Subspaces of Policies for Deep Reinforcement Learning

Jean-Baptiste Gaya

THÈSE DE DOCTORAT DE SORBONNE UNIVERSITÉ

Spécialité **Informatique**

École Doctorale Informatique, Télécommunications et Électronique (Paris)

Subspaces of Policies for Deep Reinforcement Learning

Sous-Espaces de Politiques pour l'Apprentissage par Renforcement Profond

Présentée par

Jean-Baptiste GAYA

Dirigée par

Ass. Pr. Laure SOULIER (HDR)

Co-encadrée par

Alessandro LAZARIC

Ludovic DENOYER

Pour obtenir le grade de

DOCTEUR de SORBONNE UNIVERSITÉ

Présentée et soutenue publiquement le 26 avril 2024 devant le jury composé de

Pr. Balázs Kégl *Head of AI Research, Huawei Technologies*	Rapporteur
Dir. Pierre-Yves Oudeyer *Directeur de Recherche, Inria Université de Bordeaux*	Rapporteur
Pr. Eric Gaussier *Professeur, Université Grenoble Alpes*	Examinateur
Pr. Katja Hofmann *Senior Principal Researcher, Microsoft*	Examinatrice
Pr. Marc'Aurelio Ranzato *Research director, Google Deepmind*	Examinateur
Pr. Olivier Sigaud *Professeur, ISIR Sorbonne Université*	Président du jury
Ass. Pr. Laure Soulier *Maître de Conférences HDR, ISIR Sorbonne Université*	Directrice de Thèse
Alessandro Lazaric *Research Scientist, Meta AI*	Co-encadrant
Pr. Ludovic Denoyer *Research Scientist, Ubisoft*	Co-encadrant

CONTENTS

CONTENTS iii

RÉSUMÉ EN FRANÇAIS v

1 Contexte et Motivations v

2 Sous-espaces des Politiques x

3 Autres Contributions et Conclusion xii

I Context & Motivations 1

1 INTRODUCTION 3

1.1 Context 3

1.2 Outline & Contributions 6

2 BACKGROUND 11

2.1 Deep reinforcement Learning 11

2.2 Zero-shot Generalization 19

2.3 Few-shot Adaptation 24

2.4 Continual Reinforcement Learning 28

2.5 Conclusion 36

II Zero-shot Generalization 37

3 A PATHOLOGICAL CASE 39

3.1 Context & Motivations 39

3.2 Problem Statement 41

3.3 Models 44

3.4 Experimental Protocol 49

3.5 Results 53

3.6 Conclusion 58

4 WEIGHT AVERAGING FOR MULTI-OBJECTIVE REINFORCEMENT LEARNING 61

4.1 Context & Motivations 62

4.2 Rewarded Soup 65

4.3 Experiments 68

4.4 Conclusion 79

III Subspace of Policies 81

5 DEFINING A SUBSPACE OF POLICIES 83

5.1 Towards a Population of Policies 84

5.2 The history of mode connectivity 86

5.3 Neural Network Subspaces 89

5.4 Neural Network Subspaces of Policies . . . 92
5.5 Conclusion . . . 95
6 LEARNING A LINE OF POLICIES 97
6.1 Context & Motivations . . . 98
6.2 Line of Policies (LoP) . . . 99
6.3 Experimental Protocol . . . 103
6.4 Results & Analysis . . . 105
6.5 Conclusion . . . 115
7 BUILDING A SUBSPACE OF POLICIES 117
7.1 Context & Motivations . . . 119
7.2 Continual Subspace of Policies (CSP) . . . 121
7.3 Experimental Protocol . . . 127
7.4 Results & Analysis . . . 132
7.5 Conclusion . . . 140
IV Conclusion & Perspectives 141
8 OTHER CONTRIBUTIONS 143
8.1 SaLinA . . . 143
8.2 WorldSense . . . 145
8.3 Ongoing work with Llama's team . . . 147
9 CONCLUSION & FUTURE DIRECTIONS 149
9.1 Conclusion . . . 149
9.2 Future Directions . . . 150
BIBLIOGRAPHY 153

RÉSUMÉ EN FRANÇAIS

1 Contexte et Motivations

1.1 Introduction

L'apprentissage par renforcement [1] se concentre sur la prise de décision dans des conditions d'incertitude, avec des agents apprenant à optimiser les actions pour des récompenses cumulatives. Le RL a connu des succès notables, maîtrisant des tâches complexes telles que les échecs avec Deep Blue (Campbell et al. 2002) et le jeu de Go avec AlphaGo (Silver et al. 2017). Bien que son potentiel dans ces jeux soit évident, l'application du Reinforcement Learning dans des domaines comme la robotique et les systèmes autonomes reste largement exploratoire.

Le début de ma thèse a coïncidé avec un changement dans l'IA : le récent "boom de l'IA" a orienté une partie importante de la recherche en Deep Learning vers la mise à l'échelle des méthodes existantes, telles que les architectures de transformateurs (Vaswani et al. 2017), les modèles autorégressifs pour le traitement du langage naturel (Brown et al. 2020b), et les modèles de diffusion pour la vision par ordinateur (Ho et al. 2020). Le Deep Reinforcement Learning a rapidement suivi cette tendance, s'inspirant de ces innovations en architecture (par exemple, Decision Transformers (L. Chen et al. 2021)), méthodologie (par exemple, Voyager (G. Wang et al. 2023) utilisant la chaîne de pensée avec chatGPT), et paradigme : plus de données, de plus grands modèles, des problèmes plus larges, essentiellement en augmentant l'échelle pour atteindre la *généralisation* - une capacité vaguement définie de s'adapter à des données ou des tâches inédites. L'émergence de l'Offline Reinforcement Learning (Levine et al. 2020), visant à traiter de grands lots de données, témoigne de cette tendance.

1. Le jargon relatif à l'apprentissage par renforcement est, dans la mesure du possible, traduit en français. Néanmoins certains mots - écrits en *italique* - sont laissés en anglais car difficilement traduisibles.

Malgré des progrès indéniables dans la recherche, les applications pratiques restent limitées en raison de plusieurs facteurs. Si l'on considère le Deep Learning comme l'apprentissage d'une fonction f mappant les caractéristiques x à une sortie y, la dimension temporelle dans le Reinforcement Learning élargit exponentiellement le domaine des possibles (impliquant une *séquence* de décisions optimales). De plus, la rareté des données "d'expert" dans des domaines comme la robotique limite l'applicabilité des mêmes approches utilisées pour les grands modèles de langage. Enfin, le concept de généralisation dans ces problèmes est plus large que dans l'apprentissage supervisé : il ne s'agit pas seulement de résumer un nouveau texte, mais de résoudre une nouvelle tâche, parfois d'élargir l'espace d'action, ou même de changer entièrement de domaines (comme sim-to-real en robotique).

Ces considérations m'ont naturellement conduit à étudier les méthodes *adaptatives* tout au long de ma thèse : au lieu de s'attaquer à la tâche redoutable de la généralisation, nous supposons que notre agent a du *temps libre* pour réussir une nouvelle tâche. Ce changement de paradigme n'est pas seulement une simplification du problème de généralisation ; il s'aligne également plus étroitement sur le comportement humain. Face à un nouveau casse-tête, un enfant devra nécessairement explorer, tester et s'adapter, même si ses connaissances antérieures l'aident à comprendre rapidement le problème.

La méthode des *Sous-espaces de Politiques* développée dans cette thèse est une tentative de combiner l'acquisition de connaissances et l'adaptabilité dans un modèle unique et polyvalent. Nous l'appliquerons dans divers paramètres d'apprentissage par renforcement et tirerons des conclusions intéressantes pour l'avenir de l'adaptation dans l'apprentissage par renforcement.

Cette thèse se structure en trois parties distinctes. La première partie, *Contexte et Motivations* (Section 1), pose les bases, présentant un aperçu des paradigmes du RL et introduisant la généralisation morphologique à zéro coup. La deuxième partie, *Sous-espaces de Politiques* (Section 2), forme le cœur de la thèse, détaillant le développement et l'application des sous-espaces de politiques dans divers contextes de RL, y compris l'adaptation rapide et l'apprentissage continu. La troisième partie, *Conclusion* (Section 3), résume les contributions et esquisse les orientations futures pour la recherche dans ce domaine.

1.2 Fondements

Dans l'apprentissage par renforcement, un agent développe une stratégie pour maximiser une fonction de récompense. Il utilise une politique pour l'aider à définir sa stratégie. Pour cela, il intéragit avec un environnement et utilise un algorithme pour l'optimiser. On peut le modéliser par un Processus de Décision Markovienne (MDP). L'apprentissage par renforcement profond (RL) étend cette approche en utilisant des réseaux de neurones profonds comme politique pour traiter de grands espaces d'état ou d'action. Les types d'algorithme sont catégorisées de cette manière:

- Algorithmes *on-policy* : Ces algorithmes, comme A2C (Mnih et al. 2016) et PPO (Schulman et al. 2017a), apprennent directement et exclusivement à partir des expériences de la politique actuelle. Ils sont efficaces pour l'apprentissage immédiat mais moins efficaces en termes d'échantillonnage comparés à d'autres méthodes.

- Algorithmes *off-policy* : Les algorithmes comme SAC (Haarnoja et al. 2018a) utilisent un *replay buffer* contenant un éventail d'expériences passées. Ils se distinguent par une meilleure efficacité d'échantillonnage et sont adaptés pour des environnements où les interactions sont coûteuses.

- **RL hors ligne** : Cette approche, représentée par des méthodes comme le *behaioral cloning* (Bain and Sammut 1995), se concentre sur l'apprentissage à partir d'un ensemble fixe d'expériences sans interaction supplémentaire avec l'environnement. Elle est utile dans des scénarios où l'exploration est coûteuse ou risquée, mais elle est limitée par sa capacité de généralisation dans des situations inédites.

Voici les différents paradigmes que nous allons aborder dans cette thèse:

Le domaine de la généralisation *zero-shot* (ZSG) en apprentissage par renforcement, dont l'objectif est de développer des algorithmes capables de fonctionner efficacement dans de nouveaux environnements inconnus sans formation supplémentaire. Cette approche nécessite des politiques robustes adaptées aux variations environnementales, utilisant des *processus*

de décision markovienne contextuels (CMDPs) pour gérer les changements dans les dynamiques et les récompenses. Nous l'aborderons en Section 1.3.

L'adaptation *few-shot*, qui est un sous-domaine du RL visant à permettre aux agents de s'adapter rapidement à de nouvelles tâches avec des interactions limitées. Inspiré de l'apprentissage humain, où nous généralisons souvent à partir de quelques exemples, ce problème a été étudié sous diverses terminologies, mais celle que nous aborderons en Section 2.3 est simple : l'agent a le droit à k épisodes pour s'adapter à un nouvel environnement qu'il n'a jamais vu, sans possibilité d'optimiser sa politique.

Nous explorerons aussi l'apprentissage par renforcement en continu (CRL) en Section 2.4, un domaine qui pousse le développement d'agents autonomes à évoluer et à relever une infinité de défis. Le CRL se concentre sur l'apprentissage continu à travers une séquence de tâches, chaque tâche étant un MDP stationnaire avec ses propres règles. Notre intérêt se porte sur des scénarios réalistes où les tâches partagent certains éléments, les changements affectant plutôt la fonction de transition et la fonction de récompense). Les méthodes existantes en CRL visent à éviter "l'oubli catastrophique" et à favoriser le transfert à travers la séquence sans sacrifier la mise à l'échelle. Nos travaux abordent ces défis en équilibrant la taille du modèle et la performance, plutôt que des compromis typiques menés par les transferts vers l'avant et vers l'arrière. Ce chapitre éclaircira comment nos méthodes abordent le problème.

1.3 Un Cas Pathologique : Généralisation Morphologique en Contrôle Continu

Cette section s'inspire d'un travail en cours intitulé "Advancing zero-shot Morphological Generalization", abordant les défis de la généralisation morphologique. Nous proposons une combinaison de techniques de RL hors ligne et en ligne, utilisant PPO (Schulman et al. 2017a) pour la génération de trajectoires d'experts et BC (Bain and Sammut 1995) pour l'entraînement. Nous introduisons le "Massive Ant-based Dataset" (MAD), comprenant 7000 variations morphologiques d'une morphologie générique *ant*. Les résultats indiquent que l'élargissement de la diversité des données améliore la capacité de généralisation des modèles, même pour des données non représentées dans l'ensemble d'entraînement.

Une étude systématique sur la généralisation morphologique a été menée, modulant le nombre de morphologies uniques (70, 700, 7000) et le nombre de trajectoires par morphologie (1, 10, 100). L'analyse révèle que des régimes avec un plus grand nombre de morphologies et moins de trajectoires par morphologie optimisent la capacité de généralisation, en particulier pour les modèles formés sur 7000 morphologies avec 10 trajectoires chacune. Ces constatations soulignent l'importance de la diversité morphologique dans les ensembles de données pour une meilleure performance générale des modèles.

Concernant l'architecture des modèles, nous proposons une architecture *GTR*, inspirée de Gato (Reed et al. 2022). Elle a démontré une performance supérieure par rapport aux modèles de base dans divers scénarios.

Nos recherches révèlent également que l'intégration excessive de contexte peut mener à un surapprentissage, compromettant la capacité de généralisation - notamment hors distribution - du modèle. Les embeddings contextuels issus de fichiers de configuration bruts (fichiers json par exemple, utilisés par le moteur physique pour générer un environnement), traités par l'architecture Longformer (Beltagy et al. 2020), ont montré des signes de surapprentissage lorsqu'ils étaient excessivement entraînés. Cette observation met en lumière la nécessité d'une approche équilibrée dans l'intégration du contexte, favorisant l'apprentissage sans nuire à la performance du modèle sur des données nouvelles et non représentées.

En somme, cette étude révèle les complexités et les défis liés à la généralisation morphologique en contrôle continu, soulignant l'importance d'une diversité élevée des données et une utilisation judicieuse du contexte pour éviter le surajustement. Elle ouvre également la voie à des recherches futures sur l'efficacité des architectures GTR dans ce domaine.

Ces découvertes suggèrent un changement d'approche : passer de la généralisation *zero-shot* à de l'adaptation *few-shot*, impliquant un changement d'approche radical. Dans la partie suivante, nous définissons et explorons les sous-espaces de politiques, une nouvelle manière de régler ce problème.

2 Sous-espaces des Politiques

2.1 Définition

Les *Sous-espaces de Politiques* se basent sur la notion de *mode connectivity* dans l'optimisation des réseaux de neurones. Inspiré par le travail de Wortsman et al. 2021, cette approche que nous avons mis au point considère les combinaisons de poids de différents réseaux comme un moyen de trouver des solutions plus diverses our robustes. Par exemple, on peut considerer le sous-espace de combinaisons convexes de n réseaux de neurones (qu'on appelle *anchors* car ils ne sont pas voués a être optimisés lorsqu'on cherche dans cet espace), $\{\theta_i\}_{i=1}^n$, dans un espace de dimension d, formant ainsi une enveloppe convexe.

Cette définition met en lumière un moyen innovant d'explorer et d'optimiser les politiques de renforcement à travers un espace de solution plus vaste et connecté, ouvrant de nouvelles voies pour le développement de politiques adaptatives en RL.

2.2 Application à l'apprentissage par renforcement multi-objectif

Cette partie basée sur Ramé et al. 2023 et dont je suis co-auteur, aborde le défi de la spécification des récompenses dans le contexte des préférences utilisateur. En effet, il est difficile de façonner des fonction de récompense correspondant précisément aux préférences des utilisateurs. Pour surmonter cela, nous optimisons différents modèles sur différentes fonctions de récompenses, mais en partant d'un même modèle pré-entraîné. Ensuite, nous explorons l'enveloppe convexe entre ces modèles pour trouver un modèle qui donne un compromis optimal entre toutes les fonctions de récompense. Cette approche permet de naviguer efficacement entre divers objectifs, offrant ainsi une solution flexible pour aligner les modèles sur un éventail de préférences utilisateur. Nous appliquons cette méthode à de nombreux domaines et modes : génération de texte, d'image, contrôle continu. Nous montrons que cette méthode donne des résultats similaires ou meilleurs que les méthodes de multi-objectif classique.

2.3 Apprentissage d'une Ligne de Politiques

Dans cette section basée sur Gaya et al. 2021, nous explorons l'adaptation *few-shot* dans le contexte de l'apprentissage par renforcement, en utilisant la méthode des *Subspaces of Policies*. L'approche, nommée Ligne de Politiques (LoP), emploie une combinaison linéaire de plusieurs politiques optimales mais différentes, offrant un spectre de solutions pour l'adaptation dans des environnements inconnus. LoP se distingue par sa simplicité, évitant la nécessité de peaufinements ou d'architectures complexes, tout en reconnaissant certaines limitations en termes de diversité fonctionnelle.

Nos expériences démontrent l'efficacité de LoP dans divers environnements, où il surpasse d'autres méthodes en termes de performance dans des environnements de test variés. Une caractéristique clé de LoP est sa faible sensibilité aux hyper-paramètres, en particulier au paramètre β. Cela simplifie le réglage de la méthode et la rend moins dépendante de la sélection manuelle des paramètres par rapport aux méthodes concurrentes.

L'adaptation en ligne de LoP se révèle efficace, permettant d'évaluer et d'ajuster plusieurs politiques rapidement dans de nouveaux environnements. Cette flexibilité en fait une méthode prometteuse pour les scénarios d'apprentissage par renforcement où l'adaptabilité et la capacité à réagir rapidement à de nouveaux défis sont cruciales.

En somme, la méthode LoP offre une stratégie efficace et adaptable pour l'apprentissage par renforcement, particulièrement dans des scénarios d'adaptation rapide à des environnements inédits.

2.4 Construction d'un Sous-espace de Politiques

Dans cette section, basée sur Gaya et al. 2023, nous introduisons une approche innovante dans le contexte de l'apprentissage par renforcement continu (CRL): *Continual Subspace of Policies* (CSP). CSP se distingue par sa capacité à construire de manière itérative et adaptative un sous-espace de politiques, permettant à l'agent d'atteindre des performances élevées dans une séquence de tâches tout en limitant la croissance de la taille du modèle. Cette méthode est évaluée sur 18 scénarios que nous avons façonnés, comprenant 35 tâches de contrôle continu dans les domaines

de locomotion et manipulation robotique, en utilisant les environnements Brax (C. Daniel Freeman et al. 2021b) et Continual World (Wołczyk et al. 2021).

Les résultats montrent que CSP excelle non seulement dans la gestion du transfert entre les tâches, mais également en termes de mémoire: contrairement aux méthodes traditionnelles qui augmentent linéairement en taille avec le nombre de tâches, CSP maintient une taille de modèle beaucoup plus réduite. Cette efficacité est démontrée dans différents scénarios pathologiques. CSP est comparé à plusieurs méthodes de référence en CRL, telles que PNN (Rusu et al. 2016b), EWC (Kirkpatrick et al. 2017), PackNet (Mallya and Lazebnik 2018) et démontre une performance supérieure tout en restant plus efficient en termes de coût mémoire.

Une étude approfondie sur les hyperparamètres indique que CSP peut réduire de manière encore plus significative les coûts de mémoire sans impacter gravement la performance. De plus, CSP affiche une capacité d'adaptation impressionnante dans des séquences de tâches longues et variées basées sur des environnements challengeants comme *Humanoid*. Les résultats sur Continual World (Wołczyk et al. 2021), en particulier sur le benchmark CW10, soulignent l'efficacité de CSP comparé à l'état de l'art actuel PackNet, avec un coût de mémoire réduit.

En résumé, CSP se révèle être une méthode compétitive dans le domaine de l'apprentissage par renforcement continu, offrant un équilibre optimal entre la taille du modèle et la performance, et se positionne comme une solution viable pour des séquences de tâches longues et diversifiées.

3 Autres Contributions et Conclusion

3.1 Autres Contributions

3.1.1 SaLinA

SaLinA (Denoyer et al. 2021) est une bibliothèque Python conçue pour simplifier l'implémentation de modèles d'apprentissage séquentiel, y compris les algorithmes d'apprentissage par renforcement. Elle offre une adaptabilité notable, permettant une utilisation efficace de multiples CPU et GPU.

SaLinA couvre divers scénarios, tels que l'apprentissage par renforcement basé sur des modèles, l'apprentissage par lots, l'apprentissage par renforcement hiérarchique et multi-agents. Elle traite chaque composant comme un agent, simplifiant l'implémentation et le développement de modèles complexes. Ma contribution a impliqué l'enrichissement de SaLinA avec des benchmarks et le développement d'un module d'apprentissage continu.

3.1.2 WorldSense

WorldSense (Benchekroun et al. 2023) est un benchmark conçu pour évaluer la capacité des modèles de langage de grande taille à raisonner de manière ancrée dans la réalité. Il teste leur aptitude à inférer à partir de descriptions d'arrangements simples d'entités, en se concentrant sur la cohérence et la complétude de ces descriptions. Mon rôle a été de configurer et d'optimiser l'entraînement et l'évaluation des modèles Llama, assurant des résultats fiables pour l'évaluation des capacités de raisonnement des modèles.

3.1.3 Travail en Cours avec l'équipe Llama

Je contribue actuellement au développement d'une nouvelle version de Llama avec l'équipe "Gen AI" de Meta AI. Mon rôle comprend l'affinement de la création de datasets et l'élaboration de stratégies de rejet plus avancées, ainsi que l'application des sous-espaces pendant la phase de RLHF. L'objectif est de créer une population de modèles diversifiée, en utilisant l'approche des sous-espaces développée dans la partie précédente.

3.2 Conclusion et perspectives

Cette thèse explore les *Sous-espaces de Politiques* en apprentissage par renforcement, révélant des modèles adaptatifs capables de s'ajuster à des environnements variés. Malgré l'échec de la généralisation classique, cette recherche ouvre une voie prometteuse vers l'adaptation, en créant des sous-espaces de politiques pour une réponse agile et diversifiée.

En termes de perspectives, cela ouvre la voie à des approches modulaires combinant les sous-espaces avec des mixtures d'experts. L'apprentissage actif pourrait enrichir ce cadre, offrant une exploration plus ciblée et efficace.

De plus, on peut également l'idée d'un framework décentralisé, où chaque utilisateur contribue à un modèle centralisé via la moyenne de leurs modèles, présente une avenue fascinante pour des applications personnalisées, respectant la protection des données, mais aussi évolutives (la moyenne change en fonction des joueurs). Ces perspectives ouvrent des horizons passionnants pour l'avenir de l'adaptation et de la personnalisation en IA.

PART I

Context & Motivations

CHAPTER 1

INTRODUCTION

1.1 Context

Reinforcement learning centers on decision-making under uncertainty, with agents learning to optimize actions for cumulative rewards. It has achieved notable successes, mastering complex tasks such as Chess with Deep Blue (Campbell et al. 2002) and Go with AlphaGo (Silver et al. 2017). While its potential in game playing is evident, its application in fields like robotics and autonomous systems is still largely exploratory.

The beginning of my thesis coincided with a shift in AI: the recent "AI boom" has steered a significant portion of deep learning research towards scaling existing methods, such as transformer architectures (Vaswani et al. 2017), autoregressive models for natural language processing (Brown et al. 2020b), and diffusion models for computer vision (Ho et al. 2020). Deep reinforcement learning quickly followed suit, drawing inspiration from these innovations in 1) architecture (e.g., Decision Transformers by L. Chen et al. (2021)); 2) methodology (e.g., Voyager by G. Wang et al. (2023)); 3) paradigm: more data, larger models, broader problems, essentially scaling up to achieve *generalization* (R. Kirk et al. 2023). The emergence of offline reinforcement learning (Levine et al. 2020), aimed at processing large batches of data, is a testament to this trend.

