H}_q = \mathbf{H}_{[:,Q]}\mathbf{H}^{\dagger}_{[Q,Q]}\mathbf{H}^{\top}_{[:,Q]}$, where $\mathbf{H}_{[:,Q]} \in \mathbb{R}^{p\times q}$ is a matrix of select columns of $\mathbf{H}$ at indices Q, $\mathbf{H}_{[Q,Q]} \in \mathbb{R}^{q\times q}$ is a matrix of select rows of $\mathbf{H}_{[:,Q]}$ at indices Q, and $\mathbf{H}^{\dagger}_{[Q,Q]}$ is the Moore-Penrose pseudoinverse of $\mathbf{H}_{[Q,Q]}$.

For some small regularization constant $\alpha > 0$, we define the α-regularized IHVP as $(\mathbf{H}_q + \alpha\boldsymbol{I})^{-1}$, where $\boldsymbol{I}$ is the p-dimensional identity matrix. Applying the Woodbury matrix identity gives

$$(\mathbf{H}_q + \alpha\boldsymbol{I})^{-1} = \frac{1}{\alpha}\boldsymbol{I} - \frac{1}{\alpha^2}\mathbf{H}_{[:,Q]}\left(\mathbf{H}_{[Q:Q]} + \frac{1}{\alpha}\mathbf{H}^{\top}_{[:,Q]}\mathbf{H}_{[:,Q]}\right)^{-1}\mathbf{H}^{\top}_{[:,Q]}. \tag{2.2}$$

This final expression allows us to approximate the IHVP $\boldsymbol{v}$ by $\widehat{\boldsymbol{v}} = (\mathbf{H}_q + \alpha\boldsymbol{I})^{-1}\nabla_{\boldsymbol{\theta}} f(\boldsymbol{\phi}, \hat{\boldsymbol{y}})$.

The regularization parameter α plays a crucial role in balancing numerical stability and the fidelity of curvature information. As discussed by Vicol et al. (2022), for an eigenvalue λ of the

Hessian H, the effect of regularization is to modify the inverse eigenvalue as $(\lambda + \alpha)^{-1}$. When $\lambda \gg \alpha$, this term behaves as λ^{-1}, preserving curvature information. Conversely, when $\alpha \gg \lambda$, the approximation becomes insensitive to low-curvature directions, effectively ignoring them. In practice, α is selected empirically.

One practical advantage of the Nyström method is its ability to operate effectively with smaller α. For example, we use $\alpha = 50$, compared to $\alpha = 500$ in Zheng et al. (2022) for continuous control RL, and $\alpha = 10,000$ in Fiez et al. (2020) for a GAN-based bilevel problem, both of which use CG. We find that the Nyström method offers an attractive balance, where regularization remains modest without sacrificing stability.

2.5 Implementation Details

We use as our baseline the PPO algorithm from PureJaxRL Lu et al. (2022), with separate policy and value networks of fully-connected MLPs, each with two hidden layers of 64 units. Our implementation of BLPO—BLPO-Nyström hereafter, specified in **??**—extends this implementation of PPO with a Nyström-based hypergradient computation and nested critic updates. We also test another variant, BLPO-CG, which uses CG to estimate the hypergradient instead of Nyström's method. In all cases, we calculate advantages using the Generalized Advantage Estimator (GAE) Schulman et al. (2018).

The implementation follows the practical structure of PPO. We collect rollouts using the current policy, compute advantages using the current value function, split the rollout batch into mini-batches, and perform several epochs of updates. The difference is that each actor update is preceded by nested critic updates, and the actor is updated using a hypergradient rather than only the direct policy gradient. In continuous control, we use a multivariate Gaussian to represent the policy network.

The eigenspectrum of the Hessian has been found to have low empirical rank, with the bulk of the eigenspectrum clustered around zero with several large outliers Sagun et al. (2016);

Ghorbani et al. (2019). As a result, we clip the IHVP to ensure that the inversion does not lead to an arbitrarily large gradient step, which would destabilize training. We also clip the term involving $\log \boldsymbol{\pi}_{\boldsymbol{\theta}}(\boldsymbol{a} \mid \boldsymbol{s}) A(\boldsymbol{s}, \boldsymbol{a})$ in a way analogous to PPO's clipped actor objective. These implementation details were important for making BLPO-Nyström stable in practice.

We conduct experiments on discrete control tasks from Gymnax Lange (2022) and continuous control tasks from Brax Freeman et al. (2021). All experiments were run using the default PureJaxRL PPO hyperparameters on a single RTX 3090 GPU. Our code is available online at https://github.com/Arnie-He/BLPO.

Rather than tuning separately for every environment, we use a shared set of PPO-style hyperparameters whenever possible. The main algorithm-specific hyperparameters are the actor learning rate, critic learning rate, number of nested critic updates, the IHVP bound, and the Nyström rank. Table 2.1 summarizes the most important hyperparameters used in our experiments.

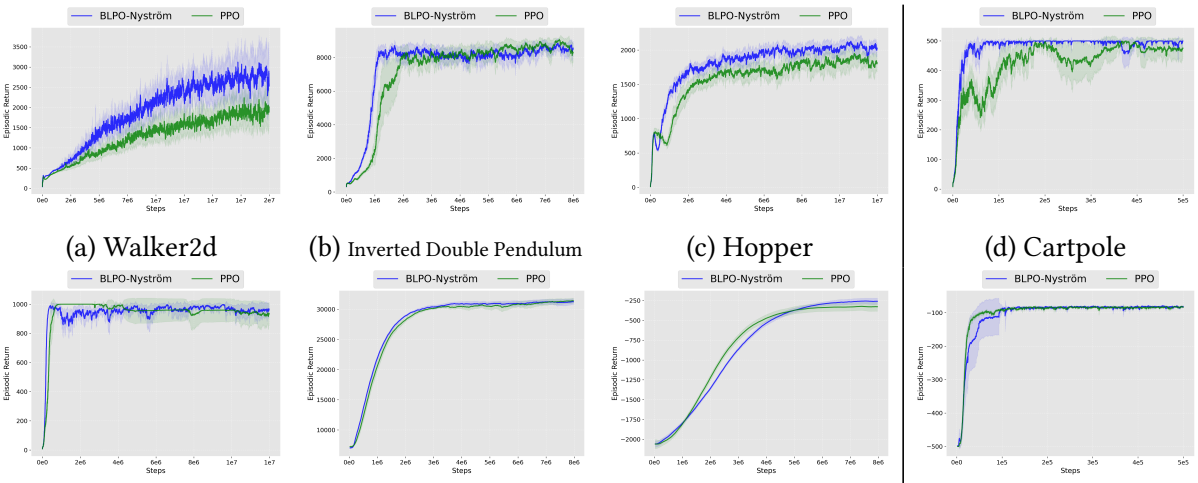
Table 2.1: Key hyperparameters for PPO, BLPO-CG, and BLPO-Nyström. Numbers in parentheses indicate the discrete-control value when it differs from the continuous-control value.

Hyperparameter	PPO	BLPO-CG	BLPO-Nyström
LR / ACTOR_LR	2.5e-4	2.5e-4	2.5e-4
CRITIC_LR	/	1e-3	1e-3
NESTED_UPDATES	/	3 (10)	3 (10)
IHVP_BOUND	/	1.0	1.0
CLIP_F	/	0.5	0.5
MAX_CG_ITER	/	20	/
NYSTROM_RANK	/	/	5
NYSTROM_RHO	/	/	50

2.6 Experiments

2.6.1 Results

In Figure 2.3, we report 95% confidence intervals around the average episodic returns across 15 seeds. We find that BLPO outperforms PPO in most of our experiments and matches PPO



(a) Walker2d (b) Inverted Double Pendulum (c) Hopper (d) Cartpole

(e) Inverted Pendulum (f) Humanoid Standup (g) Pusher (h) Acrobot

Figure 2.3: In control tasks, BLPO either outperforms PPO or performs comparably. The left three columns show continuous control experiments; the right column shows discrete control experiments.

in the rest. In Walker2D and Hopper, BLPO learns a significantly better policy than PPO. Moreover, BLPO converges with fewer samples than PPO in both Pendulum tasks.

Performance parity on simpler tasks, such as Inverted Pendulum and Acrobot, reflects a ceiling effect: PPO already finds near-optimal solutions rapidly, rendering hypergradient information unnecessary. Similarly, while Humanoid Standup involves a high-dimensional state and action space, its optimal configuration is a static hold with relatively non-chaotic dynamics. BLPO's advantage emerges in complex, chaotic environments such as Inverted Double Pendulum, and in open-ended environments without a clear performance ceiling, such as Walker2d and Hopper, where actor-critic coupling is strong.

In these settings, actor updates can drastically shift the value landscape, causing PPO's performance, which mimics simultaneous updates, to degrade. BLPO's formulation, which allows the actor to anticipate the critic's response via the hypergradient, yields better policy updates. In summary, BLPO should be the preferred choice when simulations are expensive, so that the reduced number of environment steps outweighs the per-step gradient overhead.

2.6.2 Ablations

We perform a series of ablations to isolate the contributions of the Nyström hypergradient and the nested critic update. First, we compare BLPO-Nyström against BLPO-CG, which keeps the same bilevel actor-critic structure but replaces the Nyström approximation with conjugate gradient. We find that BLPO-Nyström either outperforms or matches BLPO-CG in terms of episodic returns across all tasks. This result supports the claim that the Nyström method is not merely a computational convenience, but an important part of making hypergradient-based actor-critic learning stable in practice.

Second, we compare against a variant that nests the critic update but does not use the actor hypergradient. This ablation tests whether performance gains come only from training the critic more thoroughly. We find that nesting alone is not enough: the hypergradient is necessary for the actor to account for how the critic changes in response to policy updates.

Third, we compare against a two-timescale variant, where the actor and critic are still updated simultaneously but use separate learning rates. This comparison tests whether the improvement comes only from allowing the critic to learn faster. We find that simply changing the actor and critic learning rates is not sufficient. The Stackelberg structure, with the actor as leader and the critic as follower, is an important part of the algorithm.

Together, these ablations show that both nesting and the Nyström hypergradient are essential. Nesting improves the critic's response to the current policy, while the hypergradient allows the actor to update while anticipating this response. The Nyström approximation then makes this hypergradient stable enough to work in deep RL experiments.

2.6.3 Runtime Analysis

Algorithms that rely on second-order information can run significantly slower than their first-order counterparts. On the simple control tasks, we find that BLPO-Nyström is indeed slower than PPO. In the more complex environments, however, the gap between all algorithms

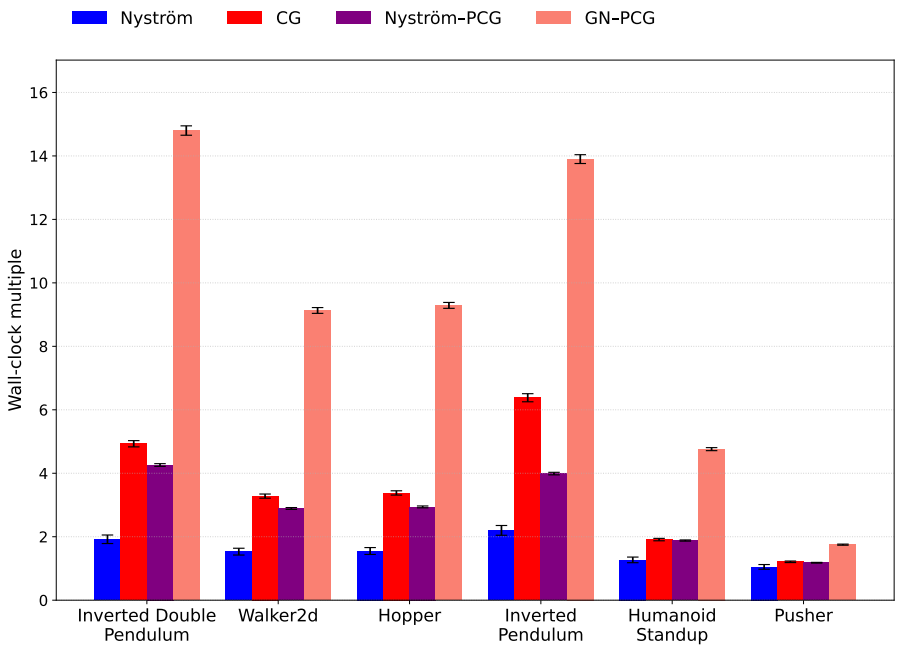


Figure 2.4: Wall-clock runtime of BLPO variants as a multiple of PPO's runtime, where PPO corresponds to $1\times$.

is smaller, as most of the computation time is spent on the simulator.