Despite undeniable progress in research, practical applications in reinforcement learning remain limited due to several factors. If we consider supervised learning as learning a function f mapping features x to an output y, the temporal dimension in reinforcement learning exponentially expands the realm of possibilities (involving a *sequence* of optimal decisions). Additionally, the scarcity of "expert" data in fields like robotics limits the applicability of the same approaches used for Large Language Models. Lastly, the concept of generalization in these problems is broader than in super-

vised learning: it's not just about summarizing a new text, but solving a whole new task, sometimes expanding the action space, or even entirely changing domains (like sim-to-real in robotics).

These considerations naturally led me to study *adaptive* methods throughout my thesis: instead of tackling the daunting task of generalization, we assume our *agent* - the decision maker we aim to train - has *free time* to succeed in a new task. This paradigm shift is not just a simplification of the generalization problem; it also aligns more closely with human behavior. Faced with a new puzzle, a child will necessarily need to explore, test, and adapt, even though their prior knowledge helps in quickly grasping the problem. This relaxed assumption paradoxically brings other challenges: how to capitalize on this *free time* to maximize performance on a new task? I was quickly captivated by a simple intuition: if we could test *several* agents with different behaviors during this free time and select the best among them, it would be much simpler. This intuition is also supported by an observation. Let consider a classical deep reinforcement learning task and k initially randomized neural networks whose parameters are in $\mathbb{R}^d$. When trained on this task, the solutions they converge to will naturally be very distant - in the sense of the L_2 norm - in $\mathbb{R}^d$ (Choromanska et al. 2015). But does this *parametric* diversity lead to *functional* diversity (*i.e.* would the outputs of these neural networks vary in a different task)? And how can we train so many neural networks at minimal computation and memory cost?

Having laid out these reflections, the research questions I sought to answer during these three years were as follows: 1) Is deep reinforcement learning mature enough to consider tackling a zero-shot generalization setting? 2) If not, can we envision a simple and intuitive method to facilitate an agent's adaptation to a new task? 3) If so, can this method be generalized in a more *real-world* context where an *autonomous* agent faces a *sequence* of tasks?

The culmination of this research is the creation of a whole new framework, the *subspaces of policies*, to which I dedicate an entire part of my manuscript. By moving beyond the traditional focus on single policy optimization, we explore the potential of maintaining a manifold of policies—a dynamic landscape of solutions—that an agent can navigate to adapt to new challenges seamlessly. The intuition behind this approach is grounded in the realization that a diverse set of policies, characterized within a smaller parametric space, can provide a rich set of behavioral strategies with minimal computational overhead. This methodology leverages *weight interpolation* between neural network parameters, offering a pragmatic yet elegant

solution to the problem of functional diversity in reinforcement learning tasks. This framework represents the core contribution of my thesis, laying the groundwork for many other applications and future research avenues, from decentralized deep learning (Koloskova et al. 2019) and active learning (P. Ren et al. 2021b) to integration with neural architecture search (P. Ren et al. 2021a), underscoring its profound implications for the field.

1.2 Outline & Contributions

This thesis unfolds across *four* main parts. Part I introduces the foundational concepts and settings that are pivotal to my work. Parts II and III contain the core of my contributions, threading through the theme of adaptability in settings: from showcasing the progress and limitations within generalization in reinforcement learning (Part II) to advocating for more adaptive approaches (Part III). This part is so central to my thesis it inspired its title. Part IV concludes the discussion, pointing towards future research avenues. Let's now delve into these chapters and the work underpinning each.

Part I (Context & Motivations) is divided into two chapters laying the foundation of the thesis:

- **Chapter 1 (Introduction)** is the current chapter, where I establish the context of my work and detail its structure.
- **Chapter 2 (Background)** offers an overview of the reinforcement learning paradigms addressed in this thesis, setting the stage for a detailed exploration. It begins with an introduction to Deep Reinforcement Learning and extends into discussions on settings tackled within this thesis. It delves into the "budget" available for mastering new tasks, starting with zero-shot generalization—where an agent must succeed in a task without any budget—progressing through few-shot adaptation—where a budget for interaction or episodes is available before evaluating the agent—and culminating in continual reinforcement learning—where the budget is an integral part of the problem and is incorporated into training as a parameter to consider, as the agent performs across sequences of tasks.

Part II (Zero-shot Generalization) presents my work within the context of zero-shot generalization, a setting detailed in Section 2.2. It chronologically occurred late in the development of my thesis, reflecting a more pronounced shift in problem formulation within the reinforcement learning community: similar to Gato (Reed et al. 2022), this generalist and multimodal agent leverages numerous datasets of various types to tackle tasks in unseen environments in a zero-shot manner. Nonetheless, this setting remains highly ambitious, despite new neural networks architectures and pretraining methods that we will discuss in these two chapters:

- **Chapter 3 (A Pathological Example)** provides a concrete introduction to zero-shot morphological generalization in continuous control. It builds upon ongoing work, highlighting the current shortcomings in deep reinforcement learning methods regarding generalization. While proposing concrete conditions (type of dataset, method, model architecture) for a certain type of generalization to emerge, this work also shows the limits of current methods: far from the generalization to text generation tasks tackled by LLMs.

  Based on *Advancing Zero-shot Morphological Generalization, Jean-Baptiste Gaya, Jonas Gehring, Alessandro Lazaric, Laure Soulier* (ongoing work).

- **Chapter 4 (Weight Averaging for Multi-objective RL)** proposes a novel and straightforward approach to deal with the challenging problem of optimizing multiple objectives in reinforcement learning. It introduces *rewarded soup*, a strategy to trade-off between multiple rewards when fine-tuning foundation models with reinforcement learning from human feedback; we first learn one network for each reward, and then linearly interpolate their weights despite the architecture's non-linearities. This is also a great introduction to the following part which extends the concept of weight averaging towards more sophisticated methods to tackle different settings.

  Based on *Rewarded soups: towards Pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards, Alexandre Rame, Guillaume Couairon, Corentin Dancette, Jean-Baptiste Gaya, Mustafa Shukor, Laure Soulier, Matthieu Cord* (**NeurIPS 2023**)

Part III (Subspaces of Policies) represents the core of my thesis. During the study of the few-shot adaptation setting, we developed an entirely new method derived from *mode connectivity*. We lay the foundations of this method in the first chapter, providing intuition for the idea and its benefits, then experimentally demonstrate how this method can be applied in the next two chapters within different settings, both of which consider the notion of task adaptation. Here are these three chapters:

- **Chapter 5 (Defining a Subspace of Policies)** thus serves as a basis for exploring the history and intuition behind subspaces. It starts with a toy example, then explains the genesis and inspirations of the method before laying down its technical foundations and initial observations.

- **Chapter 6 (Learning a Line of Policies)** is based on my first published paper. The few-shot adaptation setting (which we will develop in Chapter 2) does not allow for traditional optimization methods. Instead, it encourages testing several interesting policies to find the best one. We propose to learn a *line* of policies based on what we established in the previous chapter, thus finding interesting policies without having to train many.

  Based on *Learning a subspace of policies for online adaptation in Reinforcement Learning*, *Jean-Baptiste Gaya, Laure Soulier, Ludovic Denoyer* (**ICLR 2022**)

- **Chapter 7 (Building a Subspace of Policies)** extends the setting and method from the previous chapter to continual reinforcement learning (also described in Chapter 2). This more ambitious setting required rethinking the basic concept of subspaces: it was no longer just about selecting a good policy among others, but also about acquiring and retaining knowledge at each task, without necessarily increasing in size. This is the real problem with state-of-the-art models in this domain in terms of the number of parameters: they either grow too quickly or have a static size and are doomed to saturate. The real novelty of this paper is the adaptability of our method, which only increases in parameters when necessary (*i.e.*, when the task brings new knowledge).

  Based on *Building a Subspace of Policies for Scalable Continual Learning*, *Jean-Baptiste Gaya, Thang Doan, Lucas Caccia, Laure Soulier, Ludovic Denoyer, Roberta Raileanu* (**ICLR 2023 Spotlight**)

Part IV (Conclusion & Perspectives) offers a comprehensive summary of the dissertation, reflecting on the journey through the realms of reinforcement learning, subspaces of policies, and their implications for future AI research. This part not only ties together the various threads of investigation pursued throughout the thesis but also casts a forward-looking perspective on how these contributions can shape the next steps in the field.

- **Chapter 8 (Other Contributions)** showcases the breadth of my research, extending beyond the thesis's main theme into areas like advanced tool development and benchmarking in AI. It highlights my contributions to the SaLinA library, a Python extension for PyTorch designed to simplify sequential learning model implementations, including reinforcement learning. My involvement in WorldSense introduces a benchmark for assessing Large Language Models' grounded reasoning capabilities, emphasizing bias minimization and the promotion of genuine reasoning skills. Additionally, my ongoing collaboration with Meta AI's "Gen AI" team on the development of the next Llama model emphasizes my active role in refining the current state-of-the-art models in this field.

  Based on *SaLinA: Sequential Learning of Agents, Ludovic Denoyer, Alfredo de la Fuente, Song Duong, Jean-Baptiste Gaya, Pierre-Alexandre Kamienny, Daniel H. Thompson*,

  WorldSense: A Synthetic Benchmark for Grounded Reasoning in Large Language Models, Youssef Benchekroun, Megi Dervishi, Mark Ibrahim, Jean-Baptiste Gaya, Xavier Martinet, Grégoire Mialon, Thomas Scialom, Emmanuel Dupoux, Dieuwke Hupkes, Pascal Vincent

- **Chapter 9 (Future Directions)** concludes this work and outlines future research directions. This chapter synthesizes the main findings of the thesis, emphasizing their implications for the field of reinforcement learning, particularly in subspaces of policies and their applications to continual learning. It will discuss the broader impacts of these methods on AI, highlighting potential societal considerations. The chapter will then transition to future research directions, proposing specific areas where the concepts developed in this thesis could be expanded or applied in novel contexts. Potential directions include enhancing the scalability of subspace methods, exploring their potential in a *de-*

centralized setting, and addressing challenges related to the efficiency and interpretability of these approaches. The objective is to provide a comprehensive outlook on how the groundwork laid by this thesis can contribute to and inspire future advancements in AI research.

CHAPTER 2

BACKGROUND

This chapter lays the foundation and context for the various themes explored in this thesis. The aim here is not to provide an exhaustive state-of-the-art review or a crash course in reinforcement learning. Rather, it is to offer the key insights needed to understand the challenges and issues addressed in our work: we have delved into very different settings, each with its context, challenges, metrics, and state-of-the-art methodologies. Section 2.1 broadly introduces the reinforcement learning framework and the principal algorithms we will employ in subsequent chapters. We then explore specific problems currently central to the reinforcement learning community:, zero-shot generalization (Section 2.2), helpful for framing the context of Chapters 3 and 4, few-shot adaptation (Section 2.3) as the basis for Chapter 6, and continual reinforcement learning (Section 2.4) that set an ambitious setting for Chapter 7.

2.1 Deep reinforcement Learning

> *In uncharted territory, an agent must be able to learn from its own experience.*
>
> Richard Sutton & Andrew Barto,
> Reinforcement Learning: An Introduction

This quote from Richard S. Sutton and Barto (1998) reflects what differentiates Reinforcement Learning from classical Machine Learning: the ability of an *agent* (*i.e.* a decision-maker) to operate and learn optimal behaviors within an unknown environment through trial-and-error interactions, rather than from a static dataset. Unlike supervised learning where the learning process is guided by a labeled dataset, in RL, an agent learns to

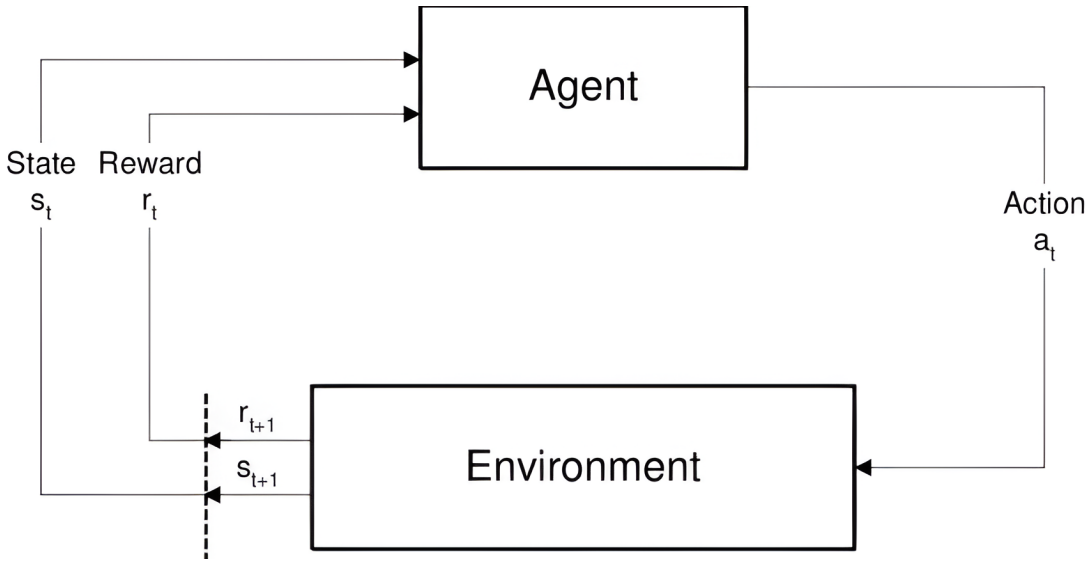


Figure 2.1 – Illustration of the RL paradigm in Richard S. Sutton and Barto (1998).

.

make decisions by receiving rewards or penalties from the dynamic environment it interacts with. This paradigm of learning enables the agent to develop a policy (*i.e.* a strategy, often represented by a probabilistic model, that dictates the agent's decisions or actions in various states) that maximizes the cumulative reward over time, essentially learning to predict which actions will yield the most beneficial outcomes based on its current state. It is often modeled as a Markov Decision Process (MDP), where the agent interacts with an *environment* by selecting actions, then receiving new observations from the environment.

Formally, an MDP is defined by:

- a set of *states* $\mathcal{S}$ that encompasses all possible situations the agent can encounter, serving as the decision-making context.
- a set of *actions* $\mathcal{A}$ that includes all possible moves or decisions the agent can make in response to a state.
- a *transition function* $p : \mathcal{S} \times \mathcal{A} \times \mathcal{S} \to [0, 1]$ that specifies the probability of reaching a subsequent state s' from a current state s by taking an action a, encapsulating the dynamics of the environment.

- a *reward function* $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$ that quantifies the immediate benefit of taking an action a in a state s, guiding the agent towards beneficial outcomes.

An agent's policy $\pi : \mathcal{S} \times \mathcal{A} \to [0, 1]$ defines a strategy by mapping states to probabilities of selecting each action. Acknowledging system dynamics, where the next state s_{t+1} follows $p(\cdot|s_t, a_t)$ and actions a_t adhere to policy $\pi(\cdot|s_t)$, the objective of reinforcement learning is to find π that maximizes the *expected discounted sum of rewards*, represented formally as:

$$\max_{\pi} \mathbb{E}_{\pi,p} \left[\sum_{t=1}^{H} \delta^{t-1} \cdot r(s_t, a_t) \right] \tag{2.1}$$

where t denotes the *timestep*, s_t is the *state* at time t, a_t is the *action* taken at time t, δ is the *discount rate* [1] reflecting the present value of future rewards, and $H \in \mathbb{N}^*$ is the planning horizon. The optimal policy π^* achieves the maximum expected return from any initial state. This formal framework is illustrated in Figure 2.1, providing a visual representation of the RL paradigm as outlined by Richard S. Sutton and Barto (1998), highlighting the interaction between an agent and its environment through states, actions, and rewards.

Deep Reinforcement Learning (RL) extends this framework by using Deep Neural Networks (NN) (Rumelhart et al. 1986) as function approximators to handle large state or action spaces, enabling agents to learn policies from high-dimensional sensory input. This integration allows RL to solve complex sequential decision-making problems that are challenging for traditional approaches. There exist several categorizations of RL paradigms (Model-Based (Richard S Sutton 1990) versus Model-free (Watkins and Dayan 1992), Single-Agent (Richard S. Sutton and Barto 1998) versus Multi-agent (Littman 1994), exploration-exploitation trade-off (Ishii et al. 2002)), but the one that will be widely used in this thesis focuses on the *algorithmic* part that reflects a gradient from interactive, trial-and-error learning approaches to more static, dataset-driven strategies. We can indeed categorize the learning strategies into on-policy (Richard S. Sutton and Barto 1998),

1. While serving as a mechanism to model the preference for immediate rewards over future rewards, it also acts as an important hyperparameter to tune during experiments (Reddy et al. 2016). Despite its significance, we will often omit this factor in equations in the rest of the thesis for simplicity.

off-policy (Richard S. Sutton and Barto 1998), and offline methods (Levine et al. 2020). This distinction is crucial as it underpins our strategic choices regarding algorithm selection for specific tasks within this thesis. For instance, online RL methods are preferred for tasks involving data collection in dynamic environments (e.g. Section 3.4.1), whereas offline RL is suitable to evaluate neural network architectures and isolate them from other hyperparameters, ensuring a focused analysis on the effectiveness of these architectures in static contexts (e.g. Section 3.2). These characteristics are summed up in Table 2.1.We now proceed to delve into each of these categories with their characteristics and applications within the thesis.

	On-policy RL	**Off-policy RL**	**Offline RL**
Sampling Cost	high	small	N/A
Update Cost	Moderate	High	Low
Bias-Variance Trade-off	high variance	mixed	high bias
Data Source	current policy	any policy	experts only

Table 2.1 – Comparison of the different RL paradigms (algorithmic view).

2.1.1 On-policy algorithms

The advent of *GPU-based simulations*, exemplified by recent physics engines like Nvidia's Isaac Gym (Makoviychuk et al. 2021) and Google's Brax (C. Daniel Freeman et al. 2021b), opened a new era for efficiently generating data for AI training in fields such as video games and virtual reality. These platforms enable rapid and cost-effective rollouts of complex simulations, shifting the bottleneck from *data acquisition* to the efficiency of *learning updates*. This paradigm shift emphasizes the need for algorithms that can swiftly adapt to the rich data provided, marking a significant evolution in how real-world problems are approached in the gaming and VR industries. On-policy algorithms, particularly efficient in such settings, capitalize on the relative affordability of rollouts to refine agent capabilities through focused and efficient policy optimization.

These algorithms, such as Advantage Actor-Critic (A2C) (Mnih et al. 2016) and Proximal Policy Optimization (PPO) (Schulman et al. 2017a), require agents to learn directly and exclusively from the current policy's experiences. Given their on-policy nature, these algorithms necessitate frequent collection of new samples to inform each policy update, albeit typically re-

quiring fewer updates due to the direct learning from every new experience (see Table 2.1). A2C improves learning by using parallel environments, thus reducing variance and improving stability. PPO, on the other hand, modifies the objective function to ensure small, consistent updates to the policy, preventing disruptive changes and ensuring steady improvement. These methods prioritize learning from fresh, immediate experiences, aligning closely with a trial-and-error approach. They are recognized for their lower sample efficiency compared to other methods, and their computational time depends more heavily on episode rollouts than on optimization. For instance, the Brax physics engine (C. Daniel Freeman et al. 2021b) capitalized on PPO's advantages by accelerating and parallelizing interactions with environments through the JAX framework (Bradbury et al. 2018), achieving the feat of teaching a Humanoid to walk in about 20 minutes on a single GPU. This is significantly quicker compared to the tens of hours required by an off-policy algorithm in Mujoco (Todorov et al. 2012), albeit necessitating substantially more environment interactions. We will employ PPO for gathering expert demonstrations in Chapter 3 and for our experimental work in Chapter 4 and 6 . Technically, PPO is adaptable for both continuous and discrete action spaces and facilitates parallel execution. It has gained renewed popularity in the Reinforcement Learning From Human Feedback (RLHF) community, as evidenced by its application in projects such as InstructGPT (Ouyang et al. 2022). Given the extensive use of this algorithm in my thesis (due to its efficiency in terms of wall time), we proceed now with an in-depth exploration of this algorithm [2].

To understand PPO, one has to dive into the foundational concept of policy gradients methods (Richard S. Sutton et al. 2000). In policy optimization, we focus on maximizing an objective function that represents the expected return of a policy parameterized by θ. The objective typically maximized an *empirical average* over a finite batch of sample in an algorithm that alternates between *sampling* and *optimization*. The frequency of alternating between sampling new data and updating the policy parameters, θ, is critical. Frequent refreshes ensure that the policy remains responsive to the latest environmental feedback, which is pivotal for the algorithm's efficacy and convergence. These two steps are:

2. You can find our implementation here.

- Sampling: Trajectories are rolled out using the most recent version of the parameterized policy π_θ. We collect tuples $\{s, a, r\}$ and employ a critic estimator to compute the Generalized Advantage Estimations (GAE) (Schulman et al. 2018) denoted by $\hat{A}$, which provides an estimate of the cumulative future rewards with reduced variance. These tuples are then stored in a dataset $\mathcal{D} = \{s, a, \hat{A}\}$, where the order of the elements is immaterial.

- Optimization: The objective is to refine the policy parameters θ to maximize expected returns. This process is formalized as follows:

$$L(\theta) = \mathbb{E}_{s,a,\hat{A}\sim\mathcal{D}}\left[\log\left(\pi_\theta(a|s)\right) \cdot \hat{A}\right] \tag{2.2}$$

  The expectation $\mathbb{E}_{s,a,\hat{A}\sim\mathcal{D}}$ is taken over the tuples sampled from the dataset $\mathcal{D}$, reflecting an aggregation of the log-probability of actions weighted by the advantage estimates, irrespective of the sequence in which they are encountered.

PPO inherits from a slighlty more sophisticated policy gradient method, Trust Region Policy Optimization (TRPO) (Schulman et al. 2015), which constrains policy updates to stay within a trust region, ensuring the new policy doesn't diverge significantly from the old policy with the use of a constraint on the Kullback-Leibler divergence (KL) (Kullback and Leibler 1951). In practice, this algorithm alternates between sampling from the environment using the latest policy $\pi_{\theta_{\text{old}}}$ and optimizing the current policy parameters θ (inherited from θ_{old}). This loop of sampling and optimizing facilitates stable and efficient learning. PPO simplifies the TRPO framework by adopting a *clipped* objective function[3], which mitigates the need for a constraint and thus simplifies computations. The PPO objective function seeks to optimize the policy by maximizing the following expected return, while keeping policy updates within a defined range to ensure stability:

$$L^{PPO}(\theta) = \mathbb{E}_{s,a,\hat{A}\sim\mathcal{D}}\left[\min\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)} \cdot \hat{A}, \text{clip}\left(\frac{\pi_\theta(a|s)}{\pi_{\theta_{\text{old}}}(a|s)}, 1-\epsilon, 1+\epsilon\right) \cdot \hat{A}\right)\right] \tag{2.3}$$

3. An alternative formulation of PPO incorporates a KL divergence penalty (Schulman et al. 2017a). However, in practice, this approach often presents challenges in hyperparameter tuning and can lead to numerical instability during KL divergence calculations, affecting implementation robustness.

The clipping mechanism is the centerpiece of PPO, ensuring that the updates to the policy are kept within a predetermined range, encapsulated by the hyperparameter ϵ. The function clip serves as a safeguard, adjusting the policy update factor to be no less than $1 - \epsilon$ and no more than $1 + \epsilon$. By taking the minimum between the unclipped and clipped ratios, it ensures that the policy update remains conservative. This confines the optimization process to a trust region around the old policy, thus facilitating stable and incremental policy improvement. PPO's clipped objective thus acts as a regularizer, maintaining the balance between exploration and exploitation, and preserving the agent's capacity to learn effectively without drastic policy deviations.

2.1.2 Off-policy algorithms

In fields where data collection is costly and labor-intensive, maximizing the utility of every piece of data is crucial. It is particularly evident in robotics (Kober et al. 2013; Gu et al. 2017), where the efficient reuse of diverse data is paramount. Off-policy algorithms address this challenge by enabling the use of data from various policies, not just the current one. This allows for the accumulation and reuse of experiences over time, enhancing learning efficiency and effectiveness. By leveraging all available data—regardless of its source—off-policy methods provide a robust framework for learning from limited data, making them particularly suited for robotics applications where re-enacting scenarios for data collection may not be feasible. Compared to on-policy algorithms, off-policy methods exhibit lower variance in their updates and offer a more cost-effective approach to sampling, given their ability to leverage a wider range of data sources (in continuous control environments offered by Brax, they can use a hundred times less samples). However, the computational complexity of updates tends to be higher in off-policy learning due to the additional effort required to reconcile the differences between the behavior policy (data-generating policy) and the target policy (policy being optimized). This trade-off is summed-up in Table 2.1.

To store this diverse range of past experiences, off-policy algorithms like Soft Actor-Critic (SAC) (Haarnoja et al. 2018a) allow the agent to learn from an experience *replay buffer* (Mnih et al. 2013), which contains a diverse range

of past experiences. SAC stands out by integrating the *entropy* (Shannon 1948) of the policy [4] into the reward structure, promoting a balance between exploration and exploitation. it further optimizes this balance by automatically adjusting the trade-off using a *dynamic temperature* paramete. This adaptive approach ensures that SAC can maintain a high level of performance across a wide range of tasks without manual tuning of exploration parameters, making it exceptionally effective in environments with sparse or deceptive rewards (Haarnoja et al. 2018b)

We will use this algorithm [5] in Chapter 7. This approach enables agents to learn from both past and current experiences, providing a more comprehensive learning process than on-policy methods.

2.1.3 Offline RL

Offline RL (Levine et al. 2020), also known as batch RL (S. Lange et al. 2012), represents the far end of the spectrum, where agents learn entirely from a pre-collected set of experiences without any further interaction with the environment. behavioral cloning (BC) (Pomerleau 1988) is a prevalent yet simple algorithm where the policy fits the actions given by a dataset of experts. In environments with continuous actions, it simply leads to reducing the *mean squared error* between expert actions and policy's actions over a dataset. This approach is particularly useful in scenarios where exploration is costly, risky, or impractical (Santara et al. 2017). The obvious downside is the necessity of obtaining this expert dataset, which is not always possible. Another critical drawback is its poor generalization: the agent often fails in unseen situations or rare states (even within the same MDP used by experts) demonstrating a significant challenge in adapting beyond the learned experiences. However, promising results have recently been demonstrated with Decision Transformers, as outlined in L. Chen et al. (2021). These models, conditioned on past timesteps and cumulative rewards, have shown the ability to surpass the performance of experts from the training dataset in the D4RL benchmark (Fu et al. 2020). The underlying question, however, remains whether this surpassing performance is

4. During training, the agent's policy we learn is all in all a *distribution* over the action space. This mechanism allows to increase uncertainty in the decision-making process, thus exploration. Entropy is a way to increase this.

5. You can find our implementation here.

due to true generalization capabilities or if the model is adeptly imitating segments of the best expert trajectories, tactically shifting among experts depending on their performance in specific parts of the trajectory. The use of a transformer architecture may also be not necessary (Siebenborn et al. 2022).

We will use BC in Chapter 3 as a backbone for all our experiments. While being an old and simple method, it remains a strong baseline compared to state-of-the-art methods (Monier et al. 2020).

Transitioning from the exploration of RL algorithms, we now shift our focus to the practical environments and settings where these algorithms will be applied. This discussion is structured to explore the transition from zero-shot generalization to few-shot adaptation. It is crucial to recognize that this progression, while appearing linear from general to specific, is fundamentally about relaxing an initial hypothesis—specifically, transitioning from an environment with 0 "free budget time" to one that allows time for adaptation. This shift in focus is particularly pronounced in the subsequent discussion on continual reinforcement learning, where exploration time is intricately woven into the setting.

2.2 Zero-shot Generalization

This section introduces the core concepts that form the basis of my work on Zero-shot generalization (ZSG) (Chang et al. 2008) elaborated in Part II. ZSG is **the ability for an agent of effective performance in new, *unseen* environments** (*i.e.* without additional training). This is an emerging area in RL (R. Kirk et al. 2023) that aims to deploy RL agents in ambitious real-world settings that are diverse, changing, and unpredictable. The underpinning concept involves producing policies that can *generalize* across varying conditions, a notion inspired by human cognitive flexibility. While the potential applications of ZSG are significant, spanning areas such as autonomous driving (Filos et al. 2020; Han et al. 2021) and sim-to-real in robotics (Zhao et al. 2020; Peng et al. 2017), practical applications remain limited. As noted by R. Kirk et al. (2023), the field is still far from resolving ZSG challenges in these domains. Given the breadth of this topic, we will specifically focus on settings that are particularly relevant to this thesis: one concerning the topic of *morphological generalization* (deeply linked with

Chapter 3), and the other more as a distinct setting or approach, namely *multi-objective reinforcement learning* (further explored in Chapter 4).

2.2.1 Problem Statement

Zero-shot generalization in RL, defined as the capability of an algorithm to generalize to new, unseen environments without further training, necessitates policies robust against variations in environmental dynamics, reward structures, and state distributions. Namely, these variations are given under a specific *context* (*i.e.* a context defines a new MDP). One good way to formalize it is the use of a *Contextual Markov decision processes* (CMDPs) (Kirkpatrick et al. 2017; Ghosh et al. 2021; C. F. Perez et al. 2020):

Definition 2.1: Contextual Markov decision process
A CMDP is a tuple $\langle \mathcal{C}, \mathcal{S}, \mathcal{A}, \mathcal{M}(c) \rangle$, where $\mathcal{C}$ is the context space, $\mathcal{S}$ the state space, $\mathcal{A}$ the action space, and $\mathcal{M}$ maps each context $c \in \mathcal{C}$ to a specific MDP with unique dynamics p^c, reward functions r^c, and initial state distributions $\mathcal{S}_0^c$. pace $\mathcal{A}$, and a mapping of each context to a specific MDP $\mathcal{M}(c)$.

In other words, what is shared across tasks is the state space and action space. The design of $\mathcal{C}$ is critical, as it encapsulates the diverse variations influencing MDP dynamics and rewards.

However, this definition is quite broad and necessitates another level of classification. Inspired by Khetarpal et al. (2020) in the context of continual reinforcement learning (refer to Section 2.4), we categorize these problems along two axes: the object and the scope of generalization. For scope, a key distinction lies between *in-distribution* (test tasks sharing the training set's distribution) and *out-of-distribution* (OOD, "entirely new" tasks) generalization. The former is akin to multitask learning, emphasizing transfer learning within similar tasks, while the latter demands adaptation (in zero-shot!) to novel scenarios and environments.

Regarding the object of generalization, we can draw a parallel with supervised learning, as characterized by J. Liu et al. (2021). Let $P_{tr}(X, Y)$ be the training distribution of random features variable X and label variable Y and $P_{te}(X, Y)$ the test distribution (*i.e.* the distribution on which we aim to

generalize). We can distinguish between *covariate shift*, where the marginal distribution of features changes ($P_{tr}(X) \neq P_{te}(X)$), and concept shift, where the conditional distribution changes ($P_{tr}(Y|X) \neq P_{te}(Y|X)$). In RL, this distinction translates to differentiating scenarios where the transition function changes (e.g., a new map in a video game) from those where the reward function changes (e.g., a new skill to aquire in a video game). An example of the first type is discussed in Section 2.2.2, and the second type in Section 2.2.3. these variations may involve changes in both p and r, making the challenge even bigger.

2.2.2 Covariate shift: example with morphology generalization

A major challenge in generalization is training RL agents that generalize effectively across environments they haven't encountered before, especially when these new environments introduce subtle variations of those they've been trained on (Kaelbling et al. 1996). This challenge is particularly acute in the context of morphological variations: in a 3D navigation problem when switching agent entity from a robot to another, both the action space and transition function change, forcing the agent to navigate with this new dynamics. Hence, the robotics community from RL have focused on morphological generalization (Zhao et al. 2020), employing frameworks like domain randomization (Agrim Gupta et al. 2022), imitation learning (Furuta et al. 2023), meta-learning (Trabucco et al. 2022), and knowledge distillation (Hejna et al. 2020). These approaches aim to equip agents with the versatility to adapt to a range of morphological changes. All these approaches share a common need: they rely on interaction with diverse training environments or learning from data derived from these environments.

However, because research in robotics often deals with a limited variety of training scenarios, there is a significant gap in the availability of diverse and comprehensive data for training, which has historically impeded the development of robustly generalizing RL agents. This scarcity of data has led researchers to rely heavily on meticulously crafted feature engineering and the creation of specialized, often bespoke architectures (Huang et al. 2020a; T. Wang et al. 2018; Agrim Gupta et al. 2022; Hong et al. 2022; Furuta et al. 2023; Kurin et al. 2021). While these methods are effec-

tive in narrowly defined scenarios, they often fall short when encountering new, unseen morphologies, especially out-of-distribution variations that are completely novel. The reliance on handcrafted features and architectures, tailored to specific morphologies, has limited scalability and flexibility, making them less effective in dealing with even in-distribution variations. This approach represents a symptomatic treatment of the broader challenge of morphological generalization. The resulting solutions, while innovative, are inherently brittle, struggling to adapt to the unpredictability and diversity of real-world scenarios.

A possible solution is to use simple yet effective BC (see Section 2.1.3) as a method common to all variants, providing a baseline for fitting policies to data. The focus then shifts to examining how different representations of context are managed within this framework, essentially looking at the role of contextual features on policy performance. This exploration of context-aware policy learning, using BC as a backbone, will be detailed in our contribution in Chapter 3.

2.2.3 Concept shift: Example with Multi-Objective Reinforcement Learning

In some scenarios, agents are tasked with maximizing rewards linked to user preferences (Chankong and Haimes 2008). How then can we quickly satisfy a new user's requirements? One strategy is to learn multiple reference rewards during training, hoping to capture a new user's reward preferences. This approach, known as Multi-Objective Reinforcement Learning (MORL) (Chankong and Haimes 2008), involves managing various proxy rewards, each evaluating performance based on different criteria, necessitating the development of policies that strike an effective balance. Some MORL settings present this challenge in zero-shot generalization: not only an agent must balance and optimize multiple, potentially conflicting objectives at train time, but it also has to infer and provide the best behavior with respect to the test environment (e.g. a new user that uses a recommender system plateform).

Typically, MORL methods employ linear scalarization strategies, as seen in the expensive MORL baseline (Barrett and Narayanan 2008; K. Li et al. 2020). However, these methods struggle with scalability due to com-

putational and memory demands. Contemporary MORL methods modify training procedures, adding complexity and new hyperparameters. From constructing and combining expert models (Won et al. 2020; C. Yang et al. 2020) to single-model training (Castelletti et al. 2013; R. Yang et al. 2019), these methods often do not surpass linearized MORL in Pareto-optimality.

In Chapter 4, we will explore this setting by introducing a novelty with weight interpolation in a *reinforcement learning from human feedback* (RLHF) setting, using *large language models* (LLMs) to address it. Drawing from recent work in LLMs using weight interpolation (H. R. Kirk et al. 2023; Hayes et al. 2022), our approach shifts from traditional single-policy frameworks to multi-policy strategies. This method emphasizes the relative preferences across a set of Pareto-optimal networks (Pareto 1964), providing a more comprehensive solution to the nuanced complexities in real-world scenarios. This also serves as an introduction to subsequently introduce the subspaces of policies in Part II which also utilize weight interpolation.

2.3 Few-shot Adaptation

Few-shot adaptation in RL is a subfield that addresses the challenge of enabling agents to **quickly adapt to new tasks with limited interactions**. This capability is crucial for practical applications where agents must perform in dynamic environments or learn new skills rapidly. The concept draws inspiration from human learning, where we often generalize from few examples. The problem has been studied under different terminologies: *Multi-task Reinforcement Learning* (Wilson et al. 2007; Teh et al. 2017), *Transfer Learning* (Taylor and Stone 2009; Lazaric 2012) and *Meta-Reinforcement Learning* (Finn et al. 2017; Hausman et al. 2018; Humplik et al. 2019). Many different methods have been proposed, but the vast majority considers that the agent is trained over multiple environments such that it can identify variations (or invariant) at train time. For instance, Duan et al. (2016b) assume that the agent can sample multiple episodes over the same environments and methods like (E. Z. Liu et al. 2021) consider that the agent has access to a task identifier at train time. This is the difference between *knowledge induction* (*i.e.* extracting general knowledge from a large quantity of training samples) and *knowledge transfer* (*i.e.* generalizing knowledge with a small number of samples in new tasks), highlighted by Z. Wang et al. (2023). Given our interest lies more with adaptation rather than generalization in this section, we will focus on the former.

2.3.1 Problem Statement

We address the challenging scenario where a policy, trained on a MDP $\mathcal{M}$, is subsequently deployed on another MDP[6], denoted $\bar{\mathcal{M}}$, which shares the same state and action space but differs in dynamics, initial state distribution, or reward function. This setting, emblematic of real-world adaptation challenges, presupposes that $\bar{\mathcal{M}}$ remains unknown during the training phase, prohibiting its use in model selection and complicating hyper-parameter tuning. let a trajectory $\tau = \{(s_1, a_1), ...\}$ denote a sequence of state-action pairs and let us consider that a policy π generates a sequence of trajectories

6. Typically, a single training environment is juxtaposed with multiple test environments to assess adaptability to varying conditions (See Chapter 6).

$\tau_1, \tau_2, ..., \tau_K$ over $\bar{\mathcal{M}}$. Reusing the classical RL objective in Equation 2.1 the performance of such a model, is defined as:

$$Perf(\pi, \bar{\mathcal{M}}, K) = \mathbb{E}_{\pi,\bar{p}} \left[\sum_{t=1}^{H} \bar{r}(s_t, a_t) \mid \tau_1, \tau_2, \ldots, \tau_K \right] \tag{2.4}$$

where $\bar{r}$ denotes the reward function of $\bar{\mathcal{M}}$, and $\mathbb{E}_{\pi,\bar{p}}$ signifies the expectation over trajectories generated by policy π in the new MDP $\bar{\mathcal{M}}$ with transition function $\bar{p}$. The expectation is conditioned on the trajectories experienced during the adaptation phase. This metric effectively quantifies the post-adaptation performance of policy π in $\bar{\mathcal{M}}$, highlighting the essence of swift and efficient environmental adaptation. Empirically, this performance is calculated by averaging the rewards obtained from executing the policy π_θ across a set of post-adaptation trajectories in $\bar{\mathcal{M}}$.

2.3.2 Existing Approaches

Several methods have been proposed to address few-shot adaptation in RL. One of the most popular, Model-Agnostic Meta Learning (MAML)(Finn et al. 2017), optimizes a model's initial parameters so that a few gradient steps on a new task will result in good performance. Extensions and variations of MAML have been developed to overcome limitations of gradient-based optimization. FOMAML(Nichol et al. 2018) simplifies the algorithm. Bayesian MAML (Kim et al. 2018) and Probabilistic MAML (Finn et al. 2019) incorporate a probabilistic approach, integrating prior distributions over the model parameters to better capture uncertainty and enhance the model's adaptability to new tasks. TAML (Jamal and Qi 2019) seeks to train a more task-agnostic initial model, enhancing generalization across different tasks. Indeed, not all adaptations to new tasks yield positive results; when a policy performs poorly, it is referred to as negative adaptation. Deleu and Bengio (2018) argue that this is often due to over-specialization of the policy parameters within the meta-learning framework, which may excel in specific tasks but fail in others, highlighting negative adaptation as a significant challenge yet to be fully addressed in meta-learning. Another drawback is that these methods typically operate under the assumption that multiple training environments are available, which are used to discern commonalities and differences to improve test-time performance (especially if test

tasks are *in-distribution*, *i.e.* coming from the same distribution of tasks as the training ones). This is a significant assumption as it implies the agent has the opportunity to experience variations prior to encountering the test environment. In the setting we described, there is no such variability during training, making it impossible to anticipate changes that will occur in new environments. An interesting strategy would be to learn different ways to solve the unique task the agent has at its disposal at train time. Diversity-based approaches have been adapted to follow this strategy, emphasizing the learning of multiple policies rather than a singular approach. For instance, DIAYN (Eysenbach et al. 2018) introduces a method to learn a discrete set of policies that can be reused and fine-tuned across new environments. This concept is further explored in SMERL (S. Kumar et al. 2020) for few-shot adaptation, where the intrinsic diversity reward is combined with the training task reward. SMERL's reward function integrates the task-specific reward, $r_{task}(s_t, a_t)$, with an entropy maximization term to encourage diversity:

$$r_{SMERL}(s_t, a_t) = r_{task}(s_t, a_t) \; + \; \alpha \cdot \mathbb{1}_{R(\pi) \geq R(\pi^*) - \epsilon} \cdot \log q(z|s) \tag{2.5}$$

where $\mathbb{1}_{R(\pi) \geq R(\pi^*) - \epsilon}$ is a boolean function that activates the entropy term if the policy's average cumulative reward denoted as $R(\pi)$ is within ϵ of the best policy π's average cumulative reward. The entropy term, $-\log q(z|s)$, penalizes predictability, thereby encouraging the policy to explore a broader set of behaviors. Here, q acts as a discriminator and z represents the latent variable denoting style space, which is typically a bounded variable over $[0, 1]^m$ with m being relatively small (1 to 4). While promising, SMERL's effectiveness is contingent on the careful tuning of hyperparameters ϵ and α, balancing the task reward and entropy reward to avoid overfitting or excessive randomness. Moreover, its reliance on a **discrete** latent space may restrict the variety of achievable policies. This limitation is underscored by the constraint in the learning objective $\max \sum_t R(\pi)(s_t) \text{s.t.} \forall z, R(\pi) \geq R(\pi^*) - \epsilon$, aiming for diversity within a specified performance threshold. Moreover, the training process in SMERL necessitates sampling complete episodes at each iteration, which may not be compatible with all reinforcement learning algorithms, particularly those that do not operate on whole episode completions. Based on this observation,Osa et al. (2021) proposed an alternative based on learning a continuous set of policies instead of a discrete one without using any intrinsic reward. In Chapter6, we delve deeper into these

methods, discussing their theoretical underpinnings and practical implementations. We aim to provide a comparative analysis against our proposed approach, highlighting the advantages and limitations of each. Specifically, our focus will be on addressing the challenges posed by the use of a discriminator in models like SMERL. The discriminator, integral for encouraging diverse behaviors through the entropy term, unfortunately, makes the model exceedingly difficult to train and highly sensitive to the choice of hyperparameters. This sensitivity can lead to instability in learning and a narrow margin for error in hyperparameter tuning, directly impacting the model's performance and generalization capabilities.

Diversity-driven approaches are also often rooted in evolutionary computation, and combining them offers a complementary perspective to few-shot adaptation in RL. Quality Diversity (QD) algorithms, such as those discussed in Pugh et al. (2016), emphasize the creation of diverse *policy populations* during training, an approach further explored in Cideron et al. (2020) and Pierrot et al. (2022). QD algorithms are designed to generate a diverse set of high-performing solutions, exploring a wide range of behaviors rather than converging on a single optimal solution. This diversity enables the agent to have multiple fallback options if certain behaviors fail due to environmental changes or damage. A seminal work in this area is the Intelligent Trial and Error (IT&E) algorithm introduced by Cully et al. (2015). IT&E creates a detailed behavior-performance map before deployment, representing the agent's "intuitions" about potential behaviors and their efficacy. This map is used to guide an intelligent trial-and-error process when the agent encounters new situations or damage, allowing for rapid adaptation. For instance, experiments with a hexapod robot demonstrated adaptation to various types of leg damage within approximately one minute. The survey by Chatzilygeroudis et al. (2019) further explores policy search algorithms for learning robot controllers with minimal trials, emphasizing the concept of "micro-data reinforcement learning." This approach is particularly relevant for physical robots where extensive training episodes are impractical. The survey highlights strategies such as leveraging prior knowledge, using surrogate models, and combining priors with data-driven models to achieve rapid learning and adaptation. Intrinsic motivation plays a crucial role in these adaptation processes. Colas et al. (2022) discuss autotelic agents that are intrinsically motivated to set and achieve their own goals without external rewards. This approach, rooted in developmental

robotics and integrated with deep RL, enables agents to autonomously generate and pursue goals in open-ended environments. Intrinsically Motivated Goal Exploration Processes (IMGEPs) allow agents to generate and pursue their own goals, facilitating the acquisition of diverse skills and enhancing adaptability.

The concept of weight interpolation for generalization has been explored in various contexts. Forestier and Oudeyer (2016) introduced Model Babbling (MB) and its active version, Modular Active Curiosity-driven Model Babbling (MACOB), which efficiently explore high-dimensional structured sensorimotor spaces. These modular architectures have shown superior performance in learning to use tools and adapting to new tasks compared to traditional methods. The modular approach allows for efficient exploration of subspaces and can potentially be extended to interpolate between learned policies for novel task adaptation. By exploring and maintaining diverse strategies, QD algorithms and related approaches equip a system with a repertoire of behaviors that can be beneficial for adaptation to new tasks. This contrasts with the single-policy optimization in traditional meta-learning approaches and mitigates the risks of negative adaptation discussed above. In essence, QD approaches cultivate a population of policies, each potentially suited to different environments or tasks, thus inherently supporting adaptation. These methods do not rely on gradient-based optimization, distinguishing them from MAML and its variants, and also from our work on the Subspace. The concept aligns with continual learning objectives in RL, especially in unpredictable scenarios, as shown in Parker-Holder et al. (2020) and S. Wu et al. (2023). We will discuss about this more ambitious framework in the next section.

2.4 Continual Reinforcement Learning

Extending the exploration of few-shot adaptation, we now turn to Continual Reinforcement Learning (CRL), a domain that pushes the envelope in the development of autonomous agents. These agents excel in **leveraging previously acquired knowledge to efficiently master and remember new tasks, thus enabling them to tackle and adapt to a wide sequence of tasks seamlessly**. This section sets the stage for Chapter 7, helping to grasp the challenges of the area and where our method CSP stands.

Continual Learning has become a hot topic in research, with many comprehensive reviews covering a lot of ground in the area, like the works of Mundt et al. 2020; M. D. Lange et al. 2021; Parisi et al. 2019. Most of these studies focus on task incremental learning, where each task comes with its own set of data, consisting of input-output pairs (X, Y) that come from a particular distribution $\mathcal{D}$. The main aim here is to minimize risk across all tasks you've learned so far, even when you can't look back at the data from previous tasks once you've moved on. On the other hand, continual RL is all about making decisions one after another across a sequence of tasks, where each task is its own little world with consistent rules, known as a stationary MDP.

We are particularly interested in this setting derived from continual learning due to its straightforwardness and relevance to real-world scenarios. We use again the example of a player starting a game, and progressing level by level. The concept of a task sequence becomes quite meaningful here, with each level representing a task, *i.e.* a specific MDP. These tasks share common elements (same possible actions, same 2D observations for the player) but also different ones (the map changes, affecting the transition function in RL, and possibly the goal, affecting the reward function). Sometimes, the player will encounter levels very similar to those they've seen before, allowing them to leverage their accumulated knowledge to perform as well or better than before. This is known as *backward transfer*. Other times, they'll face completely new levels and will need to quickly adapt using their array of knowledge. This is known as *forward transfer*. These two notions are keys in the Continual Learning community, and we will dive into them in Section 2.4.2. But ultimately, the problem of CRL is a specific aspect of the generalization problem where tasks are presented in a stream, making this framework very relevant to real-world situations.