With regards to BLPO-CG, although the calculation of an IHVP takes approximately 1.5 times a normal gradient step Pearlmutter (1994); Dagréou et al. (2024), BLPO-CG is slower than BLPO-Nyström on all tasks, as poor conditioning necessitates more CG iterations. We also experimented with BLPO variants using preconditioned conjugate gradient.[1] Interestingly, the Nyström method can itself be used as a preconditioner for conjugate gradient Frangella et al. (2023). We find that doing so improves the convergence time over unconditioned CG. We also use the Gauss-Newton matrix, a positive-semidefinite approximation of the Hessian; however, the additional overhead required to compute the Gauss-Newton matrix makes this method impractical. Our results, shown in Figure 2.4, demonstrate that the Nyström method achieves the best balance of speed and performance.

[1]Empirically, we found 50 iterations maximized the performance of CG in the RL environments in our study, and we thus set the maximum number of CG iterations to 50.

2.7 Conclusion

The main idea in this chapter is to view actor-critic learning as a bilevel optimization problem where the actor is the outer player and the critic is the inner player. This perspective leads to BLPO, which nests critic updates and updates the actor using a Nyström hypergradient. Empirically, BLPO matches or outperforms PPO across several discrete and continuous control tasks, and the ablations show that both the nested critic update and the Nyström approximation are important.

Overall, this chapter contributes to the stationary decision-making part of the thesis by contributing to policy optimization within a fixed environment.

CHAPTER 3

Spectral Collapse Drives Loss of Plasticity in Deep Continual Learning

3.1 Introduction

Human intelligence is marked not by mastering a single task, but by the ability to accumulate knowledge, refine it, and adapt to new challenges. Likewise, a central aspiration of artificial intelligence is to create systems that can learn and adapt throughout their lifetimes. Achieving artificial systems endowed with this ability would ensure that these agents remain useful, relevant, and robust in an open world (Javed and Sutton, 2024).

Deep learning and deep reinforcement learning have demonstrated remarkable progress, reaching human and even superhuman performance in image recognition (Siméoni et al., 2025), game playing (Silver et al., 2017), and reasoning (Bakhtin et al., 2022). But these advances have largely been realized under stationary data distributions. **Continual learning** concerns a system's ability to perform well through a sequence of evolving tasks. Within this framework, two complementary objectives are often distinguished: (i) *maintaining plasticity*, meaning the capacity to acquire new knowledge (Dohare et al., 2024), and (ii) *preventing catastrophic forgetting*, where the learner forgets how to solve tasks it previously mastered (Kirkpatrick

et al., 2017; Parisi et al., 2018; Baker et al., 2023).

This chapter focuses on the first objective: maintaining plasticity. Despite significant empirical progress in the area (Lyle et al., 2023, 2024; Abbas et al., 2023), there remains no unifying consensus on what drives loss of plasticity (Klein et al., 2024). Several explanations have been investigated, including inactive neurons which output constants regardless of the inputs (Bjorck et al., 2021; Sokar et al., 2023), the reduction in feature expressiveness associated with large parameter norms (Lyle et al., 2024), and overfitting (Igl et al., 2020). Recent work has begun to connect plasticity loss to second-order properties, such as the stable rank of the Hessian or layer-wise spectral conditioning (Lewandowski et al., 2024b,a).

Building on these insights, we argue that these seemingly disparate observations are different facets of the same phenomenon, which we call **spectral collapse**: a degeneration of the Hessian eigenspectrum, indicating loss of curvature in most directions. The central claim of this chapter is that loss of plasticity occurs when a network loses the curvature directions required for efficient finite-budget learning on new tasks. Inactive neurons, low feature rank, and parameter norm growth can all contribute to poor optimization geometry, but the shared bottleneck is the same: the network becomes poorly conditioned for further learning.

Figure 3.1 illustrates this phenomenon. Under standard backpropagation, the Hessian spectrum collapses as training proceeds across tasks. In contrast, the spectrum remains broader under $L2$-ER, corresponding to maintained plasticity. This motivates the two main questions of the chapter: why does spectral collapse prevent learning, and how can we prevent it?

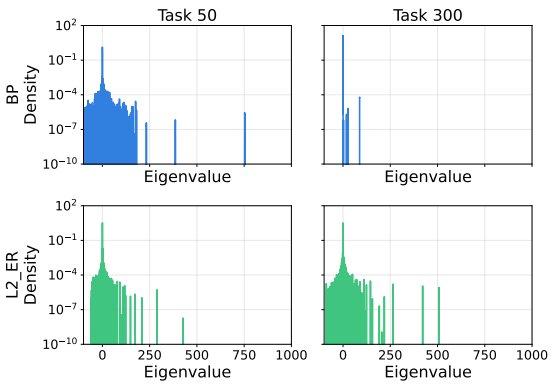


Figure 3.1: Hessian eigenspectrum on Continual ImageNet. BP undergoes spectral collapse by task 300, while $L2$-ER preserves a broader spectrum and maintains plasticity.

3.2 Contributions

This chapter studies loss of plasticity in deep continual learning from the perspective of Hessian curvature. Our main contributions can be summarized as follows:

1. We identify Hessian spectral collapse as a central mechanism underlying plasticity loss. We provide extensive empirical evidence across continual learning benchmarks, demonstrating that networks which lose plasticity exhibit collapsed Hessian spectra, while plasticity-preserving interventions maintain broader spectra.

2. We summarize the core NTK-based theory explaining why spectral collapse prevents finite-budget learning. In a linearized network, residual components contract quickly only along eigendirections with sufficiently large eigenvalues. This yields an explicit $\epsilon_{K,\gamma}$-rank condition for successful training: the initial residual must lie mostly in the fast eigenspace. For cross-entropy loss, the relevant loss-weighted Gram matrix shares its nonzero eigenvalues with the Generalized Gauss–Newton matrix, connecting function-space learning dynamics to parameter-space curvature.

3. Leveraging this perspective, we propose $L2$-Effective Feature Rank ($L2$-ER) regularization, which targets Hessian eigenspectrum collapse directly. Across continual supervised learning and reinforcement learning benchmarks, $L2$-ER matches or exceeds existing methods and preserves plasticity by maintaining usable curvature directions.[1]

3.3 Related Work

Causes of Plasticity Loss. Several pathologies have been proposed to explain loss of plasticity in continual supervised and reinforcement learning (Lyle et al., 2023; Klein et al., 2024). Dormant neurons, which output a positive constant on all inputs, reduce a network's expressive power, while dead neurons, which output zero on all inputs, prevent gradient flow (Montufar et al., 2014; Sokar et al., 2023). Reductions in effective feature rank have also been

[1] Code is available at `https://anonymous.4open.science/r/lop-jax-E17B/README.md`.

linked to plasticity loss, as they suggest that feature representations are collapsing to a lower-dimensional subspace (Roy and Vetterli, 2007; Dohare et al., 2024). Parameter norm growth is another common explanation, since large parameter norms can lead to ill-conditioned Hessians and unstable optimization (Obando-Ceron et al., 2024; Lyle et al., 2024). Rather than treating these explanations as unrelated, this chapter studies them as different ways a network may lose usable curvature directions.