2.4.1 Problem Statement

The litterature of CRL is characterized by its numerous definitions and scenarios (Wołczyk et al. 2021; Nekoei et al. 2021; Powers et al. 2022). Pursuing the "purest" form of CRL can lead to extreme cases, like the Jelly Bean World's non-stationary environments and "never-ending learning" (*i.e.* no notion of episode) (Platanios et al. 2020). Indeed, one can give a very

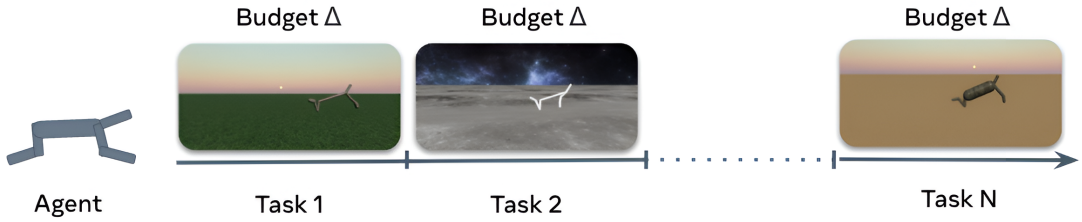


Figure 2.2 – Illustration of a specific online sequential CRL problem where tasks have the same budget Δ and reward function is the same (travel the longest possible distance), while the transition function changes across tasks. Tasks are derived from HalfCheetah environment, see Section 7.3

.

broad definition of a CRL problem. Given a state space S, action space $\mathcal{A}$, an observation space $\mathcal{O}$, a reward function $r : \mathcal{S} \times \mathcal{A} \to \mathbb{R}$, a transition function $p : \mathcal{S} \times \mathcal{A} \to S$, the most general form of a continual reinforcement learning problem can be described as $\mathcal{M}_{CRL} := \langle \mathcal{S}(t), \mathcal{A}(t), r(t), p(t), x(t), \mathcal{O}(t) \rangle$, where each component of the formulation is understood to be time-variant, with t indicating the temporal aspect of the problem.

In other words, the sequence of MDPs changes over time, which is known as *non-stationarity*. While these theoretical concepts are intriguing, their current real-world applicability is often limited. In the past decade, various classifications have been proposed, each based on different premises. Khetarpal et al. 2020 have provided a structured way to understand these varied problem settings, helping to clarify the field. They describe a setting using two main aspects: the scope and the driver of non-stationarity. The scope of non-stationarity is concerned with what changes within the MDP. This could mean changes to the entire MDP, but as mentioned earlier, such problems are not practically solvable. The most common scope discussed in the literature involves changes to the transition function p and the reward function r. As for the driver, many studies concentrate on online multi-task learning where tasks come sequentially. For the sake of simplicity, we will also consider that state spaces and observation spaces are equivalent[7].

The concept of assigning a budget per task, coupled with limited and potentially separate memory for each task, is common in CRL (Khetarpal et al. 2020). This setting aligns more closely with real-world scenarios and

7. POMDPs (Astrom et al. 1965) are out of scope of this thesis. The vast majority of the environments used in our work present fully observable MDPs.

enhances privacy, especially in cases where tasks represent different users. Indeed, in this case, one do not want to share the same *replay buffer* continuously and avoid and sensitive information sharing. In addition, the storing of such replay buffer is often memory-intensive (Wołczyk et al. 2021). Figure 2.2 illustrates a typical scenario derived from this setting that is defined in Section 7.3. Following these guidelines, we define a common CRL problem, which we will explore in Chapter 7:

Definition 2.2: Online sequential CRL problem
Consider a state space $\mathcal{S}$, an action space $\mathcal{A}$. A Online Sequential CRL problem is defined by a sequence of n tasks $T_1, ..., T_n$ s.t. each task T_i is defined by a MDP $\mathcal{M}_{T_i} = \langle \mathcal{S}, \mathcal{A}, p_i, r_{T_i} \rangle$ associated with a training budget of interactions $b_{T_i} \in \mathbb{N}$. When the system switches to task T_{i+1} it no longer has access to transitions from $\mathcal{M}_{T_i}$.

2.4.2 Metrics and Trade-offs

In the setting described above, the ultimate goal of an autonomous agent is to maximize overall **performance** across all tasks. To facilitate this, establishing a metric for comparing cumulative rewards across different tasks is crucial. This entails introducing the notion of a **reference agent** for a task T, denoted by π^T; this agent serves as a benchmark, having been trained exclusively on the single task T using a similar method to that of the autonomous agent being evaluated. Essentially, the reference agent acts as a stand-in for the evaluated agent if it had been trained solely on task T. Let us define an autonomous agent π and a online sequential CRL problem $T_1, ..., T_n$.

Definition 2.3: Performance
Assuming the cumulative sum of rewards is almost surely strictly positive, and given an horizon $H \in \mathbb{N}^*$ that aims to fairly compare trajectories between each others, the performance of an on task T_i is defined as the ratio of the average cumulative reward achieved by the agent to the average cumulative reward obtained by the reference agent, which has been trained exclusively on that single task. It is given by:

$$\text{Perf}_{T_i}(\pi) := \frac{\mathbb{E}_{\pi, p_{T_i}}\left[\sum_{t=0}^{H} r_{T_i}(s_t, a_t)\right]}{\mathbb{E}_{\pi^{T_i}, p_{T_i}}\left[\sum_{t=0}^{H} r_{T_i}(s_t, a_t)\right]}$$

In CRL, as in Continual Learning (Lopez-Paz and Ranzato 2022), the traditional trade-off is between *forward transfer*, which is the ability of an agent to apply knowledge from previous tasks to new ones, and *backward transfer*, also known as Forgetting, which measures the preservation of performance on old tasks. Since its parameters change over each training, we denote the policy π that has been trained on the i first tasks by π^i, *i.e.* π^i has been trained sequentially on $T_1, ..., T_i$. Although there are multiple ways to define these two metrics (Lopez-Paz and Ranzato 2022; Wołczyk et al. 2021; Khetarpal et al. 2020), we aim to use the above definition of Performance to make it simple:

Definition 2.4: Forward and backward transfer
Forward transfer measures how much a CRL system is able to transfer knowledge from task to task. At task i, it compares the performance of the of the system trained on all previous tasks $T_1, ..., T_i$ to the same model trained solely on task T_i. This measure is defined as $\text{FT}_i(\pi) := \text{Perf}_{T_i}\left(\pi^i\right) - 1$, and thus the global forward transfer is $\text{FT} := \frac{1}{n}\sum_{i=1}^{n} \text{FT}_i$. A $\text{FT} \geq 0$ means that the agent is using knowledge from former tasks to train better on unseen tasks.

Backward transfer evaluates how much a system has forgotten about task i after training on the full sequence of tasks. It thus compares the performance of policy π^i with the performance of policy π^n and is defined as $\text{BT}_i : (\pi) = \text{Perf}_{T_i}(\pi^i) - \text{Perf}_{T_i}(\pi^n)$. Similarly to the average trans-

fer, we report the average forgetting across all tasks $\mathrm{BT} := \frac{1}{n}\sum_{i=1}^{n}\mathrm{BT}_i$. Growing methods (see Section 2.4.3) typically have a backward transfer near 0.

While forward and backward transfer are pivotal metrics in Continual Learning literature for interpreting, comparing, and benchmarking various methods, ultimately, they serve as proxies for the more comprehensive Performance metric previously described. Consider, for example, the outcomes of two simple baselines against these metrics. First, repeatedly finetuning a single policy might yield a modest forward transfer but typically results in catastrophic forgetting. A straightforward remedy might be to "finetune and store" each task's policy π^i, thus completely preventing any forgetting (making backward transfer effectively zero). However, with an increasing task stream, this approach becomes untenable as the memory required to store the policies escalates dramatically (increasing linearly with the number of tasks). Díaz-Rodríguez et al. 2018 made the same observation, and like them, we propose a third metric, vital for assessing the feasibility of CRL models, termed *model size*:

Definition 2.5: Model Size
Assuming scales are appropriately calibrated (*i.e.* the methods are sharing the same architecture backbone or at least with similar size), model size represents the total parameter count of a method after training on all tasks, divided by the parameter count of a single policy trained on all tasks.

For instance, if we train and store a separate policy for each task, the model size equates to n. Conversely, if we continually finetune a single policy, the model size remains 1. Yet, nuances exist within this metric. A method might not increase in size but could accrue costly parameters across tasks, such as gradients and coefficients (Kirkpatrick et al. 2017). In such cases, the model size might, for instance, total 3 (accounting for parameters, gradients, and coefficients).

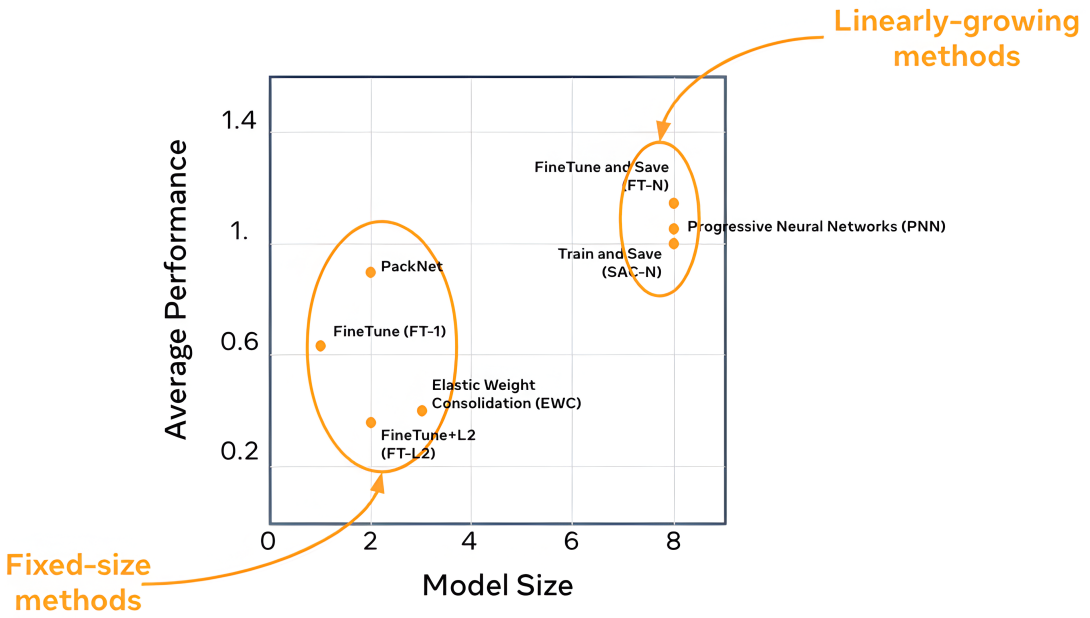


Figure 2.3 – The dichotomy of fixed-size versus linearly-growing methods in CRL, illustrating a stark balance between model complexity and task performance, suggesting an absence of a middle ground. It is a central theme of Chapter 7.

2.4.3 Existing Methods

In Chapter 7, we will confront these challenges through a streamlined CRL framework, balancing model size and performance instead of the typical trade-offs lead by forward and backward transfers. This chapter will elucidate how our methods address the problem. For now we present the existing methods, highlighting the dichotomy between fixed-size and linearly-growing methods. Figure 2.3 illustrates it.

CRL methods strive to circumvent *catastrophic forgetting* (McCloskey and Cohen 1989) and to foster transfer across numerous tasks without sacrificing scalability. A diverse array of solutions illustrates the field's multifaceted challenges (Tessler et al. 2017; Parisi et al. 2019; Khetarpal et al. 2020; Powers et al. 2021; Wołczyk et al. 2021; Berseth et al. 2022; Nagabandi et al. 2018; A. Xie et al. 2020; M. Xu et al. 2020; H. Ren et al. 2022; Kessler et al. 2022).

One major concern is forgetting, addressed by strategies like storing previous model parameters (Cheung et al. 2019; Wortsman et al. 2020). Although these methods are effective, their scalability is hindered by linearly increasing computational and memory demands (*i.e.* Model Size is growing too

fast). In particular, PNN (Rusu et al. 2016b) creates a new network at the beginning of each task, as well as lateral networks that will take as inputs - for each hidden layer - the output of the networks trained on former tasks. In this method, the number of parameters, training and inference times are growing quadratically with respect to the number of tasks. One have to reduce the number of hidden layers with respect to the final number of task (which assumes to have access to the final number of tasks) to make it comparable with other methods. Methods like PACKNET (Mallya and Lazebnik 2018) perform admirably. PACKNET is the leading method in Continual World benchmark (Wołczyk et al. 2021)). It aims to learn on a new task and prune the networks in the end, and retrain the weights that have the highest amplitude. This has two drawbacks : one has to allocate in advance a certain number of weight per task. Without additional knowledge on the sequence, authors recommend to evenly split the number of weights per task, meaning that one has to know the final number of tasks. In addition, the training procedure after having pruned the network does not require any interaction with the environment but is computationally expensive. Its variant, EfficientPackNet (Schwarz et al. 2021), though promising, faces limitations in broader evaluations.

Similarly, methods enhancing *transfer* capability, such as Elastic Weight Consolidation (EWC) (Kirkpatrick et al. 2017), aim to minimize forgetting but may constrain adaptability. EWC is a regularization based method that aims to protect parameters that are important for the previous tasks. After each task, it uses the Fisher information matrix to approximate the importance of each weight. A baseline is proposed in the same paper, FT-L2, and re-used by Wołczyk et al. (2021). It can be seen as a simplified version of EWC where the regularization coefficients for each parameters are equal to 1. Knowledge distillation attempts to aid transfer but is challenged by the necessity of capturing a wide array of behaviors within a single network (Hinton et al. 2006; Rusu et al. 2016a; Z. Li and Hoiem 2018; Schwarz et al. 2018).

Research exploiting task commonalities, meta-learning, and generative models seek to bolster CRL. Yet, these approaches frequently fall short in practicality (Ju Xu and Z. Zhu 2018; Pasunuru and Bansal 2019; K. Lu et al. 2021; Mankowitz et al. 2018; Abel et al. 2017; Sodhani et al. 2021; Javed and White 2019). S. Lee et al. (2020) propose a network expansion strategy, but it remains confined to supervised learning without clear task delineations.

2.5 Conclusion

In this Chapter, we have reviewed the various algorithms and settings that we will use in this thesis. Zero-shot generalization emerges as the first significant leap beyond basic RL principles, challenging agents to perform adeptly in entirely new environments. This challenge sets the stage for the nuanced exploration of few-shot adaptation, where the hypothesis relaxes slightly, allowing the agent a minimal set of interactions to adjust to new tasks. The narrative then evolves to encompass CRL, a setting that captures the essence of real-world applications by considering the "free time" or the sequential task exposure an agent undergoes. This progression underscores a natural escalation from mastering unseen environments to adapting quickly with sparse data, culminating in the agent's ability to learn continually in a sequence of changing tasks. We will explore these topics in the rest of the thesis:

- Part II is dedicated to dissecting the intricacies of generalization within the domain of RL. Here, we delve into the theoretical underpinnings, practical challenges, and the evolving landscape shaped by advancements such as transformers. This part aims to offer a comprehensive view of generalization, its potential, and its boundaries in the current era of AI research.

- Part III pivots to the novel contributions of this thesis, specifically focusing on the application of subspaces in RL under the varied settings of few-shot adaptation and CRL. This part is poised to introduce innovative approaches and methodologies that address the nuanced challenges identified in earlier discussions, setting a new direction for future research in RL.

PART II

Zero-shot Generalization

CHAPTER 3

A PATHOLOGICAL CASE

This chapter provides an introductory exploration into a challenging case: morphological zero-shot generalization in continuous control. It is based on an ongoing work [1] called *Advancing zero-shot Morphological generalization*. As discussed in Section 2.2.2, current attempts in this area are still nascent, especially when compared to the advancements seen in Computer Vision or Natural Language Processing. Drawing inspiration from the settings of transformers training in other supervised learning areas, we elaborate settings and demonstrate results that surpass those currently documented in the literature, while also highlighting that there are still significant efforts required to achieve *true generalization*. Section 3.1 serves as an introduction, briefly revisiting the problem of morphological generalization and outlining the research questions we aim to address. Section 3.2 formalizes the problem using the definition of CMDP we saw when discussing about ZSG (see Section 2.2). Section 3.3 describes the model we developed and the state-of-the-art models we benchmark against. Section 3.4 details our experimental protocol, while Section 3.5 presents an analysis of our findings. Finally, Section 3.6 concludes this chapter and this first part of the thesis.

3.1 Context & Motivations

As discussed in Section 2.2, a major challenge in RL is training agents that generalize effectively across environments they haven't encountered before, especially when these new environments introduce complex or subtle variations of those they've been trained on (Kaelbling et al. 1996). In addressing the challenges of morphological generalization, this chapter posits that a

1. Please note that the content presented in this chapter is part of an ongoing project. The current state of the research is a snapshot of a dynamic and evolving process, with the intention to provide an initial framework and preliminary findings that will be refined and expanded upon as the project progresses.

fundamental issue lies in the scarcity of diverse and comprehensive data. The lack of a dataset encompassing a wide range of morphologies has led to an over-reliance on feature engineering and specialized architectures. These methods, while offering immediate solutions, fall short in addressing the fundamental challenge of generalizing across diverse morphological variations. Consequently, we delve into established practices and propose novel methodologies to tackle these issues. Our analysis leverages offline and online RL techniques, including PPO (see Section 2.1.1) for the creation of expert trajectories and BC (see Section 2.1.3) as a training method to ensure fair comparisons and ablations.

To thoroughly investigate these challenges, we raise several research questions that guide our study:

RQ1: In what specific data regime can we observe Morphological Generalization?

The confluence of dataset size and its inherent diversity plays a pivotal role in the success of morphological generalization. Our investigation encompasses three dimensions: two quantitative — the number of robots and the number of trajectories — and a qualitative one, the diversity of the training set. This diversity can be visualized on a spectrum, with one end representing robots sampled from a narrowly defined parameter space, and the other end representing robots from a broader, more varied parameter space. To overcome this issue and conduct an in-depth analysis of the problem, we introduce the "Massive Ant-based Dataset" (MAD), a comprehensive dataset of expert trajectories from 7000 variations of the Ant morphology from the Brax physics engine (C. Daniel Freeman et al. 2021b). MAD is conceived to establish a new standard for the research community, exploring the potential of curated data in markedly enhancing morphological generalization performance. Our experimental framework utilizing MAD encompasses various data sizes in two dimensions: the number of morphologies and the number of expert trajectories per morphology. These experiments are crafted to test the generalization capabilities of agents, both in distribution (originating from the same distribution as the training data) and out-of-distribution, posing greater challenges due to the inclusion of more "extreme" morphologies.

RQ2: What architectural improvements are needed to better process information for Morphological Generalization?

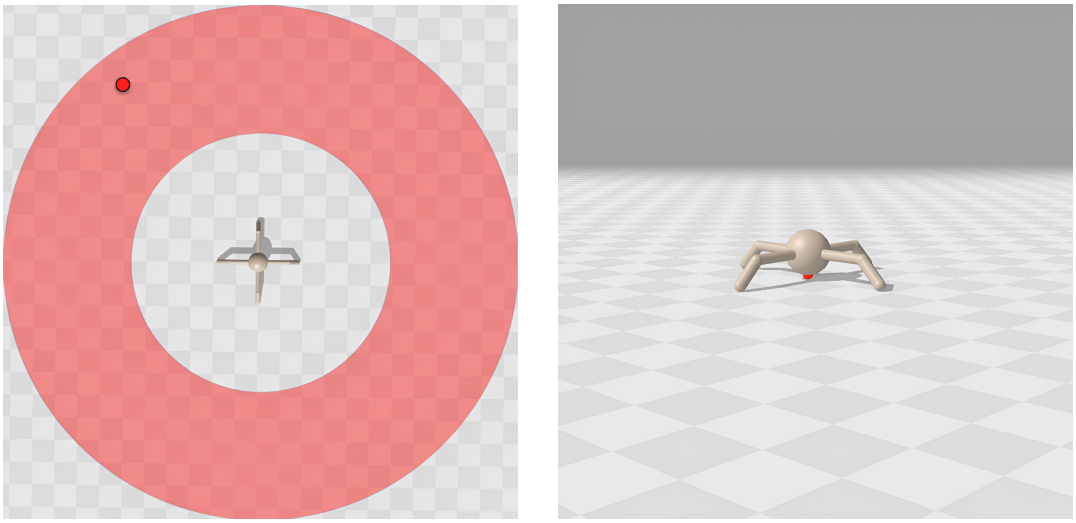
We examine the role of model architecture, particularly contrasting node-based models with Gato-Inspired architectures, in enhancing the generalization capabilities in RL. While node-based models have been the preferred choice, they have been contested (Kurin et al. 2021). Drawing inspiration from Gato (Reed et al. 2022) which demonstrates a versatile architecture to combine multiple task capabilities within a single model, we introduced an alternative model architecture. Our experimental results elucidate that this Gato-Inspired architecture not only competes with but surpasses the performance of traditional node-based models (Agrim Gupta et al. 2022; Furuta et al. 2023; T. Wang et al. 2018). This revelation underscores the significance of re-evaluating long-standing beliefs and prompts a reconsideration in the design of architectures tailored for morphological generalization.

RQ3: How essential is Feature Engineering for advancing Morphological Generalization?

Contrary to the common belief in the necessity of complex feature engineering for morphological generalization, our findings suggest that simpler, normalized features can deliver comparable or even superior performance. This insight potentially simplifies the design process and decrease the computational load. We also designed an end-to-end method based on the computational definition of a morphology with the use of LLMs. The experiments conducted with this model reveals that the more specific to the task the contextual function is, the more overfitting we see.

3.2 Problem Statement

A major challenge in current RL research is achieving zero-shot generalization—utilizing expert data to generalize to new environments without further interaction (R. Kirk et al. 2023; Fu et al. 2020). Our approach revises the conventional RL generalization model within the CMDP framework (see Section 2.2) to focus on learning from a static dataset D of expert trajectories. Aligning with these modern RL strategies, this dataset-centric methodology underscores the importance of leveraging pre-existing expert knowledge to facilitate learning in a controlled, interaction-free context. It



(a) Top view of the Brax ant environment.

(b) Side view of the ant reaching the goal.

Figure 3.1 – Views of the Ant environment that we will use as a backbone in this chapter. Left: The goal region is depicted as a red disc, demonstrating the random distribution of the goal's x,y coordinates at the start of each episode, with the goal's position provided in the agent's observation space including kinetics. Right: An illustration of the ant model successfully reaching the goal area, as seen from a lateral perspective.

provides a robust framework for zero-shot generalization by utilizing offline RL, which eliminates the need for direct environmental interaction during the training phase. This approach significantly reduces the variability typically seen with classical RL algorithms (refer to Table 2.1 from Chapter 2), allowing for a focused analysis on the distinct impacts of *architectural design* (RQ2) and *contextual features* (RQ3) on model effectiveness. The dataset D is comprised of expert trajectories (see later in Section 3.4.1) that have learned on various MDPs characterized by a context c. We have broken down the problem we are tackling into three stages (Figure 3.3).