Mitigating Plasticity Loss. Existing methods for preserving plasticity typically target one or more of these pathologies. Continuous interventions apply throughout optimization, while intermittent interventions periodically reset or perturb parameters (Juliani and Ash, 2024). For example, $L2$ regularization controls parameter norm growth, layer normalization stabilizes pre-activation distributions, and spectral regularization controls layer-wise conditioning (Lyle et al., 2024; Lewandowski et al., 2024a). Architectural changes such as PELU, CRELU, and adaptive rational activations attempt to prevent saturation (Godfrey, 2019; Shang et al., 2016; Abbas et al., 2023; Delfosse et al., 2021). Reset-based methods, including Shrink & Perturb, continual backpropagation, and ReDo, revive dormant or dead neurons by periodically replacing parts of the network (Ash and Adams, 2020; Dohare et al., 2024; Sokar et al., 2023). Our view explains why these methods can help: they either reduce the number of inactive directions or preserve curvature in more parameter directions.

Curvature-Based Explanations. This work is closely related to recent studies connecting plasticity loss to second-order properties of neural networks. Lewandowski et al. (2024b) show that plasticity loss correlates with a decline in the stable rank of a partial Hessian and propose a Wasserstein penalty to maintain characteristics of the initial parameter distribution. Lewandowski et al. (2024a) regularize the spectral properties of layer-wise weight matrices to maintain favorable conditioning and gradient diversity. In contrast, this chapter focuses directly on Hessian spectral collapse: the loss of non-negligible curvature directions in parameter space.

Neural Tangent Kernel in Continual Learning. The neural tangent kernel (NTK) (Jacot et al., 2018) provides a tractable lens on training dynamics by approximating neural network learning as kernel regression around initialization. Prior work has used NTK-style analyses to study transfer and forgetting (Doan et al., 2021), and empirical NTK degeneracy has been used as a diagnostic for plasticity loss (Lyle et al., 2024). The original paper underlying this chapter also contains an NTK-based analysis of linearized ReLU networks. In this thesis, we summarize only the main implication: finite-budget learning requires residuals to align with fast eigendirections, and spectral collapse destroys these directions.

3.4 Spectral Collapse

We use **spectral collapse** to refer to the degeneration of the Hessian eigenspectrum at the initialization of a new task. Let $\mathbf{H}_\tau^{(0)}$ denote the Hessian of the task-τ loss at the parameters inherited from the previous task. If most eigenvalues of $\mathbf{H}_\tau^{(0)}$ are close to zero, then the loss landscape is effectively flat in most parameter directions. In this regime, even though the network may contain many parameters, gradient descent has only a small number of directions along which it can make meaningful progress.

To quantify this phenomenon, we define the ϵ-rank of a Hessian as

$$\epsilon\text{-rank}(\mathbf{H}) \doteq \#\{i : |\lambda_i(\mathbf{H})| > \epsilon\},$$

where $\lambda_i(\mathbf{H})$ is the ith eigenvalue of $\mathbf{H}$. The ϵ-rank measures the number of curvature directions whose magnitudes are above a threshold. A large ϵ-rank indicates that many directions remain useful for optimization, while a small ϵ-rank indicates that the spectrum has collapsed.

The empirical evidence shows that spectral collapse is tightly coupled with plasticity loss. On Continual ImageNet, standard backpropagation initially has a broad Hessian spectrum, but later tasks exhibit a spectrum concentrated near zero. At the same time, task accuracy

drops. Methods that preserve plasticity, such as $L2$-ER, maintain a broader Hessian spectrum throughout the tasks.

Figure 3.2 makes this connection explicit. The normalized ϵ-rank is positively associated with classification accuracy: when the Hessian retains more non-negligible curvature directions, the task is more likely to be successfully trained. Conversely, low ϵ-rank corresponds to spectral collapse and unsuccessful training. Therefore, the central theoretical question of the chapter is to connect the disappearance of curvature directions with a neural network's failure of learning new tasks.

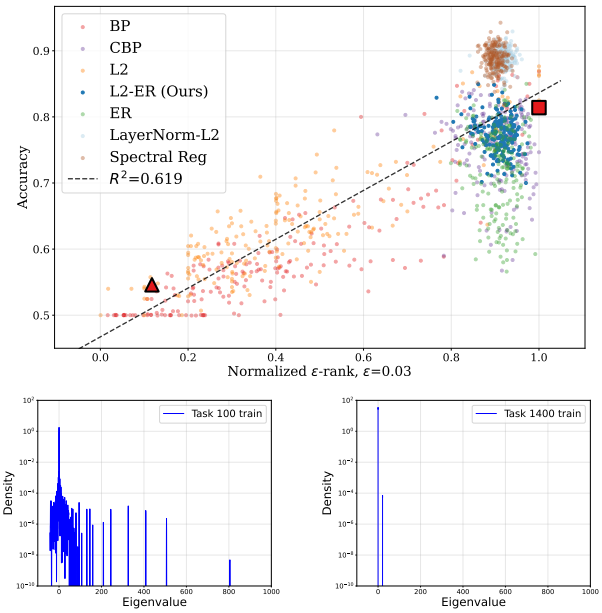


Figure 3.2: Accuracy versus normalized ϵ-rank($\mathbf{H}$) on Continual ImageNet, with BP Hessian spectra before and after spectral collapse. Higher ϵ-rank is associated with successful training.

3.5 Spectral Collapse Drives Loss of Plasticity

The previous section shows that spectral collapse is tightly coupled with loss of plasticity. We now summarize the theoretical reason why this happens. The key idea is finite-budget learning: in continual learning, each task is trained for only a limited number of gradient steps. A network can learn a new task only if the inherited initialization still has directions along which gradient descent can reduce the new task residual quickly.