Since we aim to study morphological generalization, it is useful to consider that these MDPs only vary by their transition function. In practice, we choose, like Furuta et al. (2023), a goal-reaching task with a random goal appearing within a 2 to 10m disc (see Figure 3.1). For morphology, we will choose one in particular (also see later in Section 3.4.1) and make variations on it: the context can therefore be essentially assimilated to the coefficients that change these parameters, or even to the configuration file of the mor-

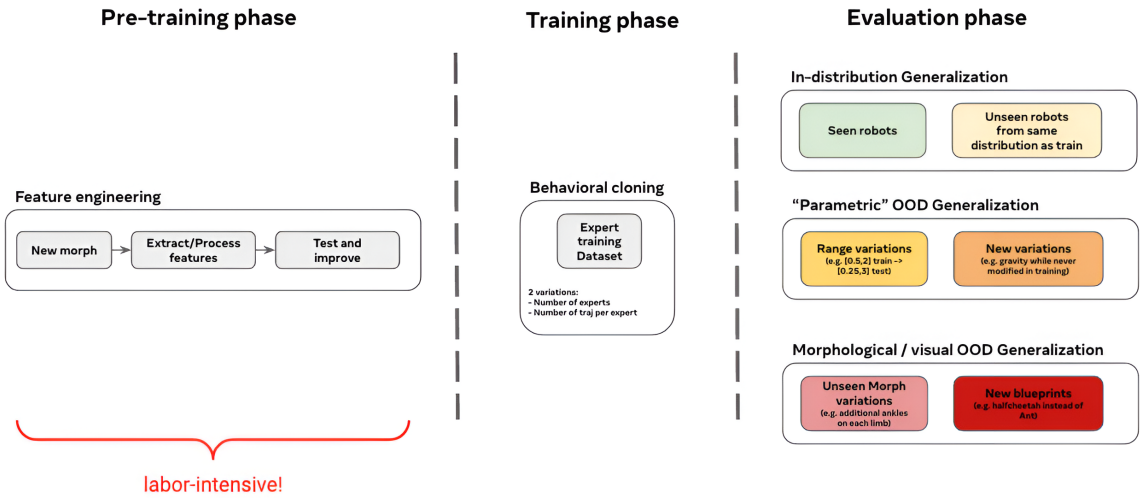


Figure 3.2 – The typical framework used by current methods (Furuta et al. 2023; Kurin et al. 2021) in the morphological generalization. On each phase, it raises several questions we aim to study in this Chapter. On the left: the use of labor intensive feature-engineering as a mandatory pre-training phase when a new morphology appears (RQ3). In the middle: a training phase using BC as a backbone, which demands to use the right architecture and context feature (RQ2, RQ3). On the right, the different levels of generalization we aim to evaluate that are not always detailed nor discussed in the litterature and are deeply linked with the training data we feed to the model (RQ1).

phology given to a physics engine. This is what we want to highlight in Figure 3.3 on the left: feature engineering may not be necessary because the data is already available. We will therefore test models with and without a "pre-training phase" (which is a phase of feature engineering in this case).

The training phase is a simple BC algorithm on a given dataset (see Figure 3.3 in the middle). But the size and diversity of the training dataset can also vary. If we are only going to consider expert data to build our dataset, unlike other works (Fu et al. 2020), our goal here is to look at how variations in the diversity and size of the dataset can influence the results during evaluation.

Finally, the evaluation phase also includes subtleties that we wanted to account for in Figure 3.3 on the right. Indeed, we can seek different types of generalization. One way to characterize it is by the level of difficulty: in green, generalization on seen robots (i.e., we are almost talking about task generalization, which in our case proposes random goals); in light yellow, we add a difficulty with unseen robots but from the same distribution as

the robots in the dataset. In dark yellow, we move to out-of-distribution generalization: unseen robots from outer parametric morphological ranges. We have not tested the rest (whose climax is the generalization to all-new robots) because the results were very weak: this is one of the first limitations to the question RQ1. We will discuss the evaluation datasets in more detail in Section 3.4.1.

3.3 Models

Our research is focused on a two-fold evaluation: examining the influence of architectural designs (RQ2) and the incorporation of several contextual features (RQ3) within reinforcement learning models. We aim to explore a spectrum of models, starting with a baseline (Multi-Layer Perceptron) to serve as a control, progressing through state-of-the-art methods (Node Transformer), and including our own Gato-like approach. The contextual features, representing diverse environmental conditions, are integral to our tests. They are not merely inputs but critical elements that could dictate the success or adaptability of a model. This study is designed to scrutinize both the individual and combined impacts of architecture and contextual understanding on the performance of reinforcement learning systems.

3.3.1 Architectures

The chosen architectures for our investigation are carefully designed to be similar in terms of number of parameters (around 100 millions). This deliberate design choice ensures a level playing field for comparing the different models, where size does not confound the results. The architectures represent a range of computational approaches, from the simplicity of an MLP to the complex encoding mechanisms of a transformer. This setup allows us to focus on how architectural differences affect the models' ability to generalize and process contextual information efficiently in reinforcement learning tasks.

3.3.1.1 Multi-Layer Perceptron (MLP)

The Multi-Layer Perceptron (MLP) employed in our context is a dense, feed-forward neural network consisting of an input layer that aggregates observations and contextual information, followed by multiple hidden layers and an output layer. The observations and context are concatenated, with padding applied as needed to manage discrepancies in their sizes.

The MLP configuration is as follows: it comprises 6 hidden layers with a size of 2048 units each. Between these layers, ReLU activation functions are implemented to introduce non-linearity into the model, enabling it to capture complex relationships in the data. The final layer employs a tanh activation function, ensuring that the action outputs are bounded within a normalized range suitable for downstream tasks that require such constraints. This MLP structure is designed to effectively map the high-dimensional input space to the action space required for the task at hand.

3.3.1.2 Node Transformer

Node Transformers, as depicted in Figure 3.3, are designed for environments where observations are naturally partitioned into entities or "nodes", and each node is associated with a scalar action output. This architectural design is inspired by the encode-process-decode framework prevalent in graph neural networks and has been successfully applied in transformer-based methodologies like AMORPHEUS (Kurin et al. 2021) for multi-task reinforcement learning in continuous control.

The intuition behind the Node Transformer is that the state information of a system, when represented as a fully connected graph by the transformer, can facilitate the learning of message-passing schemas that are dynamically adjusted for each state, rather than relying on predefined graph structures. This flexibility allows Node Transformers to handle a wider range of morphologies, including those not encountered during training, without the physical connectivity constraints seen in models like Shared Modular Policies (Huang et al. 2020b).

Node Transformers are modular, consisting of a linear encoder for each node observation, a central transformer for processing the encoded information, and a MLP as the decoder for action outputs. The model we implemented is characterized by a 6-layer transformer encoder with 512-

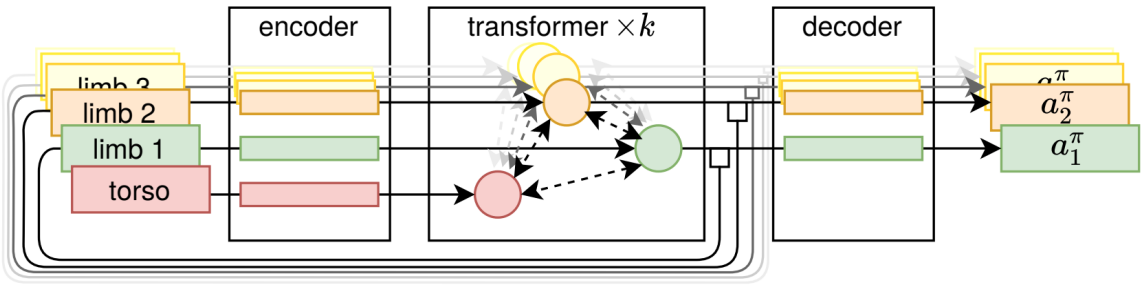


Figure 3.3 – Architecture of a Node Transformer, taken from AMORPHEUS (Kurin et al. 2021). This diagram illustrates the policy network, with individual nodes processed independently through the encoder and decoder, while message passing is facilitated by the transformer block. The critic network (that we do not use in these experiments since they are not necessary with BC) follows an identical architecture, with the decoder outputs representing value function estimates.

dimensional embeddings, 8 attention heads, and a 4x expansion factor for the feedforward network. A dropout of 0.1 is incorporated to prevent overfitting. Notably, the encoder and decoder are designed to process each node independently, enabling the model to adapt to varying morphologies without reconfiguration.

In our implementation, observations for each node are first encoded into a 32-dimensional vector. This encoding includes information such as limb type, relative position, velocities, and rotations. Linear layers are applied to these node observations to facilitate the transformer's ability to learn from continuous control signals. Additionally, residual connections are introduced from the input features to the decoder output, ensuring that the nodes retain their unique features throughout the transformation process.

The architecture's ability to generalize across different environments and morphologies without constraints on the physical connectivity makes it particularly powerful for RL tasks involving robotic control or embodied agents. The Node Transformer's utility is further demonstrated in its capacity to produce scalar actions for each node, allowing for fine-grained control over the actions of complex agents.

3.3.1.3 Gato-like Transformer (Gl-Tr) (ours)

Our Gato-like transformer (GL-TR) extends the architecture of the original Gato model (Reed et al. 2022), incorporating adjustments suited for deep reinforcement learning (RL). While Gato utilizes a decoder-only transformer with 1.2 billion parameters, including 24 layers and an embedding size of 2048, our GL-TR model employs a more compact configuration (100 millions parameters) suitable for RL tasks. Specifically, GL-TR features a 6-layer transformer encoder with an embedding dimension of 512, 8 attention heads, and a post-attention feedforward hidden size set to four times the embedding dimension. This configuration supports efficient processing of contextual information and observations.

Key components of GL-TR include:

- Utilizing a transformer encoder core to process tokenized inputs, where inputs are encoded using a μ-law algorithm with 1024 bins to effectively capture the nuances of context and observations (like in Gato). The μ-law encoding compresses the input space, facilitating the model's ability to generalize across diverse input distributions.

- Implementing a non-auto-regressive approach for action prediction (unlike Gato). This is achieved by inserting placeholders for actions in the input sequence, which are subsequently filled by the model's predictions. This method contrasts with Gato's auto-regressive prediction strategy, offering advantages in terms of computational efficiency and latency in action generation.

- Embedding context information at the layer level prior to concatenation. This approach allows for a more nuanced integration of context into the model's decision-making process, compared to direct pass-through or simplistic encoding methods.

These technical details underscore the GL-TR model's tailored design for RL, balancing computational efficiency with the depth of tokenization to optimize performance in dynamic environments. The architecture leverages the strengths of transformers while introducing modifications that align with the unique requirements of reinforcement learning tasks.

3.3.2 Contextual Features

As seen in Section 2.2, the primary objective in a CMDP is to find a policy π that maximize average cumulative sum of rewards across a context set, which necessitates the extraction and utilization of relevant information from c. In practice, this leads to the development of a mapping function $f : \mathcal{C} \to \mathcal{X}$, where $\mathcal{X}$ is a feature space. The function f translates the raw context c into actionable features $x = f(c)$, which are then used by the policy π for decision-making in each context-specific MDP. This function f, translating context to features, is pivotal in our study. Traditional approaches implement f through feature engineering, but we propose investigating a novel end-to-end approach using a transformer encoder. This exploration aims to enhance the understanding of how different representations of the context space can influence policy optimization in CMDPs, moving beyond conventional methods towards more versatile and effective solutions. In this section, we explore various methodologies for modeling the mapping function f, which is crucial for translating raw context into actionable features for policy decision-making in CMDPs.

3.3.2.1 Morphology-task graph V2 (`mtg-v2`)

Designed first by Kurin et al. (2021) and reshaped by Furuta et al. (2023) it leverages transformers over traditional Grah Neural Networks (GNNs), optimizing message-passing in morphological domain knowledge and bypassing challenges posed by multi-hop propagation in sparse graphs. Its modular architecture is designed for compatibility across diverse environments, free from initial physical connectivity constraints. Rather than one-hot encoding, node observations utilize a linear layer, encapsulating limb type, relative positions, velocities, and normalized angle values. Importantly, residual connections are integrated to ensure that node-specific features are retained throughout the decoding phase for precise action computation.

3.3.2.2 Automated Feature Extraction (`afe`)

We directly compare two configuration files: one of the given morphology and another of a predefined base morphology in the environment. A

script extracts the ratio between corresponding parameters. For instance, when contrasting a torso radius of 0.75 against the base's 0.5, it straightforwardly derives a feature value of 1.5. This streamlined process ensures a consistent feature vector length, regardless of the morphology in question, eliminating the need for intricate feature enginering or adjustments for new morphologies.

3.3.2.3 Configuration embedding (`cfg-emb`)

Our method stems from a straightforward yet critical insight: each morphology is computationally described by a configuration file, e.g.a JSON file (C Daniel Freeman et al. 2021a), an XML file (Todorov et al. 2012) that helps the physics engine to design the dynamical system of a task. Thus, these configuration files contain all the necessary information about a given morphology and its dynamics. We use a pretrained Longformer transformer encoder (Beltagy et al. 2020) to directly embed these configuration files. This approach bypasses the need for traditional feature engineering, allowing our model to interpret the raw data without preprocessing biases. The encoder's efficiency in handling the lengthy and complex configuration files is key to our method. Our primary goal was to determine whether this direct encoding could faithfully represent the original morphology data, which is essential for effective generalization to different morphs. To further explore this, we introduced variations of the model, including an auxiliary task. This task is designed to enrich the learning process by providing supplementary context and learning signals, potentially enhancing the model's capacity for generalizing to unseen morphological variations.

3.4 Experimental Protocol

The objective is to understand the effect of various contextual features on performance. Our methodology comprises of two distinct stages. First stage consists in a Offline RL algorithm : BC (see Section 2.1.3). The goal is to minimize the Mean Squared Error (MSE) with the expert data, which serves as a proxy for the success rate. It is treated as a consistent "black box" to ensure a fair comparison across all experiments. In the second stage, the trained models are subjected to zero-shot evaluations on multiple test sets.

Performance across these test sets is primarily evaluated using the success rate metric, which assigns a binary reward (0 or 1) based on whether the goal is achieved. While MSE measures the fidelity of our models to expert behavior, the ultimate metric of success is robust performance across diverse test sets, evaluating both in-distribution and out-of-distribution scenarios. This two-fold goal ensures both high-quality mimicry of expert actions and generalization to varied test environments.

3.4.1 Building an adequate dataset

The lack of large-scale datasets hampers progress in morphological generalization. We tackle this by introducing a large-scale, diverse dataset named "Massive Ant-based Dataset" (MAD) with 7,000 variations of the Ant morphology (drawn from a distribution shown in Table 3.4.1), aiming to create a new benchmark for the community. Our approach is visually summarized in Figure 3.4, which presents the morphological parameters of both training and test ants within MAD. Our work also delves into how a well-curated, diverse dataset significantly improves performance over a sheer size, guiding the curation of future datasets for morphological generalization.

Our dataset creation procedure takes as input a CMDP $\mathcal{C}$, a number of MDPs k, a batch of trajectories of size m, and a maximum wallclock time. k policies are trained using PPO (see Section 2.1.1 on these k MDP. While training, the deterministic form of each policy π is being asynchronously evaluated over m trajectories with different initial states. The metric for evaluation is simply the average sum of rewards , such that if m is chosen carefully, it should approximate $\mathbb{E}_{S_0^c, p^c}\left[\sum_{t=0}^{H} R^c(s_t, a_t) \middle| \pi\right]$ and thus cover the entire initial state distribution. when the wallclock time is reached, the training procedure outputs the m-batch that has the highest average cumulative sum of rewards. These m trajectories are called *expert trajectories*. After $\left\{\left(c_i, \tau_i^*\right)\right\}_{i=1...n=k\times m}$, i.e. a collection of n pairs of contexts $c \in \mathcal{C}$ and *expert trajectories* $\tau^* := \left\{s_0^*, a_0^*, r_0^*, ..., s_H^*, a_H^*, r_H^*\right\}$.

Our offline RL scenario is centered around the use of a unique, custom dataset we refer to as the "Massive Ant-based Dataset" (MAD). In our exploration of the impact of data variability on model performance, we've utilized 2 additional datasets derived from MAD: a more compact version with 700 experts; and an even more distilled dataset with 70 experts. Each

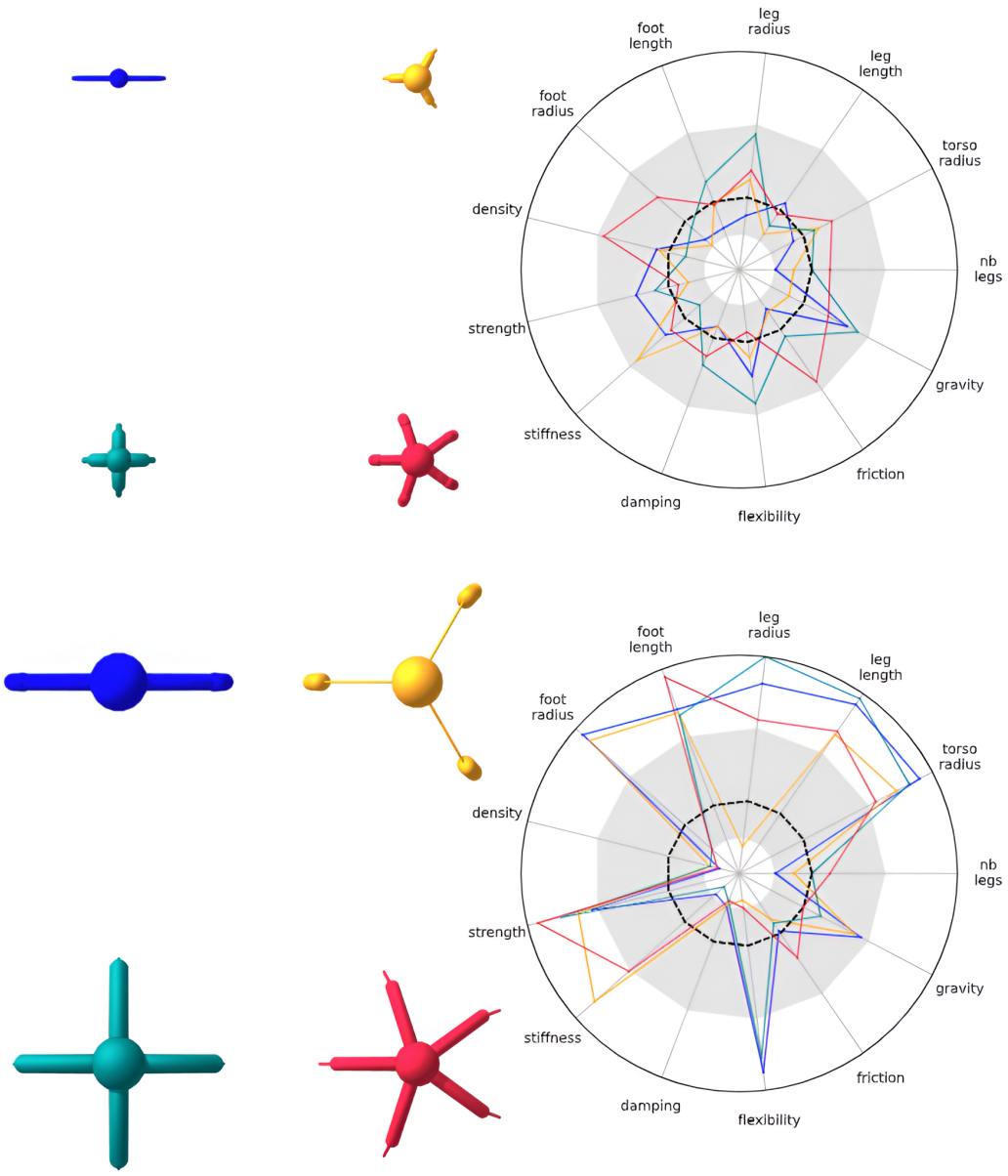


Figure 3.4 – Training and test sets of ant morphologies from MAD. The top row displays four representative morphologies (3D top view) from the training set, while the bottom row presents four from the test set. The radar plot depicts each ant's morphological parameters by the corresponding color line, using a coefficient scaled with respect to the dotted black line (equal to 1): it represents the baseline ant, corresponding to the classical ant morphology in the Brax physics environment. The grey shade depicts the distribution of the training set morphological parameters, from 0.5 to 2. The outermost circle with radius 3 delimits the boundary of the parameter space explored.

Parameter	Range	
	In Distribution	Out Distribution
n (legs)	$[2, 5]$	$[2, 5]$
torso_radius	$[0.5, 2]$	$[0.25, 0.5] \cup [2, 3]$
leg_length	$[0.5, 2]$	$[0.25, 0.5] \cup [2, 3]$
leg_radius	$[0.5, 2]$	$[0.25, 0.5] \cup [2, 3]$
foot_length	$[0.5, 2]$	$[0.25, 0.5] \cup [2, 3]$
foot_radius	$[0.5, 2]$	$[0.25, 0.5] \cup [2, 3]$
density	$[0.5, 2]$	$[0.25, 0.5] \cup [2, 3]$
strength	$[0.5, 2]$	$[0.25, 0.5] \cup [2, 3]$
stiffness	$[0.5, 2]$	$[0.25, 0.5] \cup [2, 3]$
damping	$[0.5, 2]$	$[0.25, 0.5] \cup [2, 3]$
flexibility	$[0.5, 2]$	$[0.25, 0.5] \cup [2, 3]$

Table 3.1 – Distribution of robot parameters. Parameters, with the exception of *n legs*, are expressed as coefficients of the base ant provided by Brax (C. Daniel Freeman et al. 2021b). These ranges have been crafted to strike a balance between robots that are feasible and those that present a challenge. The MAD dataset, which encompasses 7,000 robots, is sourced from the in-distribution, while the out-distribution is set aside for testing purposes.

of these datasets offers a unique perspective on the interplay between the number of experts and the number of trajectories per expert. This dual-scale approach, considering both the number of experts and trajectories per expert, is pivotal in understanding different facets of generalization in Reinforcement Learning. By varying the scale of data, we aim to dissect the nuances of morphological generalization, examining how models generalize across diverse morphologies with varying amounts of training data. This methodology provides insights into the balance between breadth (number of experts) and depth (trajectories per expert) in training data, and its implications for achieving robust morphological generalization in RL.

3.4.2 Model selection & evaluation

To ensure a fair comparison between different methods, we isolate the training phases from the rest. This approach guarantees consistency in model architecture and hyperparameters across all methods. We created a sweep over each architecture and kept the same hyperparameters accross

all models. We conducted a comprehensive sweep for each architecture, maintaining identical hyperparameters for all models. Ultimately, the consistency in training loss across models indicates that the training was effectively conducted, thereby assuring that the outcomes observed during evaluation are not influenced by disparate training treatments.

Post training, we evaluate these models on several subsets derived from MAD. This includes subsets composed of robots seen during training, unseen robots drawn from the same distribution as the training set (Table 3.4.1 left column), as well as robots from divergent distributions (Table 3.4.1 right column), encompassing changes in morphology, dynamics, and capabilities. Moreover, we have designed specific test sets to determine if the models exhibit increased sensitivity to particular changes. This extensive testing and ablation study enables us to understand the different forms of generalization that the models can achieve.

We will utilize the goal-reaching success rate as a key metric to assess the results. All of the evaluation dataset are made of 700 robots and averaged over 512 rollouts.

3.5 Results

3.5.1 RQ1: More morphologies, less trajectories

Our study investigates the model's ability to generalize by varying both the number of unique robotic morphologies (70, 700, 7000) and the trajectories per morphology (1, 10, 100), represented on the heatmap's y and xaxes, respectively. For architectural consistency, a Gatolike model devoid of additional contextual inputs is analyzed across three categories of robots: seen robots (within the training data's range), in-distribution unseen robots, and outofdistribution unseen robots, each evaluated over 512 episodes. The success rates are then averaged across three independent runs of the model.

The heatmap in Figure 3.5 reveals distinct patterns in the data regimes:

- Regimes with a low number of robots and trajectories (e.g., 70 robots with 1 trajectory each) show inadequate generalization capabilities.