The NTK perspective makes this statement precise. In the linearized regime, the network

[1]The x-axis is **normalized ϵ-rank(H)**, i.e., $\epsilon\text{-rank}(\mathbf{H})/\#\{\text{eigenvalues of } \mathbf{H}\}$, for $\epsilon = 0.1$.

prediction on task τ can be approximated by

$$\widehat{F}_{\boldsymbol{\theta}}(\boldsymbol{X}_\tau) \approx F_{\boldsymbol{\theta}_\tau^{(0)}}(\boldsymbol{X}_\tau) + \mathbf{J}_\tau(\boldsymbol{\theta} - \boldsymbol{\theta}_\tau^{(0)}),$$

where $\mathbf{J}_\tau$ is the Jacobian at task initialization. For squared loss, define the residual $r_\tau^{(k)}$ after k gradient updates and the empirical Gram matrix $\mathbf{G}_\tau \doteq \mathbf{J}_\tau \mathbf{J}_\tau^\top$. Gradient descent then obeys the closed-form residual dynamics

$$r_\tau^{(k+1)} = (I - \eta \mathbf{G}_\tau) r_\tau^{(k)}, \qquad r_\tau^{(K)} = (I - \eta \mathbf{G}_\tau)^K r_\tau^{(0)}.$$

Let $\{(\lambda_i, v_i)\}_{i=1}^n$ be the eigenpairs of $\mathbf{G}_\tau$. Along eigendirection v_i, the residual contracts by $(1 - \eta\lambda_i)^K$ after K updates. Thus, finite-budget training separates the spectrum into fast and slow directions. For a target contraction factor $\gamma \in (0, 1)$, define

$$\epsilon_{K,\gamma} \doteq \frac{1 - \gamma^{1/K}}{\eta} \approx \frac{-\log \gamma}{\eta K}.$$

Eigenvalues above $\epsilon_{K,\gamma}$ form the fast eigenspace, while eigenvalues below $\epsilon_{K,\gamma}$ form the slow eigenspace:

$$\mathcal{F}_\tau \doteq \{i : \lambda_i(\mathbf{G}_\tau) \geq \epsilon_{K,\gamma}\}, \qquad \mathcal{S}_\tau \doteq \{i : \lambda_i(\mathbf{G}_\tau) < \epsilon_{K,\gamma}\}.$$

Let $P_{\mathcal{F}}$ and $P_{\mathcal{S}}$ denote the corresponding projection matrices.

The core theoretical result is that successful finite-budget training requires the initial residual to lie almost entirely in the fast eigenspace. If task τ is successfully trained within K gradient steps, meaning

$$\frac{1}{2}\|r_\tau^{(K)}\|^2 \leq \upsilon,$$

then the initial residual must satisfy

$$\|P_{\mathcal{S}} r_\tau^{(0)}\| \leq \frac{\sqrt{2\upsilon}}{\gamma}.$$

Equivalently, writing $v_\tau \doteq r_\tau^{(0)}/\|r_\tau^{(0)}\|$, successful training requires

$$\|P_{\mathcal{F}} v_\tau\|^2 \geq 1 - \left(\frac{\sqrt{2\upsilon}}{\gamma \|r_\tau^{(0)}\|}\right)^2.$$

This condition makes the role of spectral collapse explicit. If the fast eigenspace is large, then a new task residual is more likely to have enough projection onto directions that gradient descent can reduce within the task budget. If the spectrum collapses and the fast eigenspace shrinks, then the residual has too much mass in slow directions, and learning fails even if the model is still highly overparameterized.

For cross-entropy loss, the same idea applies to the loss-weighted Gram matrix

$$\mathbf{G}_{\mathrm{LW}} \doteq H_z^{1/2} \mathbf{G} H_z^{1/2},$$

where H_z is the Hessian of the loss with respect to logits. This operator is important because it connects the NTK dynamics to Hessian curvature. Let

$$\mathbf{H}_{\mathrm{GGN}} \doteq \mathbf{J}^\top H_z \mathbf{J}$$

be the Generalized Gauss–Newton matrix. If $A \doteq H_z^{1/2}\mathbf{J}$, then $\mathbf{G}_{\mathrm{LW}} = AA^\top$ and $\mathbf{H}_{\mathrm{GGN}} = A^\top A$. Therefore, $\mathbf{G}_{\mathrm{LW}}$ and $\mathbf{H}_{\mathrm{GGN}}$ share the same nonzero eigenvalues.

This spectral equivalence is the bridge from function space to parameter space. The NTK analysis suggests that learning depends on the fast eigenspace of the loss-weighted Gram matrix while GGN equivalence says that this is the same nonzero spectrum as a parameter-space curvature matrix. Thus, spectral collapse of the Hessian or GGN removes the directions needed for finite-budget learning. We refer the full derivation and linearized-network analysis to the original paper.

3.6 Mitigating Loss of Plasticity with $L2$-ER

The previous section suggests a natural mitigation strategy: preserve enough curvature directions so that future tasks remain trainable. Since computing and regularizing the full Hessian is infeasible for modern neural networks, we instead use structured approximations that reveal which quantities control curvature rank.

The Hessian can be decomposed into a generalized Gauss–Newton term and a residual term:

$$\mathbf{H} = \underbrace{\frac{1}{N}\sum_{i=1}^{N} \mathbf{J}_i^\top \left[\nabla^2_{F^{\boldsymbol{\theta}}} \ell(F^{\boldsymbol{\theta}}(\boldsymbol{x}^i), \boldsymbol{y}^i)\right] \mathbf{J}_i}_{\mathbf{H}_{\mathrm{GGN}}} + \mathbf{R}.$$

Dropping the residual term yields the generalized Gauss–Newton matrix $\mathbf{H}_{\mathrm{GGN}}$, a standard curvature proxy.

To make this approximation scalable, we use a Kronecker-factored approximate curvature perspective. For layer l, the layer-wise GGN block can be approximated by

$$\widehat{\mathbf{H}}^l_{\mathrm{GGN}} \approx \mathop{\mathbb{E}}_n[a^l_n (a^l_n)^\top] \otimes \mathop{\mathbb{E}}_n[g^l_n (g^l_n)^\top],$$

where a^l_n is the input activation to layer l and g^l_n is the corresponding backpropagated gradient. Since $\mathrm{rank}(A \otimes B) = \mathrm{rank}(A)\mathrm{rank}(B)$, the rank of the approximate curvature block is controlled by the ranks of the activation and gradient covariance matrices.

This motivates effective feature rank regularization. The rank of the activation covariance $\mathbb{E}_n[a^l_n (a^l_n)^\top]$ reflects the intrinsic dimensionality of the layer representation. If the representation collapses to a low-dimensional subspace, then the approximate Hessian also loses rank. Encouraging high effective feature rank therefore helps preserve curvature.

Effective rank alone, however, is not sufficient. It acts primarily through the GGN approximation and does not fully control parameter norm growth or the residual component of the Hessian. We therefore combine effective-rank regularization with an $L2$ penalty. The $L2$ term

controls parameter norm growth and helps maintain better-conditioned curvature.