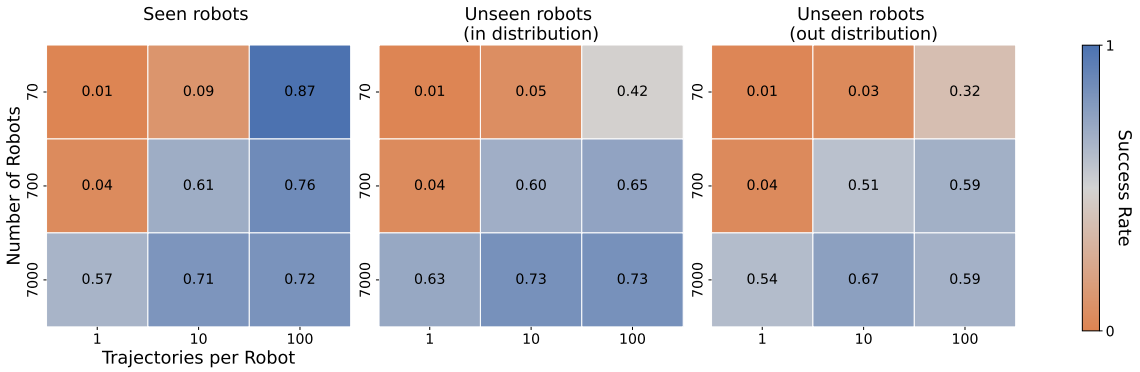


Figure 3.5 – Heatmap of success rates for a Gatolike model under varying dataset configurations. The x-axis details the number of trajectories for each robotic morphology, while the y-axis shows the count of unique morphologies in the training dataset. Panels from left to right indicate model performance on seen robots, in-distribution unseen robots, and out-of-distribution unseen robots. Color intensity reflects the success rate, with a full spectrum from low (blue) to high (orange). These patterns elucidate the relationship between dataset composition and the model's generalization ability, highlighting the superior performance achieved through increased morphological diversity.

- The region corresponding to high trajectory counts per robot (particularly 70 robots with 100 trajectories each) indicates potential overfitting, evidenced by high success rates on seen robots but diminished performance on unseen ones.
- Comparatively, data regimes with a larger set of robots (7000 with 10 trajectories each) suggest an ideal balance, outperforming regimes with a higher number of trajectories per fewer robots (700 with 100 trajectories each) in terms of generalization to unseen morphologies.

3.5.2 RQ2: Advocacy for GatoInspired Architectures

In our comprehensive evaluation across various test scenarios, the GL-TR architecture consistently demonstrated superior performance, outstripping both MLP and NODE models (see Figure 3.6). When benchmarked against intricate datasets tailored for morphological generalization, GL-TR achieved an average performance metric that is notably higher. Specifically, in tests involving complex morphological variations, GL-TR outperformed

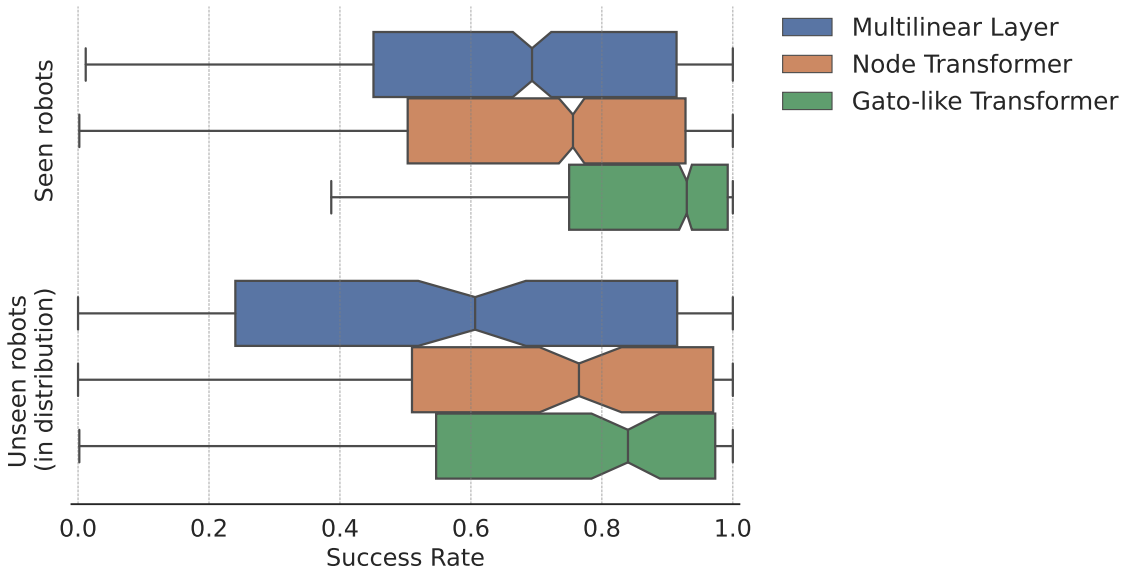


Figure 3.6 – Distribution of the success rate of diverse architectures after having trained on MAD dataset. Evaluation involves averaging over 512 rollouts per robot, then averaging these results across seeds.

the MLP by a margin of 31% and the NODE model by 22%. This performance differential is even more pronounced in scenarios with dynamic environmental changes, where GL-TR's adaptive capabilities are most evident. Such results not only emphasize the robustness and efficiency of the GatoInspired architecture but also suggest a promising direction for future research in the realm of morphological generalization.

3.5.3 RQ3: hand-crafted methods lead to overfitting

Our objective is to assess the role of context in model generalization. To achieve this, we conducted tests on both MLP and GL-TR architecture over diverse test variations (morphology, environmental dynamics, capabilities). The test set comprises 100 distinct morphologies drawn from the same distribution as the `MAD-100` training set. As depicted in Figure 3.7, both MLP and GL-TR architectures with normalized features surpass the performance of `mtg-v2` features. Intriguingly, the model without any features displays comparable performance to `mtg-v2`, prompting us to reevaluate the relevance of such features For unseen robots within distribution, `afe` again dominates with a success rate of 79.53%. Both None and `mtg-v2` feature sets

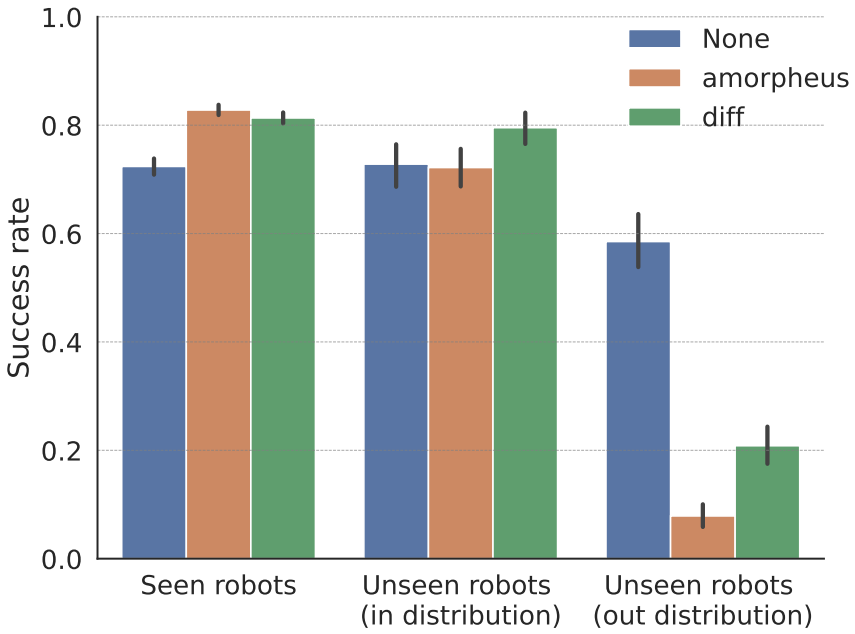


Figure 3.7 – Success rates of the deterministic policies produced by GL-TR architectures trained on the `MAD` dataset. Evaluation involves averaging over 512 rollouts per robot, then averaging these results across robots. The displayed error bars represent the standard deviation over 3 seeds per method.

display similar performance, scoring 72.81% and 72.18%, respectively. When it comes to unseen robots outside of the distribution, all models exhibit a significant drop in success rate. However, the `afe` feature set still leads with a rate of 24.68%. The None feature set follows at 11.01%, while `mtg-v2` lags at 9.87%.The above results indicate that the `afe` feature set consistently outperforms the other models across all test conditions. Interestingly, the model with no designated features (None) competes closely with `mtg-v2`, prompting us to reconsider the efficacy of such features in the context of our problem. To ensure our conclusions aren't skewed by selective feature preference, we further evaluated on ant robots, modifying only specific features.

In an effort to enhance the performance of our morphological generalization model, we explore the use of contextual embeddings derived from raw configuration files. The Longformer (Beltagy et al. 2020) architecture, known for its ability to handle long sequences, is employed to embed these configurations without any additional training, thus creating a more

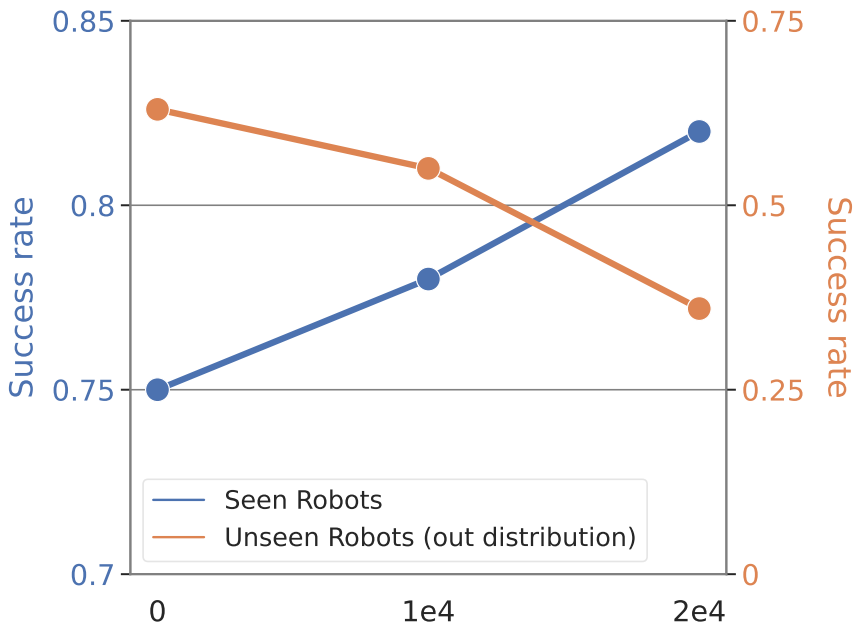


Figure 3.8 – Success rate on training robots and unseen robots with respect to the amount of auxiliary training received by the Longformer's architecture before training the transformer on expert data.

generalized representation of the robot's features. The preliminary task designed for the Longformer is to classify certain key attributes—mass, radius, length—from the raw context. The classification is relative to a typical robot's attributes, simplifying the task to a binary classification rather than a regression problem. The Longformer is trained to discern if a particular attribute is 'large' or 'small' in comparison to a normalized standard. Three models are compared: No auxiliary training, auxiliary training for $1e4$ steps, auxiliary training for $2e4$ steps. During the training of the policy transformer on expert data, the Longformer's embedding is frozen to isolate the effect of the context encoding. Each training step represented exposure to 128 examples, culminating in a near-zero loss at $2e4$ steps, indicating a successful auxiliary task learning.

The graph presented in Figure 3.8 illustrates the success rates for both training on seen robots and testing on unseen robots. The success rates improved with the amount of auxiliary training received by the Longformer's architecture, suggesting that the embedding carried meaningful information during the training phase.

As shown in Figure 3.8, the success rate increased for seen robots but decreased for unseen robots as the amount of auxiliary training grew. This discrepancy points to a case of overfitting, where the model's performance improves on the training data at the expense of its ability to generalize to new, out-of-distribution data. In comparison, models trained without context achieved a success rate of 0.72 on seen robots and 0.59 on unseen robots. On the other hand, the highly context-dependent `mtg-v2` model reached 0.83 on seen robots but plummeted to 0.08 on unseen robots, reinforcing the evidence of overfitting when excessive context is provided.

The experiment conducted reveals a cautionary tale in the pursuit of morphological generalization in continuous control tasks. While contextual embeddings can enhance training performance, their improper application may lead to overfitting, thereby degrading the model's generalizability. This finding underscores the need for a balanced approach to context integration, one that aids learning without compromising the model's out-of-distribution performance.

3.6 Conclusion

Our findings highlight that an emergence of generalization in morphological adaptation is possible under certain circumstances. We observed that larger and more diverse datasets significantly enhance the model's generalization capabilities, especially with out-of-distribution challenges. For instance, using the Gato-Inspired architecture, we achieved improved performance in complex morphological variations, and our experiments indicated that simpler, normalized features often outperform complex, handcrafted ones. Yet, these results also testify that true OOD generalization is not fully realized in our current models (see results for unseen robots from out-of-distribution).

These advancements point to the need for a shift in approach: moving from zero-shot generalization to adaptation, specifically few-shot adaptation. This shift involves not only changing the generalization setting but also rethinking methodologies. In the upcoming part, we delve into the Subspaces of Policies approach, exploring its application in the context of adaptation. This method promises a more nuanced and effective way to handle the unpredictability inherent in real-world scenarios. Before that,

we now continue our journey through ZSG with another setting utilizing multiple reward functions as objectives.

CHAPTER 4

WEIGHT AVERAGING FOR MULTI-OBJECTIVE REINFORCEMENT LEARNING

This chapter can be seen as another generalization setting we aim to tackle, but also as an initial exploration into the weight interpolation strategies that we will use in Part III. We draw upon the NeurIPS paper *Rewarded soups: towards Pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards* [1] (Ramé et al. 2023), led by Alexandre Ramé and to which I contributed by setting up the PPO algorithm and conducting experiments, notably the locomotion task part. I also designed the `website` of the project. This paper is notable for its collaborative nature (5 PhD students), covering a broad spectrum of AI domains including image captioning, image generation, visual segmentation, visual grounding questions, text generation, and continuous control.

Foundation models, typically pre-trained on vast unsupervised datasets, set the standard for neural network weight learning, leveraging self-supervision followed by fine-tuning on labeled data (Bommasani et al. 2021; Oquab et al. 2014; Vapnik 1999). However, the reliance on expensive label collection and the challenges in aligning the trained network with intended applications reveal limitations (Ngo et al. 2022). Reinforcement learning, particularly from human feedback (RLHF), offers a way forward by aligning the network through diverse rewards, despite the potential imperfections in proxy rewards that may hinder optimal training outcomes. The diversity of real-world tasks and human opinions further complicates this alignment. To address these issues, our work proposes a multi-policy strategy, termed rewarded soup, that embraces the heterogeneity of rewards. This strategy involves specializing multiple networks independently for each proxy reward and then interpolating their weights linearly, a method validated by our empirical findings that weights remain linearly connected when fine-tuned on diverse rewards from a shared pre-trained initialization. This

1. You can find the appendix in the arxiv paper `here`. Code is available `here`.

novel approach demonstrates effectiveness across a broad spectrum of tasks, including text-to-text (e.g., summarization, Q&A), text-image (e.g., image captioning, text-to-image generation), and control tasks, aiming to enhance the alignment of deep models with the real world's diversity (Pinto et al. 2023).

We will set the context in Section 4.1, then introduce a simple yet elegant weight averaging-based solution in Section 4.2. The effectiveness of this method will be demonstrated through diverse experiments across various domains in Section 4.3. We will conclude in Section 4.4.

4.1 Context & Motivations

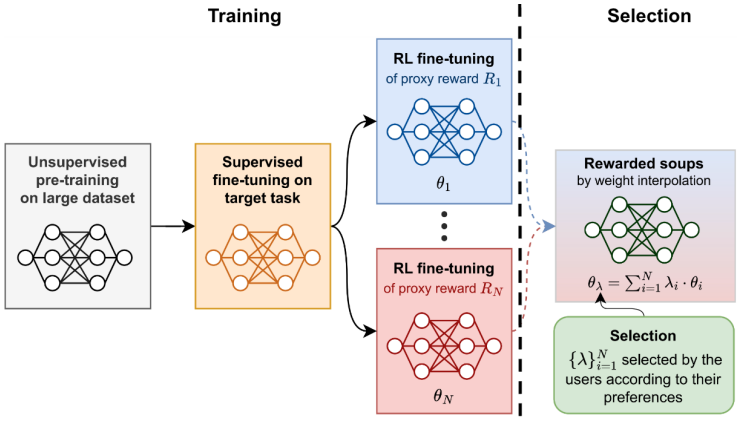
A prominent example is RLHF, which appears as the current go-to strategy to refine LLMs into powerful conversational agents such as ChatGPT (OpenAI 2023). After pre-training on next token prediction (Radford et al. 2018) using Web data, the LLMs are fine-tuned to follow instructions (Taori et al. 2023) before reward maximization. This RL strategy enhances alignment by evaluating the entire generated sentence instead of each token independently, handling the diversity of correct answers and allowing for negative feedback (Goldberg 2023). Similar strategies have been useful in computer vision (CV) (Rennie et al. 2017), for instance to integrate human aesthetics into image generation (Zhang et al. 2023). The recognition of diversity in proxy rewards within RLHF poses a significant challenge, necessitating a shift towards a multi-policy approach to better align models with the wide range of human preferences. This approach, aiming for a nuanced alignment, sets the stage for the introduction of *rewarded soup*, a novel strategy designed to effectively harness this diversity. The forthcoming sections will delve into this approach, illustrating how it can address the complexities of model alignment with varied preferences and laying the foundation for a detailed exploration of its implementation and impact.

4.1.1 Diversity of proxy rewards

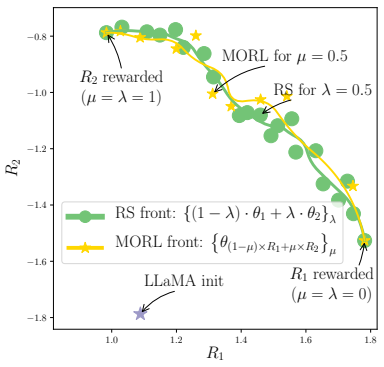
RL is usually seen as more challenging than supervised training (Dulac-Arnold et al. 2021), notably because the real reward - ideally reflecting the users' preferences - is often not specified at training time. Proxy rewards are therefore developed to guide the learning, either as hand-engineered metrics (Vedantam et al. 2015) or more recently in RLHF as models trained to reflect human preferences (Jiazheng Xu et al. 2023). Nonetheless, designing reliable proxy rewards for evaluation is difficult. This *reward misspecification* (Pan et al. 2022) between the proxy reward and the users' actual rewards can lead to unforeseen consequences (Michaud et al. 2020). Moreover, the diversity of objectives in real-world applications complicates the challenge. In particular, human opinions can vary significantly on subjects such as aesthetics (Nadal and Chatterjee 2019), politics or fairness (Lopez-Paz et al. 2022). Humans have also different expectations from machines: for example, while Ganguli et al. 2022 stressed aligning LLMs towards harmless feedback, Bai et al. 2022b requested helpful non-evasive responses, and others' (Irvine et al. 2023) interests are to make LLMs engaging and enjoyable. Even hand-engineered metrics can be in tension: generating shorter descriptions with higher precision can increase the BLEU (Papineni et al. 2002) score but decrease the ROUGE (C.-Y. Lin and Hovy 2003) score due to reduced recall.

4.1.2 Towards multi-policy strategies

Considering these challenges, a single model cannot be aligned with everyone's preferences (Ouyang et al. 2022). Existing works align towards a consensus-based user (Ovadya 2023), relying on the "wisdom of the crowd" (Bai et al. 2022a), inherently prioritizing certain principles Kovač et al. 2023, resulting in unfair representations of marginalized groups (H. R. Kirk et al. 2023). The trade-offs Pan et al. 2023 are decided a priori before training, shifting the responsibility to the engineers, reducing transparency and explainability (Hayes et al. 2022), and actually aligning towards the "researchers designing the study" (Santurkar et al. 2023). These limitations, discussed in Section 2.2.3, highlight the inability of single-policy alignment strategies to handle human diversity. Yet, "human-aligned artificial intelli-



(a) Illustration of our proposed rewarded soup (RS).



(b) LLaMA RLHF for summarization.

Figure 4.1 – Figure 4.1a details the different steps in rewarded soup. After unsupervised pre-training and supervised fine-tuning, we launch N independent RL fine-tunings on the proxy rewards $\{R_i\}_{i=1}^N$. Then we combine the trained networks by interpolation in the weight space. The final weights are adapted at test time by selecting the coefficient λ. Figure 4.1b shows our results (see Section 4.3) with LLaMA-7b (Touvron et al. 2023a) instruct fine-tuned on Alpaca (Taori et al. 2023), when RL fine-tuning for news summarization (Stiennon et al. 2020) with $N = 2$ reward models assessing diverse preferences of summaries. With only two trainings (R_1 and R_2 rewarded on Figure 4.1b), the λ-interpolation ($0 \leq \lambda \leq 1$) reveals the green front of Pareto-optimal solutions, *i.e.* that cannot be improved for one reward without sacrificing the other. RS matches the costly yellow front of multi-objective (MORL) (Barrett and Narayanan 2008; K. Li et al. 2020) requiring multiple trainings on different linear weightings over the rewards $(1-\mu) \times R_1 + \mu \times R_2$ with $0 \leq \mu \leq 1$.

gence is a multi-objective problem" (Vamplew et al. 2018). Thus, we draw inspiration from the MORL literature (also see Section 2.2.3). Z. Wu et al. (2023) and Hayes et al. (2022) argue that tackling diverse rewards requires shifting from single-policy to multi-policy approaches. As optimality depends on the relative preferences across those rewards, the goal is not to learn a single network but rather a *set of Pareto-optimal networks* (Pareto 1964).

In this Chapter, we propose *rewarded soup* (RS), an efficient and flexible multi-policy strategy to fine-tune any foundation model. As shown in Figure 4.1a, we first use RL to learn one network for each proxy reward; then, we combine these expert networks according to user preferences. This a posteriori selection allows for better-informed trade-offs, improved trans-

parency and increased fairness (Hayes et al. 2022). The method to combine those networks is our main contribution: we do this through simple *linear interpolation in the weight space*, just like proposed in Chapter 5 despite the non-linearities in the network. This is in line with recent findings on linear mode connectivity (see Chapter 5). Actually, the name *rewarded soups* follows the terminology of *model soups* (Wortsman et al. 2022), as we combine various *ingredients* each rewarded differently. Unlike previous works, which focused on supervised learning, we explore Linear Mode Connectivity [2] (LMC) in RL, in a challenging setup where each training run uses a different reward. Perhaps surprisingly, we show that we can trade off the capabilities of multiple weights in a single final model, thus without any computational overhead. This enables the creation of custom weights for any preference over the diverse rewards. We summarize our contributions as follows:

- We advocate a multi-policy paradigm to align deep generative models with human preferences and reduce reward misspecification.
- We then propose a new multi-policy strategy, rewarded soup, possible when fine-tuning foundation models with diverse rewards. By weight interpolation, it defines a continuous set of (close to) Pareto-optimal solutions, approximating more costly multi-policy strategies.

In Section 4.3, we consistently validate the effectiveness of RS across a variety of tasks and rewards: RLHF fine-tuning of LLaMA, multimodal tasks such as image captioning or text-to-image generation with diffusion models, as well as locomotion tasks.

4.2 Rewarded Soup

4.2.1 RL fine-tuning with diverse rewards

We consider a deep neural network f of a fixed non-linear architecture (*e.g.* with batch normalization (Ioffe and Szegedy 2015), ReLU layers (Agarap 2018) or self-attention (Vaswani et al. 2017)). It defines a policy

2. In Chapter 5, we will further explore this subfield.

by mapping inputs x to $f(x, \theta)$ when parametrized by θ. For a reward $\hat{R}$ (evaluating the correctness of the prediction according to some preferences) and a test distribution T of deployment, our goal is to maximize $\int_{x \in T} \hat{R}\left(f\left(x, \theta\right)\right)$. For example, with f a LLM, x would be textual prompts, $\hat{R}$ would evaluate if the generated text is harmless (Askell et al. 2021), and T would be the distribution of users' prompts. Learning the weights θ is now commonly a three-step process: unsupervised pre-training, supervised fine-tuning, and reward optimization. Yet $\hat{R}$ is usually not specified before test time, meaning we can only optimize a proxy reward R during training. This *reward misspecification* between R and $\hat{R}$ may hinder the alignment of the network with $\hat{R}$. Moreover, the *diversity of human preferences* complicates the design of R. Rather than optimizing one single proxy reward, our paper's first key idea is to consider a family of N diverse proxy rewards $\{R_i\}_{i=1}^N$. Each of these rewards evaluates the prediction according to different (potentially conflicting) criteria. The goal then becomes obtaining a coverage set of policies that trade-off between these rewards. To this end, we first introduce the costly MORL baseline. Its inefficiency motivates our rewarded soups, which leverages our second key idea: weight interpolation.