Given task τ with data distribution $\boldsymbol{d}_\tau$ and loss ℓ_τ, standard training minimizes $\mathbb{E}_{\boldsymbol{d}_\tau}[\ell_\tau(F^{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y})]$. We instead use the $L2$-Effective Feature Rank objective:

$$\min_{\boldsymbol{\theta}} \underbrace{\mathbb{E}_{\boldsymbol{d}_\tau}[\ell_\tau(F^{\boldsymbol{\theta}}(\boldsymbol{x}), \boldsymbol{y})]}_{\text{task loss}} - \beta \underbrace{\sum_{l \in \mathcal{L}} \operatorname{erank}\left(\mathbb{E}_n[a_n^l (a_n^l)^\top]\right)}_{\text{effective-rank penalty}} + \underbrace{\lambda \|\boldsymbol{\theta}\|_2^2}_{L2 \text{ regularization}} .$$

Here, β controls the strength of the effective-rank penalty and λ controls the strength of the $L2$ penalty. The effective-rank term encourages representations to use more independent directions, while the $L2$ term prevents uncontrolled parameter growth. Together, they directly target spectral collapse.

3.7 Experiments

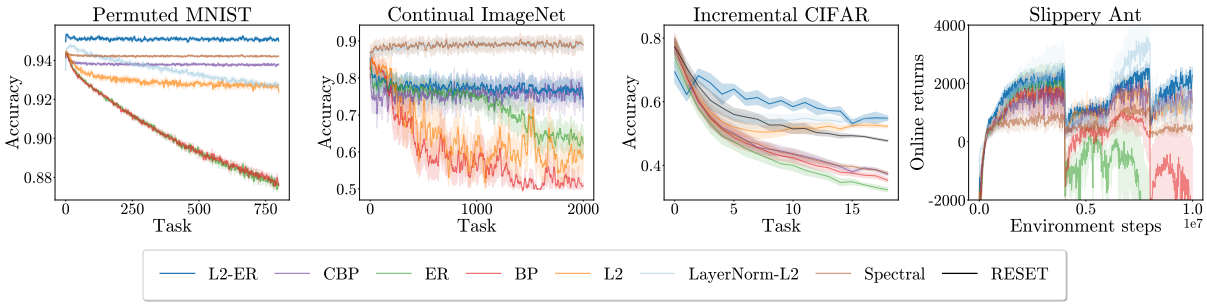


Figure 3.3: Performance across all four environments. Classification accuracy is reported for Permuted MNIST, Continual ImageNet, and Incremental CIFAR, while online returns are reported for Slippery Ant. Results are averaged across 5 seeds for all environments, except Continual ImageNet, where we used 10 seeds due to higher variance. Shaded regions denote a 95% confidence interval.

We conduct experiments across a range of network architectures and continual learning tasks adapted from supervised and reinforcement learning environments.

1. **Permuted MNIST** (Goodfellow et al., 2013): a variant of the MNIST dataset (LeCun, 1998) where new tasks are created by permuting the pixels in all images in the same way. Each permutation represents a new task. We model the classifier with a ReLU MLP.

2. **Continual Imagenet** (Dohare et al., 2024): an adaptation of Imagenet (Krizhevsky et al., 2012) where pairs of classes are randomly sampled for binary classification. Distinguishing the two classes in each new class pair is a new task. We use a convolutional neural network.

3. **Incremental CIFAR** (Dohare et al., 2024): an adaptation of the CIFAR-100 dataset (Krizhevsky et al., 2009) where classes are added incrementally. Starting with 5 classes, each task adds 5 new classes until all 100 classes are included. We use a ResNet architecture He et al. (2015).

4. **Slippery Ant** (Dohare et al., 2024): a continual reinforcement learning environment where the friction between the ant and the ground changes every T timesteps, forcing the agent to continually adapt its policy.

We compare L2-ER against six baselines: vanilla backpropagation (BP), L2 regularization (L2), effective rank regularization (ER), continual backpropagation (Dohare et al., 2024) (CBP), layer-normalization with L2 (Lyle et al., 2024) (LayerNorm-L2), and spectral regularization (Lewandowski et al., 2024a) (Spectral). For Incremental CIFAR, where task difficulty increases with the number of classes, we include a RESET baseline that reinitializes parameters at each task change; algorithms outperforming RESET are considered to maintain plasticity. We perform extensive hyperparameter sweeps and report accuracy for supervised learning and online returns for RL. Implementation details can be found in Appendix G to J of He et al. (2025).

Performance Figure 3.3 depicts results across all environments. L2-ER maintains plasticity in all four environments, and on Permuted MNIST provides an early performance boost before plasticity loss would typically occur. L2 preserves plasticity in most cases but fails on Continual ImageNet; CBP succeeds except on Incremental CIFAR. LayerNorm-L2 performs well except on Permuted MNIST, while Spectral Regularization succeeds except on Incremental CIFAR but requires roughly twice the training time. ER and BP exhibit plasticity loss across all environments.

CHAPTER 4

Outlook

This thesis leaves several directions for future work. A first direction is to study policy optimization and plasticity jointly. The bi-level view of actor-critic learning suggests that policy improvement depends heavily on the quality of the critic's response. In continual settings, however, the critic is also trained over a long sequence of changing distributions. If the critic loses plasticity or becomes poorly conditioned, then the actor may receive weaker learning signals, making policy improvement less stable. Understanding how plasticity affects the actor-critic loop is therefore an important step toward continual reinforcement learning.

A second direction is to design optimization algorithms that explicitly preserve long-term trainability. Most learning algorithms optimize performance on the current task or current environment. In continual learning an update should also be evaluated by how it affects future adaptation. The spectral perspective studied in this thesis suggests one possible way to formalize this problem. If collapse in the Hessian spectrum is associated with reduced adaptability, then future methods could regularize the optimization landscape directly by preserving curvature, feature diversity, or effective rank during training.

A third direction is to better understand the role of representations in continual reinforcement learning. Deep RL agents do not only learn policies and value functions but they also

learn internal representations of states, dynamics, and rewards. If these representations become too specialized to early tasks, then later adaptation may become difficult even when the network still has enough parameters. Future work could study how representation geometry changes across tasks, and how to learn representations that are useful for the current task while remaining flexible for future tasks.

Finally, continual reinforcement learning needs stronger empirical benchmarks. Many reinforcement learning benchmarks still evaluate final performance in a fixed environment, while many continual learning benchmarks focus on supervised classification. To study continual decision making more directly, we need benchmarks where agents face long sequences of changing tasks, dynamics, and state distributions. Such benchmarks should measure not only performance, but also whether the agent remains able to learn after many updates.

Bibliography

Abbas, Z., Zhao, R., Modayil, J., White, A., and Machado, M. C. (2023). Loss of plasticity in continual deep reinforcement learning. In *Conference on lifelong learning agents*, pages 620–636. PMLR.

Afcsar, S., Brotcorne, L., Marcotte, P., and Savard, G. (2016). Achieving an optimal trade-off between revenue and energy peak within a smart grid environment. *Renewable Energy*, 91:293–301.

Ash, J. and Adams, R. P. (2020). On warm-starting neural network training. *Advances in neural information processing systems*, 33:3884–3894.

Baker, M. M., New, A., Aguilar-Simon, M., Al-Halah, Z., Arnold, S. M., Ben-Iwhiwhu, E., Brna, A. P., Brooks, E., Brown, R. C., Daniels, Z., et al. (2023). A domain-agnostic approach for characterization of lifelong learning systems. *Neural Networks*, 160:274–296.

Bakhtin, A., Brown, N., Dinan, E., Farina, G., Flaherty, C., Fried, D., Goff, A., Gray, J., Hu, H., et al. (2022). Human-level play in the game of diplomacy by combining language models with strategic reasoning. *Science*, 378(6624):1067–1074.

Bard, J. F., Plummer, J., and Sourie, J. C. (1998). Determining tax credits for converting nonfood crops to biofuels: an application of bilevel programming. *Multilevel Optimization: Algorithms and Applications*, pages 23–50.

Bjorck, J., Gomes, C. P., and Weinberger, K. Q. (2021). Is high variance unavoidable in rl? a case study in continuous control. *arXiv preprint arXiv:2110.11222*.

Castro, D. D. and Meir, R. (2010). A convergent online single time scale actor critic algorithm. *The Journal of Machine Learning Research*, 11:367–410.

Chakraborty, S., Bedi, A. S., Koppel, A., Wang, H., Manocha, D., Wang, M., and Huang, F. (2024). Parl: A unified framework for policy alignment in reinforcement learning from human feedback. In *ICLR*.

Dagréou, M., Ablin, P., Vaiter, S., and Moreau, T. (2024). How to compute hessian-vector products? In *ICLR Blogposts 2024*. https://iclr-blogposts.github.io/2024/blog/bench-hvp/.

Delfosse, Q., Schramowski, P., Mundt, M., Molina, A., and Kersting, K. (2021). Adaptive rational activations to boost deep reinforcement learning. *arXiv preprint arXiv:2102.09407*.

Dempe, S. (2018). *Bilevel optimization: theory, algorithms and applications*, volume 3. TU Bergakademie Freiberg, Fakultät für Mathematik und Informatik Freiberg ....

Ding, M., Chakraborty, S., Agrawal, V., Che, Z., Koppel, A., Wang, M., Bedi, A., and Huang, F. (2024). Sail: Self-improving efficient online alignment of large language models. *arXiv preprint arXiv:2406.15567*.

Doan, T., Bennani, M. A., Mazoure, B., Rabusseau, G., and Alquier, P. (2021). A theoretical analysis of catastrophic forgetting through the ntk overlap matrix. In *International Conference on Artificial Intelligence and Statistics*, pages 1072–1080. PMLR.

Dohare, S., Hernandez-Garcia, J. F., Lan, Q., Rahman, P., Mahmood, A. R., and Sutton, R. S. (2024). Loss of plasticity in deep continual learning. *Nature*, 632(8026):768–774. Publisher: Nature Publishing Group.

Drineas, P., Mahoney, M. W., and Cristianini, N. (2005). On the nyström method for approximating a gram matrix for improved kernel-based learning. *journal of machine learning research*, 6(12).

Fiez, T., Chasnov, B., and Ratliff, L. (2020). Implicit Learning Dynamics in Stackelberg Games: Equilibria Characterization, Convergence Analysis, and Empirical Study. In *Proceedings*

of the 37th International Conference on Machine Learning, pages 3133–3144. PMLR. ISSN: 2640-3498.

Frangella, Z., Tropp, J. A., and Udell, M. (2023). Randomized nyström preconditioning. *SIAM Journal on Matrix Analysis and Applications*, 44(2):718–752.

Freeman, C. D., Frey, E., Raichuk, A., Girgin, S., Mordatch, I., and Bachem, O. (2021). Brax - a differentiable physics engine for large scale rigid body simulation.

Ghadimi, S. and Wang, M. (2018). Approximation methods for bilevel programming. *arXiv preprint arXiv:1802.02246*.

Ghorbani, B., Krishnan, S., and Xiao, Y. (2019). An investigation into neural net optimization via hessian eigenvalue density. In *International Conference on Machine Learning*, pages 2232–2241. PMLR.

Godfrey, L. B. (2019). An evaluation of parametric activation functions for deep learning. In *2019 IEEE international conference on systems, man and cybernetics (SMC)*, pages 3006–3011. IEEE.

Goodfellow, I. J., Mirza, M., Xiao, D., Courville, A., and Bengio, Y. (2013). An empirical investigation of catastrophic forgetting in gradient-based neural networks. *arXiv preprint arXiv:1312.6211*.

Haarnoja, T., Zhou, A., Abbeel, P., and Levine, S. (2018). Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International conference on machine learning*, pages 1861–1870. PMLR.

Hataya, R. and Yamada, M. (2023). Nyström method for accurate and scalable implicit differentiation. In *International Conference on Artificial Intelligence and Statistics*, pages 4643–4654. PMLR.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. corr abs/1512.03385 (2015).

He, N., Guo, K., Prakash, A., Tiwari, S., Tao, R. Y., Serapio, T., Greenwald, A., and Konidaris, G. (2025). Spectral collapse drives loss of plasticity in deep continual learning. *arXiv preprint arXiv:2509.22335*.

Hestenes, M. R., Stiefel, E., et al. (1952). *Methods of conjugate gradients for solving linear systems*, volume 49. NBS Washington, DC.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2017). Gans trained by a two time-scale update rule converge to a local nash equilibrium. *Advances in neural information processing systems*, 30.

Igl, M., Farquhar, G., Luketina, J., Boehmer, W., and Whiteson, S. (2020). Transient non-stationarity and generalisation in deep reinforcement learning. *arXiv preprint arXiv:2006.05826*.

Jacot, A., Gabriel, F., and Hongler, C. (2018). Neural tangent kernel: Convergence and generalization in neural networks. *Advances in neural information processing systems*, 31.

Javed, K. and Sutton, R. S. (2024). The big world hypothesis and its ramifications for artificial intelligence. In *Finding the Frame: An RLC Workshop for Examining Conceptual Frameworks*.

Ji, K., Yang, J., and Liang, Y. (2021). Bilevel optimization: Convergence analysis and enhanced design. In *International conference on machine learning*, pages 4882–4892. PMLR.