MORL baseline. The standard MORL scalarization strategy (Barrett and Narayanan 2008; K. Li et al. 2020) (recently used in Z. Wu et al. (2023) to align LLMs) linearizes the problem by interpolating the proxy rewards using M different weightings. Specifically, during the *training phase*, M trainings are launched, with the j-th optimizing the reward $\sum_{i=1}^N \mu_i^j R_i$, where $\forall j \in \{1, ..., M\}, \{\mu_i^j\}_{i=1}^N \in \Delta_N$ the N-simplex such that $\sum_{i=1}^N \mu_i^j = 1$ and $0 \leq \mu_i^j \leq 1$. Then, during the *selection phase*, the user's reward $\hat{R}$ becomes known and the j-th policy that maximizes $\hat{R}$ on some validation dataset is selected. We typically expect to select j such that $\sum_{i=1}^N \mu_i^j R_i \approx \hat{R}$ linearly approximates the user's reward. Finally, this j-th weight is used during the *inference phase* on test samples. Yet, a critical issue is that "minor [preference] variations may result in significant changes in the solution" (Vamplew et al. 2008). Thus, a high level of granularity in the mesh of Δ_N is necessary. This requires explicitly maintaining a large set of $M \gg N$ networks, practically one for each possible preference. Ultimately, this MORL strategy is unscalable in deep learning due to the **computational, memory, and engineering costs** involved (see further discussion Appendix A.2).

Rewarded soup (RS). The idea is to learn expert anchors and then interpolate them linearly at inference time to combine their abilities. Specifically,

we propose RS, illustrated in Figure 4.1a and whose recipe is described below. RS alleviates MORL's scaling issue as it requires only $M = N$ trainings while being flexible and transparent.

1. During the *training phase*, we optimize a set of N expert anchors $\{\theta_i\}_{i=1}^N$, each corresponding to one of the N proxy rewards $\{R_i\}_{i=1}^N$, and all from a shared pre-trained initialization.
2. For the *selection phase*, we linearly interpolate those weights to define a continuous set of rewarded soups policies: $\{\sum_{i=1}^N \lambda_i \cdot \theta_i\}_{\{\lambda_i\}_{i=1}^N \in \Delta_N}$. Practically, we uniformly sample M interpolating coefficients $\{\{\lambda_i^j\}_{i=1}^N\}_{j=1}^M$ from the N-simplex Δ_N and select the j-th that maximizes the user's reward $\hat{R}$ on validation samples, *i.e.* $\arg\max_{j=1}^M \hat{R}\left(\sum_{i=1}^N \lambda_i^j \theta_i\right)$.
3. For the *inference phase*, we predict using the network f parameterized by $\sum_{i=1}^N \lambda_i^j \theta_i$.

While MORL interpolates the rewards during training, RS interpolates the weights during inference. This is a considerable advantage as the appropriate weighting λ, which depends on the desired trade-off, can be selected *a posteriori*; the selection is achieved without additional training, only via inference on some samples. We will conduct experiment in the next section to prove that.

4.2.2 Exploring the Properties of Rewarded Soups Set of Solutions

We center our attention on two pivotal research question that would fundamentally inform our understanding of this method:

RQ1 (Linear Mode Connectivity). For any combination of weights fine-tuned on different rewards, does the performance for each individual reward *at least* match the combined performance of these rewards when averaged? This question suggests that the performance of a blended policy should not be less than the average performance of its constituent policies.

RQ2(Pareto Optimality). Does our solution set forms a Pareto coverage set? This would implies that within this set, any improvement in performance on one objective necessitates a compromise on another. A Pareto coverage set is an efficient compromise among various objectives, underlining the

effectiveness of different combinations of weights in achieving a balance across multiple rewards.

Based on this understanding of Pareto Optimality, we would like to test a critical implication for scenarios where rewards are linear combinations of various proxy rewards, as captured in the following research question:

RQ3 (Reduced Reward Misspecification in the Linear Case). If our system's Pareto Optimality holds true, and for any linear combination of rewards, does there exist an optimal set of weights in our solution set? In other words, for a linear reward composed of different proxy rewards, we can always find a combination of weights from our set that is optimal for this reward.

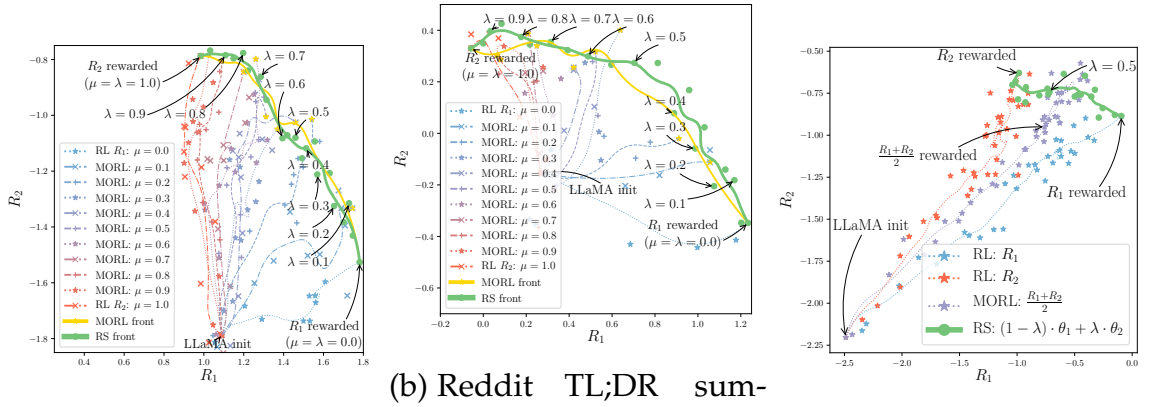
These research questions are the backbone of our approach and provide insight into the capabilities and potential results of our method in varied reinforcement learning scenarios. In Section 4.3, we plan to empirically test them, further validating the effectiveness of the Rewarded Soups.

4.3 Experiments

In this section we implement RS across a variety of standard learning tasks: text-to-text generation, image captioning, image generation, visual grounding, visual question answering, and locomotion. We use either model or statistical rewards. We follow a systematic procedure. First, we independently optimize diverse rewards on training samples. For all tasks, we employ the default architecture, hyperparameters and RL algorithm; the only variation being the reward used across runs. Second, we evaluate the rewards on the test samples: the results are visually represented in series of plots. Third, we validate **RQ1** by examining whether RS's rewards exceed the interpolated rewards. Lastly, as the true Pareto front is unknown in real-world applications, we present empirical support for **RQ2** by comparing the front defined by RS (sliding λ between 0 and 1) to the MORL's solutions optimizing the μ-weighted rewards for $0 \leq \mu \leq 1$ (sometimes only $\mu = 0.5$ for computational reasons). Implementations are released on `github`, and this `website` provides additional qualitative results.

4.3.1 Text-to-text: LLaMA with diverse RLHFs

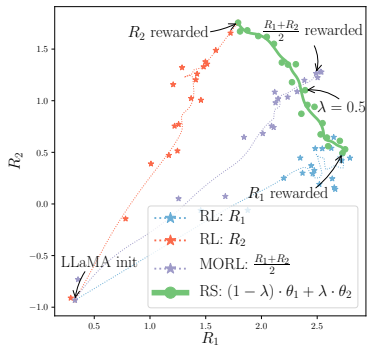
Given the importance of RLHF to train LLMs, we begin our experiments with text-to-text generation. Our pre-trained network is LLaMA-7b Touvron



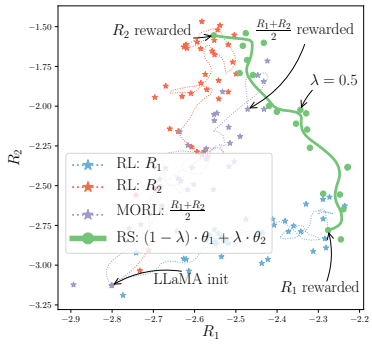
(a) News summary.

(b) Reddit TL;DR summary.

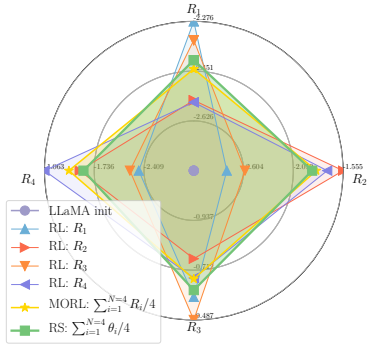
(c) Stack Exchange Q&A.



(d) Useful movie review.



(e) Helpful assistant.



(f) Helpful assistant: spider map.

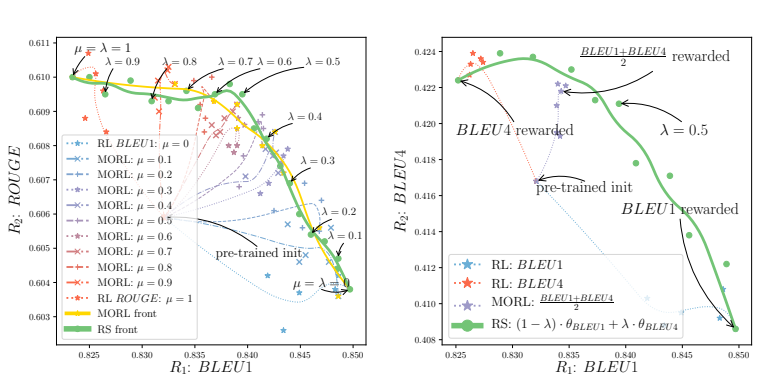
Figure 4.2 – RLHF results in NLP with LLaMA-7b Touvron et al. 2023a and reward models R_i from HuggingFace Wolf et al. 2020. The blue line reports checkpoints' results along the training trajectory of θ_1 rewarding R_1, the red line θ_2 rewarding R_2, and the purple line the MORL rewarding $\frac{R_1+R_2}{2}$. Our rewarded soup (RS) linearly interpolates between the weights θ_1 and θ_2; sliding the interpolation coefficient λ from 0 to 1 reveals the green solid front of rewarded soups solutions. In Figures 4.2a and 4.2b, we additionally show the multiple MORL runs rewarding $(1-\mu) \times R_1 + \mu \times R_2$ with preferences $0 \leq \mu \leq 1$. It reveals a similar yellow front, yet more costly. In Figure 4.2f, we uniformly ($\lambda_i = \frac{1}{4}$) average the weights fine-tuned for the assistant task on $N = 4$ reward models.

et al. 2023a, instruction fine-tuned (Y. Wang et al. 2022) on Alpaca (Taori et al. 2023). For RL training with PPO, we employ the trl package (Werra et al. 2020) and the setup from Beeching et al. (2023) with low-rank adapters (LoRA) (E. J. Hu et al. 2022) for efficiency. We first consider summarization tasks on two datasets: Reuter news (Ahmed 2017) in Figure 4.1b and 4.2a and Reddit TL;DR (Völske et al. 2017) in Figure 4.2b; the latter, hosted on HuggingFace, was created in Völske et al. (2017), and contains preprocessed posts from the social network Reddit. We also consider answering Stack Exchange questions (Lambert et al. 2023) in Figure 4.2c, movie review generation in 4.2d, and helpfulness as a conversational assistant (Bai et al. 2022a) in Figures 4.2e and 4.2f. To evaluate the generation in the absence of supervision, we utilized $N = 2$ different reward models (RMs) for each task, except in 4.2f where $N = 4$. These RMs were trained on human preferences datasets (Christiano et al. 2017) and all open-sourced on HuggingFace Wolf et al. 2020. For example in summarization, R_1 follows the "Summarize from Human Feedback" paper (Stiennon et al. 2020) and focuses on completeness, while R_2 leverages "contrast candidate generation" (S. Chen et al. 2021) to evaluate factuality. For other tasks, we rely on diverse RMs from OpenAssistant Köpf et al. 2023; though they all assess if the answer is adequate, they differ by their architectures and procedures.

The results are reported in 4.2. The green front, defined by RS between the two weights specialized on R_1 and R_2, is above the straight line connecting those two points. Second, the front passes through the point obtained by MORL fine-tuning on the average of the two rewards, supporting our claim in **RQ2**. Moreover, when comparing both full fronts, they have qualitatively the same shape; quantitatively in hypervolume (Yen and He 2013) (lower is better, the area over the curve w.r.t. an optimal point), RS's hypervolume is 0.367 0.340 for MORL in Figure 4.2a, while it is 1.176 1.186 in Figure 4.2b. Finally, in Figure 4.2f, we use $N = 4$ RMs for the assistant task and uniformly average the $N = 4$ weights, confirming that RS can scale and trade-off between more rewards.

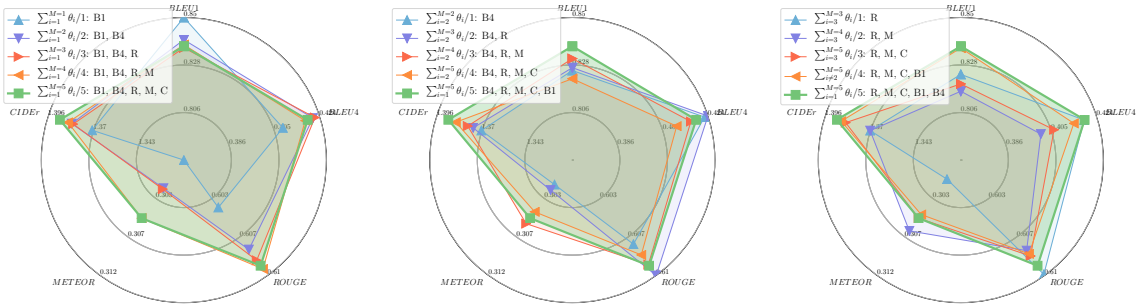
4.3.2 Image-to-text: captioning with diverse statistical rewards

RL is also effective for multimodal tasks (Pinto et al. 2023) such as in image captioning (Rennie et al. 2017), to generate textual descriptions of



(a) BLEU1 and ROUGE. (b) BLEU1 and BLEU4. (c) Captioning: spider map.

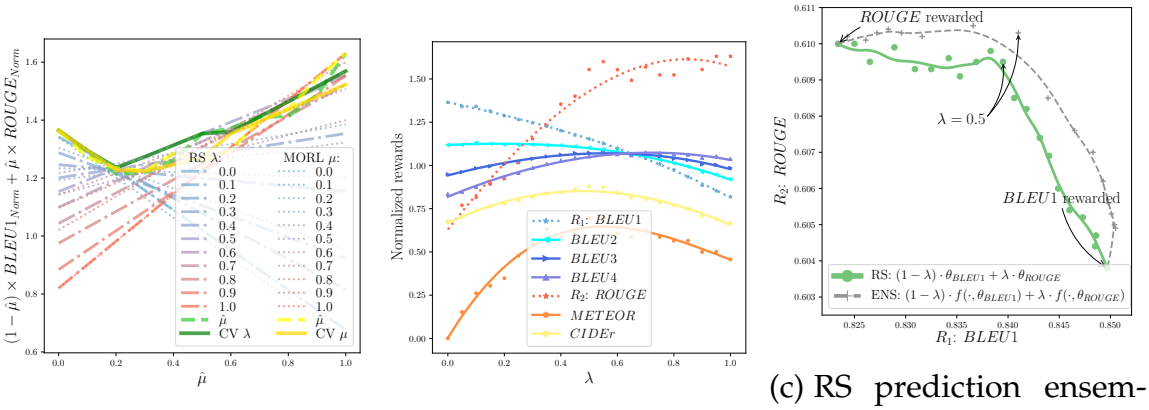
Figure 4.3 – Results in image captioning on COCO (T.-Y. Lin et al. 2014). As rewards R_1 (blue stars every epoch) and R_2 (red stars), we consider standard statistical metrics: BLEU1 (1-gram overlap), BLEU4 (4-grams overlap), ROUGE, METEOR and CIDEr. Figure 4.3a includes the MORL training trajectories optimizing $(1-\mu) \times BLEU1 + \mu \times ROUGE$, uncovering a yellow front similar to RS's green front. In Figure 4.3c, RS uniformly averages the 5 weights (one for each reward), resulting in the largest area and the best trade-off between the 5 rewards.



(a) BLEU1 first. (b) BLEU4 first. (c) ROUGE first.

Figure 4.4 – Those spider maps uniformly average $1 \leq M \leq 5$ weights for captioning, where θ_1 is fine-tuned on BLEU1 (B1), θ_2 on BLEU4 (B4), θ_3 on ROUGE (R), θ_4 on METEOR (M) and θ_5 on CIDEr (C). To show different combinations among the $\binom{5}{M}$ possible, we iterate in a clockwise direction starting in Figure 4.4a from $i = 1$ (always including θ_1 optimized on BLEU1), in Figure 4.4b from $i = 2$ (always including θ_2 optimized on BLEU4), and in Figure 4.4c from $i = 3$ (always including θ_3 optimized on ROUGE).

images. Precisely evaluating the quality of a prediction w.r.t. a set of human-written captions is challenging, thus the literature relies on various non-



(a) Analysis of **RQ3**. (b) Performances w.r.t. λ. (c) RS prediction ensembling.

Figure 4.5 – Results in captioning for $R_1 = BLEU1$ and $R_2 = ROUGE$. When normalized, rewards are set to 1 for the init and 0 for the worst model. Figure 4.5a validates **RQ3** by reporting results of RS (for varying λ) and of MORL (for varying μ) for varying user's preference $\hat{\mu}$. Figure 4.5b evaluates different rewards as a function of the interpolating coefficient. Figure 4.5c reports ensembling scores when interpolating predictions.

differentiable metrics: *e.g.* the precision-focused BLEU (Papineni et al. 2002), the recall-focused ROUGE (C.-Y. Lin and Hovy 2003), METEOR (Banerjee and Lavie 2005) handling synonyms and CIDEr (Vedantam et al. 2015) using TF-IDF. As these metrics are proxies for human preferences, good trade-offs are desirable. We conduct our experiments on COCO (T.-Y. Lin et al. 2014), with an ExpansionNetv2 (J. C. Hu et al. 2022) network and a Swin Transformer (Ze Liu et al. 2022) visual encoder, initialized from the state-of-the-art weights of J. C. Hu et al. (2022) optimized on CIDEr. We then utilize the code of J. C. Hu et al. (2022) and their self-critical (Rennie et al. 2017) procedure (a variant of REINFORCE Williams 1992) to reward the network on BLEU1, BLEU4, ROUGE or METEOR. More details and results can be found in Appendix E.

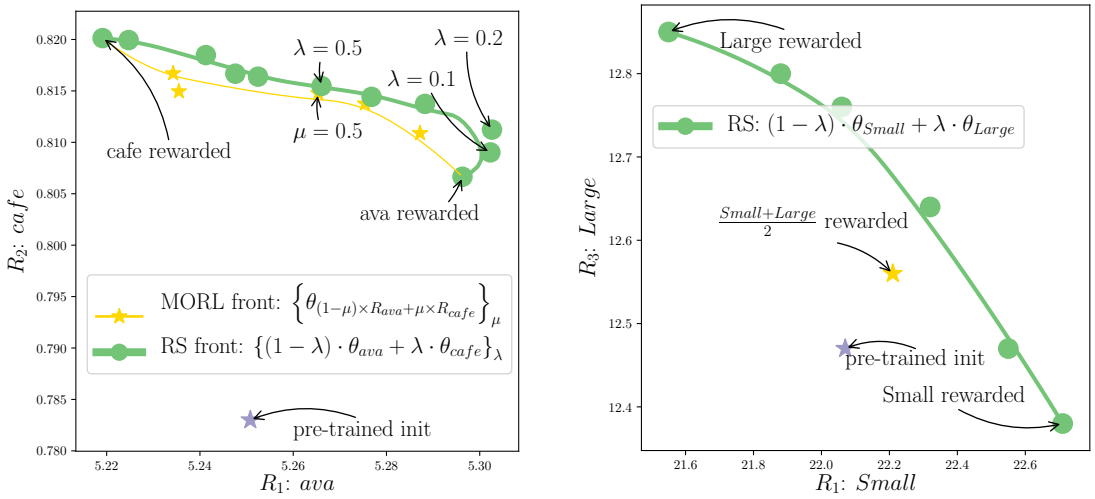
We observe in Figure 4.3 that tuning solely BLEU1 sacrifices some points on ROUGE or BLEU4. Yet interpolating between θ_1 and θ_2 uncovers a convex set of solutions approximating the ones obtained through scalarization of the rewards in MORL. When comparing both full fronts in Figure 4.3a, they qualitatively have the same shape, and quantitatively the same hypervolume (Yen and He 2013) of 0.140. One of the strengths of RS is its ability to scale to any number of rewards. In Figure 4.3c, we uniformly ($\lambda_i = \frac{1}{5}$)

average $N = 5$ weights fine-tuned independently. It improves upon the initialization and current state-of-the-art on all metrics, except for CIDEr, on which was explicitly optimized. We confirm in Figure 4.4 that RS can handle more than 2 rewards through additional spider maps. Specifically, we compare the performances across all $N = 5$ metrics when averaging $1 \leq M \leq N$ networks (each fine-tuned on one of the N rewards, thus leaving out $N - M$ rewards at training) and sequentially adding more networks to the weight average. We consistently observe that adding one additional network specialized on one additional reward extends the scope of the possible rewards that RS can tackle Pareto-optimally.Figure 4.5 refines our analysis of RS. Figure 4.5a validates **RQ3**: for any linear preference $\hat{\mu}$ over the proxy rewards, there exists an optimal solution in the set described by RS. Two empirical strategies to set the value of λ are close to optimal: selecting $\lambda = \hat{\mu}$ if $\hat{\mu}$ is known, or cross-validating (CV) λ if a different data split Karpathy and Fei-Fei 2015 is available. Moreover, Figure 4.5b (and Appendix E) investigate all metrics as evaluation. Excluding results' variance, we observe monotonicity in both training rewards, linear in BLEU1 and quadratic in ROUGE. For other evaluation rewards that **cannot be linearly expressed** over the training rewards, the curves' concavity shows that RS consistently improves the endpoints, thereby mitigating reward misspecification. The optimal λ depends on the similarity between the evaluation and training rewards: *e.g.* best BLEU2 are with small λ. Lastly, as per Izmailov et al. (2018), Figure 4.5c suggests that RS succeeds because WI approximates *prediction ensembling* (Lakshminarayanan et al. 2017) when weights remain close, interpolating the predictions rather than the weights. Actually, ensembling performs better, but it cannot be fairly compared as its inference cost is doubled.

4.3.3 Text-to-image: diffusion models with diverse RLHFs

Beyond text generation, we now apply RS to align text-to-image generation with human feedbacks (Jiazheng Xu et al. 2023). Our network is a diffusion model (Ho et al. 2020) with 2.2B parameters, pre-trained on an internal dataset of 300M images; it reaches similar quality as Stable Diffusion (Rombach et al. 2022), which was not used for copyright reasons. To represent the subjectivity of human aesthetics, we employ $N = 2$ open-

source reward models: *ava*, trained on the AVA dataset (Murray et al. 2012), and *cafe*, trained on a mix of real-life and manga images. We first generate 10000 images; then, for each reward, we remove half of the images with the lowest reward's score and fine-tune 10% of the parameters (E. Xie et al. 2023) on the reward-weighted negative log-likelihood (K. Lee et al. 2023). Details and generations for visual inspection are in Appendix F. The results displayed in Figure 4.6a validate **RQ1**, as the front described by RS when sliding λ from 0 and 1 is convex. Moreover, RS gives a better front than MORL, validating **RQ2**. Interestingly, the *ava* reward model seems to be more general-purpose than *cafe*, as RL training on *ava* also enhances the scores of *cafe*. In contrast, the model θ_{cafe} performs poorly in terms of *ava* in Figure 4.6a. Nonetheless, RS with $(1-\lambda)\cdot\theta_{ava}+\lambda\cdot\theta_{cafe}$ outperforms θ_{ava} alone, not only in terms of *cafe*, but also of *ava* when $\lambda \in \{0.1, 0.2\}$. These findings confirm that RS can better align text-to-image models with a variety of aesthetic preferences. This ability to adapt at test time paves the way for a new form of user interaction with text-to-image models, beyond prompt engineering.