Juliani, A. and Ash, J. (2024). A study of plasticity loss in on-policy deep reinforcement learning. *Advances in Neural Information Processing Systems*, 37:113884–113910.

Kirkpatrick, J., Pascanu, R., Rabinowitz, N., Veness, J., Desjardins, G., Rusu, A. A., Milan, K., Quan, J., Ramalho, T., Grabska-Barwinska, A., et al. (2017). Overcoming catastrophic forgetting in neural networks. *Proceedings of the National Academy of Sciences*, 114(13):3521–3526.

Klein, T., Miklautz, L., Sidak, K., Plant, C., and Tschiatschek, S. (2024). Plasticity loss in deep reinforcement learning: A survey.

Krizhevsky, A., Hinton, G., et al. (2009). Learning multiple layers of features from tiny images.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

Lange, R. T. (2022). gymnax: A JAX-based reinforcement learning environment library.

LeCun, Y. (1998). The mnist database of handwritten digits. *http://yann. lecun. com/exdb/mnist/*.

Lewandowski, A., Bortkiewicz, M., Kumar, S., Schuurmans, D., Ostaszewski, M., Machado, M. C., et al. (2024a). Learning continually by spectral regularization. *arXiv preprint arXiv:2406.06811*.

Lewandowski, A., Tanaka, H., Schuurmans, D., and Machado, M. C. (2024b). Directions of curvature as an explanation for loss of plasticity.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. (2019). Continuous control with deep reinforcement learning.

Lorraine, J., Vicol, P., and Duvenaud, D. (2020). Optimizing millions of hyperparameters by implicit differentiation. In *International conference on artificial intelligence and statistics*, pages 1540–1552. PMLR.

Lu, C., Kuba, J., Letcher, A., Metz, L., Schroeder de Witt, C., and Foerster, J. (2022). Discovered policy optimisation. *Advances in Neural Information Processing Systems*, 35:16455–16468.

Lyle, C., Zheng, Z., Khetarpal, K., van Hasselt, H., Pascanu, R., Martens, J., and Dabney, W. (2024). Disentangling the causes of plasticity loss in neural networks. *arXiv preprint arXiv:2402.18762*.

Lyle, C., Zheng, Z., Nikishin, E., Pires, B. A., Pascanu, R., and Dabney, W. (2023). Understanding plasticity in neural networks.

Montufar, G. F., Pascanu, R., Cho, K., and Bengio, Y. (2014). On the number of linear regions of deep neural networks. *Advances in neural information processing systems*, 27.

Obando-Ceron, J., Courville, A., and Castro, P. S. (2024). In value-based deep reinforcement learning, a pruned network is a good network. *arXiv preprint arXiv:2402.12479*.

Paige, C. C. and Saunders, M. A. (1975). Solution of sparse indefinite systems of linear equations. *SIAM journal on numerical analysis*, 12(4):617–629.

Parisi, G. I., Kemker, R., Part, J. L., Kanan, C., and Wermter, S. (2018). Continual lifelong learning with neural networks: A review. *CoRR*, abs/1802.07569.

Pearlmutter, B. A. (1994). Fast exact multiplication by the hessian. *Neural computation*, 6(1):147–160.

Prakash, A., He, N., Goktas, D., Makar-Limanov, J., and Greenwald, A. (2025). Bi-level policy optimization with nystr\" om hypergradients. *arXiv preprint arXiv:2505.11714*.

Rajeswaran, A., Finn, C., Kakade, S. M., and Levine, S. (2019). Meta-learning with implicit gradients. *Advances in neural information processing systems*, 32.

Roy, O. and Vetterli, M. (2007). The effective rank: A measure of effective dimensionality. In *2007 15th European signal processing conference*, pages 606–610. IEEE.

Saarinen, S., Bramley, R., and Cybenko, G. (1993). Ill-conditioning in neural network training problems. *SIAM Journal on Scientific Computing*, 14(3):693–714.

Sagun, L., Bottou, L., and LeCun, Y. (2016). Eigenvalues of the hessian in deep learning: Singularity and beyond. *arXiv preprint arXiv:1611.07476*.

Schulman, J., Levine, S., Moritz, P., Jordan, M. I., and Abbeel, P. (2015). Trust region policy optimization. *CoRR*, abs/1502.05477.

Schulman, J., Moritz, P., Levine, S., Jordan, M., and Abbeel, P. (2018). High-dimensional continuous control using generalized advantage estimation.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. (2017). Proximal policy optimization algorithms. *CoRR*, abs/1707.06347.

Shang, W., Sohn, K., Almeida, D., and Lee, H. (2016). Understanding and improving convolutional neural networks via concatenated rectified linear units. In *international conference on machine learning*, pages 2217–2225. PMLR.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., Bolton, A., et al. (2017). Mastering the game of go without human knowledge. *nature*, 550(7676):354–359.

Siméoni, O., Vo, H. V., Seitzer, M., Baldassarre, F., Oquab, M., Jose, C., Khalidov, V., Szafraniec, M., Yi, S., Ramamonjisoa, M., et al. (2025). Dinov3. *arXiv preprint arXiv:2508.10104*.

Sinha, A., Fang, F., An, B., and Kiekintveld, C. (2018). Stackelberg security games: Looking beyond a decade of success. In *Proceedings of the Twenty-Seventh International Joint Conference on Artificial Intelligence, IJCAI-18*, pages 5704–5710.

Sokar, G., Agarwal, R., Castro, P. S., and Evci, U. (2023). The dormant neuron phenomenon in deep reinforcement learning. In *International Conference on Machine Learning*, pages 32145–32168. PMLR.

Sutton, R. S. and Barto, A. G. (2018). Reinforcement learning: An introduction. *A Bradford Book*.

Vicol, P., Lorraine, J. P., Pedregosa, F., Duvenaud, D., and Grosse, R. B. (2022). On Implicit Bias in Overparameterized Bilevel Optimization. In *Proceedings of the 39th International Conference on Machine Learning*, pages 22234–22259. PMLR. ISSN: 2640-3498.

Wen, J., Kumar, S., Gummadi, R., and Schuurmans, D. (2021). Characterizing the gap between actor-critic and policy gradient. In *International Conference on Machine Learning*, pages 11101–11111. PMLR.

Zheng, L., Fiez, T., Alumbaugh, Z., Chasnov, B., and Ratliff, L. J. (2022). Stackelberg actor-critic: Game-theoretic reinforcement learning algorithms. In *Proceedings of the AAAI conference on Artificial Intelligence*, volume 36, pages 9217–9224.