(a) Image generation: *ava* and *cafe*.

(b) VG: Small and Large.

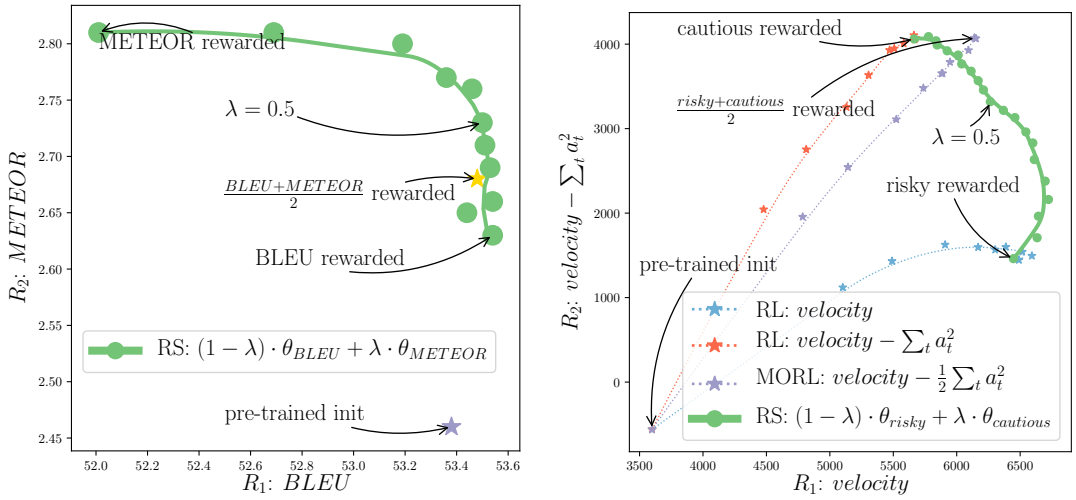
Figure 4.6 – Figure 4.6a reports our RLHF experiments on text-to-image generation with diffusion models. From the pre-trained initialization, we learn θ_{ava} and θ_{cafe} by optimizing the two reward models *ava* and *cafe*. Interpolation between them reveals the green Pareto-optimal front, above the yellow MORL front. Figure 4.6b report our results in visual grounding (VG) on RefCOCO+ (L. Yu et al. 2016), where we optimize to predict boxes with IoU> 0.5 w.r.t. the ground-truth, for objects of either small, medium or large size.

4.3.4 Text-to-box: visual grounding of objects with diverse sizes

We now consider visual grounding (VG) (L. Yu et al. 2016): the task is to predict the bounding box of the region described by an input text. We use UnIVAL (Shukor et al. 2023), a seq-to-seq model that predicts the box as a sequence of location tokens (P. Wang et al. 2022). This model is pre-trained on a large image-text dataset, then fine-tuned with cross-entropy for VG; finally, we use a weighted loss between the cross-entropy and REINFORCE in the RL stage. As the main evaluation metric for VG is the accuracy (*i.e.* intersection over union (IoU) > 0.5), we consider 3 non-differentiable rewards: the accuracy on small, medium, and large objects. We design this experiment because improving results on all sizes simultaneously is challenging, as shown in Appendix Figure 19, where MORL performs similarly to the initialization. The results in Figure 4.6b confirm that optimizing for small

objects degrades performance on large ones; fortunately, interpolating can trade-off. In conclusion, we can adapt to users' preferences at test time by adjusting λ, which in turn changes the object sizes that the model effectively handles. On the one hand, if focusing on distant and small objects, a large coefficient should be assigned to θ_{Small}. On the other hand, to perform well across all sizes, we can recover initialization's performances by averaging uniformly (in Appendix Figure 19). More details are in Appendix G.

4.3.5 Text&image-to-text: VQA with diverse statistical rewards



(a) VQA results.

(b) Locomotion: risky and cautious.

Figure 4.7 – Figure 4.7b report our results from Section 4.3.6 for the locomotion task with humanoids.

We explore visual answering questions (VQA), where the task is to answer questions about images. The models are usually trained with cross-entropy, as a classification or text generation task, and evaluated using the VQA accuracy: it compares the answer to ten ground truth answers provided by different annotators and assigns a score depending on the number of identical labels. Here, we explore the fine-tuning of models using the BLEU (1-gram) and METEOR metrics: in contrast with accuracy, these metrics enable assigning partial credit if the ground truth and predicted answers are not identical but still have some words in common. In prac-

tice, we use the OFA model (P. Wang et al. 2022) (generating the answers token-by-token), on the VQA v2 dataset, pre-trained with cross-entropy, and fine-tuned with REINFORCE during the RL stage. More details can be found in Appendix H.

Our results in Figure 4.7a validate the observations already made in previous experiments: RL is efficient to optimize those two rewards, and RS reveals a Pareto-optimal front to balance between them.

4.3.6 Locomotion with diverse engineered rewards

Teaching humanoids to walk in a human-like manner (Duan et al. 2016a) serves as a benchmark to evaluate RL strategies (Ng et al. 1999) for continuous control. One of the main challenges is to shape a suitable proxy reward (Dorigo and Colombetti 1994; Dewey 2014), given the intricate coordination and balance involved in human locomotion. It is standard (Todorov et al. 2012) to consider dense rewards of the form $R = velocity - \lambda \times \sum_t a_t^2$, controlling the agent's velocity while regularizing the actions $\{a_t\}_t$ taken over time. Yet, the penalty coefficient λ is challenging to set. To address this, we devised two rewards in the Brax physics engine (C. Daniel Freeman et al. 2021b): a risky R_1 with $\lambda = 0$, and a more cautious R_2 with $\lambda = 1$. Like in all previous tasks, RS's front in Figure 4.7b exceeds the interpolated rewards, as per **RQ1**. Moreover, the front defined by RS indicates an effective balance between risk-taking and cautiousness, providing empirical support for **RQ2**, although MORL with $\mu = 0.5$ (*i.e.* $\lambda = 0.5$) slightly surpasses RS's front. We provide animations of our RL agent's locomotion on our website, and more details are in Appendix I.

This experiment takes on the intricate challenge of controlling a running humanoid in the Brax C. Daniel Freeman et al. 2021b physics engine. The complexities involved in achieving natural or fast movement in continuous control environments serve as a testament to the robustness of our approach. The fine-tuning procedure is carried out on two distinct reward functions, with the aim of refining the running behavior of the humanoid, potentially resulting in smoother motion patterns.

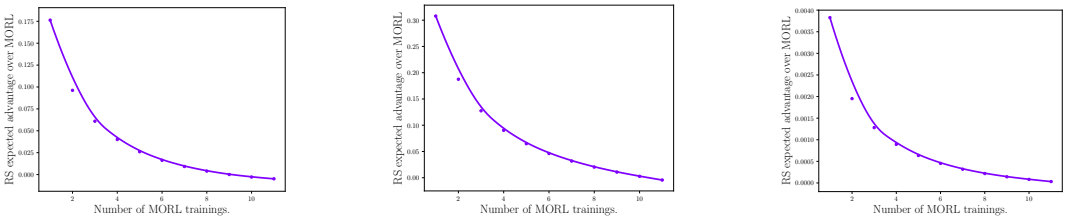
The LMC requires pre-training the base policy before fine-tuning. Thus, as the pre-training task, we use the default dense reward implemented in Brax: $R = velocity - 0.1 \times \sum_t a_t^2$. This pre-training phase also serves to

collect statistics about observations and normalize them before inputting to the model (as it facilitates training). We used the Brax implementation of PPO Schulman et al. 2017b. The pre-trained policy is saved while the value function is discarded.

We keep the same environment as in pre-training. We also use the normalization procedure inherited from pre-training but freeze the statistics. Two reward functions are designed: a *risky* one for $R_1 = velocity$ and a *cautious* one where $R_2 = velocity - \sum_t a_t^2$. We tried a few hyperparameters (see the values in brackets in Appendix Table 7 but results (see Appendix Figure 20) remain close and consistently validate our research questions.

4.3.7 Efficiency gain of RS over MORL

The efficiency gain of RS versus MORL is by design; when considering 2 rewards, RS only requires 2 fine-tunings, while MORL actually requires an infinite number of fine-tunings to reveal the entire front of preferences. To end this experimental section, we quantify this efficiency gain by introducing in Figure 4.8 the expected reward $\mathbb{E}_{\hat{\mu}\sim Unif(0,1)}\hat{R}_{\hat{\mu}}$ where $\hat{R}_{\hat{\mu}} = (1-\hat{\mu}) \times R_1 + \hat{\mu} \times R_2$ and the expectation is over all the possible user's preferences $\hat{\mu}$. We then measure the difference between the expected rewards for RS (with 2 runs) and MORL (with M runs). Plotting this expected reward advantage for different values of M shows that MORL needs $M \gg 2$ to match RS.



(a) News summary. (b) Reddit summary. (c) Captioning.

Figure 4.8 – Expected reward advantage of RS (always requiring only 2 trainings) over MORL (with M trainings), defined as $\mathbb{E}_{\hat{\mu}\sim Unif(0,1)}\left[max_{\lambda\in\Lambda}\hat{R}_{\hat{\mu}}(\theta^{RS}_{\lambda}) - \mathbb{E}_{\Lambda_M}\left[max_{\mu\in\Lambda_M}\hat{R}_{\hat{\mu}}(\theta^{MORL}_{\mu})\right]\right]$, where $\hat{R}_{\hat{\mu}} = (1-\hat{\mu})\times R_1 + \hat{\mu}\times R_2$ is the user reward for user linear preference $\hat{\mu}$ sampled uniformly between 0 and 1, $\Lambda = \{0, 0.1, , 1.0\}$ is the set of the 11 possible values for λ, and where the expectation for the MORL term is over the $\binom{11}{M}$ possible combinations Λ_M of M elements from Λ (representing the M linear weightings μ used for MORL training). We observe that MORL matches RS only for M sufficiently big.

4.4 Conclusion

As AI systems are increasingly applied to crucial real-world tasks, there is a pressing issue to align them to our specific and diverse needs, while making the process more transparent and limiting the cultural hegemony of a few individuals. In this chapter, we propose rewarded soup, a strategy that efficiently yields Pareto-optimal solutions through weight interpolation after training. Our experiments have consistently validated our working hypotheses for various significant large-scale learning tasks, demonstrating that rewarded soup can mitigate reward misspecification. We hope to inspire further research in exploring how the generalization literature in deep learning can help for alignment, to create AIs handling the diversity of opinions, and benefit society as a whole.

However, several limitations are observable: 1) Throughout training, none of the models benefited from each other, limiting this method to a mere *post-hoc* selection. 2) While the reward function may vary, the inputs remain substantially the same across experiments, raising the question: would we observe the same smoothness post-training by interpolating weights in

more ambitious settings? The next part introduces a novel concept: *subspace of policies*, which aims to address these observed limitations.

PART III

Subspace of Policies

CHAPTER 5

DEFINING A SUBSPACE OF POLICIES

As discussed in Section 2.3 and 2.4 the pinnacle of adaptation in RL is achieved when an agent, already trained in fundamental problems, can seamlessly navigate through a diverse array of behaviors to address any given new challenge. This concept endorses the development of a *population of agents* (Long et al. 2023): rather than focusing on a single policy, maintaining an entire manifold of policies significantly increases the chances of identifying optimal behavior for specific scenarios. In this part, we define a new methodology for evolving and refining this manifold, setting the stage for a sophisticated approach aimed at constructing this dynamic policy landscape.

This chapter lays the foundations for understanding the origins and defining this approach we created during this thesis—called *Subspaces of Policies*—which will be further explored in Chapters 6 and 7. The core idea behind the Subspaces of Policies is to train and maintain a population of diverse yet effective policies in a memory-efficient manner, drawing upon the principles of *mode connectivity*, a cornerstone concept in neural network optimization that reveals the interconnected nature of solution spaces. This approach, founded on distinct hypotheses, facilitates the creation of neural networks by employing *weight interpolation*. This technique enables the selection of a specific set of parameters within the trained subspace, with the anticipation that this novel parameter configuration will yield different responses to inputs (*i.e.* all in all, actions output by policies will differ), thereby eliciting new behaviors. Through this mechanism, we aim to explore and harness the potential diversity within the parameter space, offering a systematic way to generate a spectrum of adaptable and nuanced policies for varied scenarios.

We start with a toy example in Section 5.1 to contextualize and impart the intuition behind our method. Section 5.2 delves into the relevant literature, transitioning from theoretical underpinnings to contemporary appli-

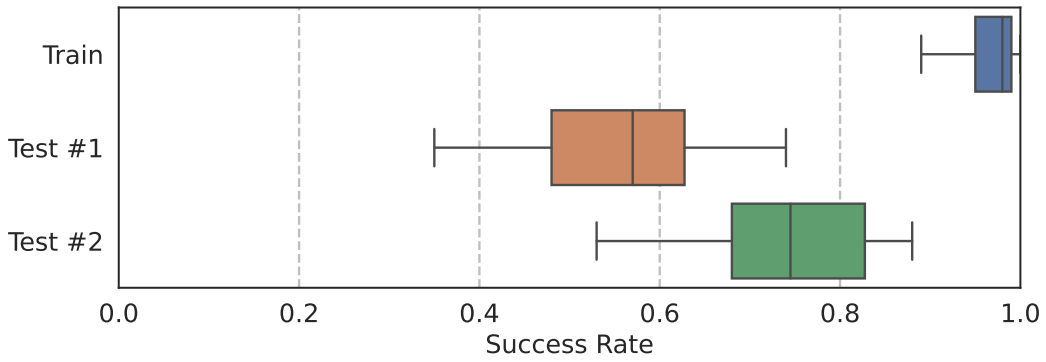


Figure 5.1 – Distribution of the success rate of a population of 100 trained policies. At the end of the training, we rollout trajectories with each policy on 3 different environments: Train (Ant, the one used for training), Test #1 (Ant with a Torso 20% bigger), Test #2 (Ant with legs 30% smaller). Success rate is averaged on 100 rollouts. The task remains the same for all environments (navigating from point A to B)

cations. Subsequently, we formalize the concept of *neural network subspaces* in Section 5.3 and elucidate our novelty *subspaces of policies* in Section 5.4, supplemented by initial observations and remarks.

5.1 Towards a Population of Policies

Training a single policy on a given MDP is often perceived as an acquisition of skills or knowledge, thereby broadening the potential for performance: the expectation is that the agent will be able to solve a task (the training one), and perhaps more since it acquired knowledge. From an optimization perspective, however, the narrative is different: we iteratively update the policy weights $\theta \in \mathbb{R}^d$, progressively constraining it within a local minimum from which escape is often challenging. Beyond the risk of overfitting, the optimization phase is parametrically restrictive: we incrementally solidify the weights towards an immutable parametric solution. While this phase is necessary for knowledge acquisition, it is detrimental to *functional diversity*: unless the policy is retrained, it is destined to behave in one specific manner at test time (i.e., when faced with a different task).

To empirically illustrate this point, consider the following toy example. Let us take a navigation task as described in Section 3.4: we train 100 policies with PPO on the same MDP, using the Ant environment from Brax (C. Daniel Freeman et al. 2021b). Each policy is a neural network with two layers of 512 units, randomly initialized (i.e., all weights are different). At the end of training, we examine the distribution of success rate in the training environment among the population (Figure 5.1): all policies achieve a success rate above 90%, with very low standard deviation (0.03 see blue bloxplot). While these policies appear to perform similarly, they have almost certainly converged to radically different solutions within the parameter space $\mathbb{R}^d$. This is a well-known characteristic of deep learning models, resulting from the highly non-convex nature of their loss landscapes: a NN's learning trajectory is profoundly impacted by its initial conditions(Glorot and Bengio 2010). But what of their functional diversity? To evaluate this, we test the policies in two slightly modified environments depicted as Test #1 and Test #2 (see orange and green boxplots in Figure 5.1) by performing "0-shot" rollouts to measure their success rate. The results are unequivocal: there is significant variance in success rates (standard deviation of respectively 0.12 and 0.10). Some policies naturally perform well in this new environment, while others fail completely, confirming the presence of substantial functional diversity among them. One might argue that, perhaps, among the learned policies, some may be robust to any type of environment and the challenge would be to find a way to identify the best one and discard the rest. This is why Test #2 was introduced: Figure 5.2 shows us that the policies (one dot refers to one single policy) that perform well in Test #1 (the color depicts the performance on this particular test set) are not the same as those that perform well in Test #2 (e.g. blue dots representing good policies at Test #1 are shuffled in Test #2 plot). This toy example leads to two critical insights: 1) Similar performance levels across policies trained on the same task can mask underlying functional diversity, which becomes apparent when they are subjected to slightly different tasks. 2) It is impossible to predict which policy will demonstrate robustness and efficiency in response to changes, as the optimal policy varies with the nature of the change.

In an ideal world, we could "carry" this entire population of policies at test time and select them based on performance. This, however, is infeasible for two reasons: computational cost (both memory and inference) and the

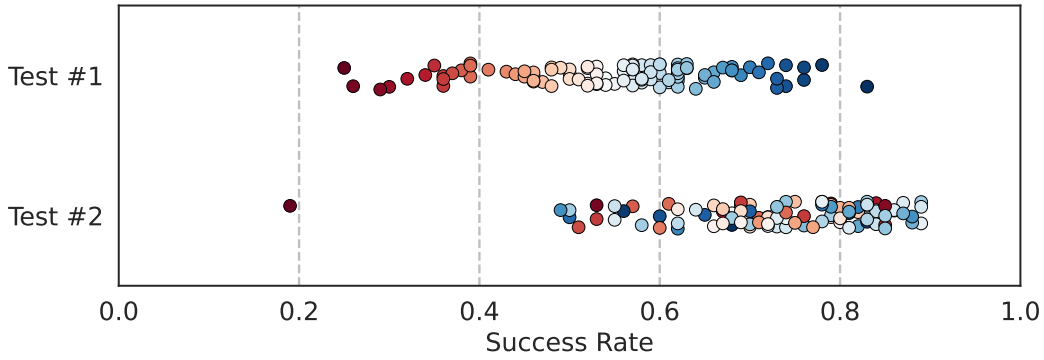


Figure 5.2 – Same results as Figure 5.1 but with a split per policy. The color reflects the ranking of the policies with respect to success rate on Test #1 to verify that these are not the same that perform well on Test #2. The color scale is indeed completely shuffled on Test #2.

.

interaction cost (for k policies, at least k episodes are needed to estimate their performance). Hence, the concept of subspaces of policies emerges: train a population of policies without fully instantiating them, that is, characterizing them within a much smaller parametric space.

5.2 The history of mode connectivity

Before delving into the formalization of subspaces, I find it interesting to outline a brief history of mode connectivity to elucidate the origins of subspaces. The ideas tested in the Chapters 6 and 7 are not merely based on intuition but are deeply rooted in theoretical realities that mode connectivity continues to explore and demonstrate to this day.

The concept of mode connectivity has dramatically shifted the perception of Neural Network optimization, highlighting a landscape of interconnected solutions rather than isolated optima. This transformative perspective historically challenged the long-standing notion of isolated minima in non-convex optimization landscapes, proposing instead a richly interconnected topography within neural networks. Within this framework, the introduction of the *connector*—a pathway through the weight space—serves as a pivotal element. Given some weights $\theta_i \in \mathbb{R}^d$, a single-parametrized connec-

tor is a continuous function $\alpha : [0, 1] \to \mathbb{R}^d$ that ensures that the NN created along $[0, 1]$ perform similarly or better . Recalling what we have seen in the last Section 5.1 in terms of population of policy, if such connector exists between e.g. two NN parametrized by θ_1 and θ_2, this could be a way to bring a whole manifold of policies at a small cost (twice a single policy). So, does it exist and if yes, how to find it ?

In the mid-2010s, theoretical analysis of NNs optimization emerged. As a pioneer paper in this field, Choromanska et al. (2015), has paved the way for *loss landscape analysis*: when the number of parameters in a neural network increases, the parameter space surprisingly offers an exponential growth of viable solutions for the test set loss. Do these solutions have more in common than good performances at test time ? If so, how are they connected ? These analytical forays have given birth to the concept of *Mode Connectivity*. Lakshminarayanan et al. (2017) make a first interesting observation on ensembles. Ensembling two trained NN in *output space* boosts accuracy, calibration, and robustness. This is attributed to functional diversity meaning the two neural networks make different errors. On the contrary, Fort et al. (2020) show that ensembling them in the *parameter space* - i.e. perform a *weight interpolation* - dramatically fails and achieves a performance near to an untrained NN. Garipov et al. (2018) introduce the notion that optima are not isolated, but are instead connected by simple, high-accuracy curves (for instance a quadratic Bezier curve, that we will talk about in Section 5.4.3). This revelation paves the way for innovative training and ensembling methods. In the same paper, they introduce Fast Geometric Ensembling that exploits these paths to enhance model performance and robustness. Building upon these insights, subsequent studies have delved deeper into the implications and applications of mode connectivity. Draxler et al. (2019) further challenge the traditional view of distinct valleys in loss landscapes by demonstrating surprisingly flat continuous paths between minima, suggesting a more unified and smooth energy landscape than previously believed. Similarly, Gotmare et al. (2018) extend the understanding of mode connectivity, affirming its resilience across varied training and initialization conditions and leveraging it as a diagnostic tool for loss landscape analysis. These works collectively contribute to a growing body of literature that reconceptualizes the optimization terrain of neural networks as an interconnected manifold rather than a disjointed collection of solutions.

In parallel, efforts to understand and optimize neural network behavior have led to methodologies like those proposed by Lakshminarayanan et al. (2017) and Izmailov et al. (2019), which, while not directly addressing mode connectivity, resonate with its underlying principles. Lakshminarayanan et al. (2017) advocate for ensembles as a means of enhancing predictive uncertainty, echoing mode connectivity's theme of leveraging multiple solutions. Izmailov et al. (2019) introduces Stochastic Weight Averaging (SWA), a technique that navigates the connected regions of loss surfaces for better generalization, embodying the spirit of exploring the connexion between minima. Furthermore, the implications of mode connectivity extend into practical applications and theoretical explorations alike. Frankle et al. (2020) investigate the intersection of mode connectivity with the lottery ticket hypothesis, revealing how neural networks stabilize to specific linearly connected regions under SGD noise. This work not only deepens the understanding of network optimization dynamics but also illustrates mode connectivity's role in finding efficient network structures.

Above all, the research highlighted in Wortsman et al. (2021) demonstrated that, under certain conditions, it is entirely feasible to train a connector rather than searching for one post hoc, enabling the *training* of high-performance ensembles through *weight interpolation* within these *subspaces*. This paradigm shift not only corroborates the interconnected nature of neural network solutions but also furnishes a computationally efficient framework—unlike previous methods in the mode connectivity literature—for capturing diverse, high-accuracy models. This work further advances our understanding by demonstrating the practicality and benefits of directly learning neural network subspaces, which include diverse and high-accuracy models within a single training run. The ability to learn such subspaces efficiently not only validates the existence of high-accuracy paths connecting different models but also showcases a method to build these paths from scratch for improved generalization and robustness against label noise. Partly inspired by this groundbreaking work, our forthcoming section lays the groundwork for apprehending the subspace of policies.

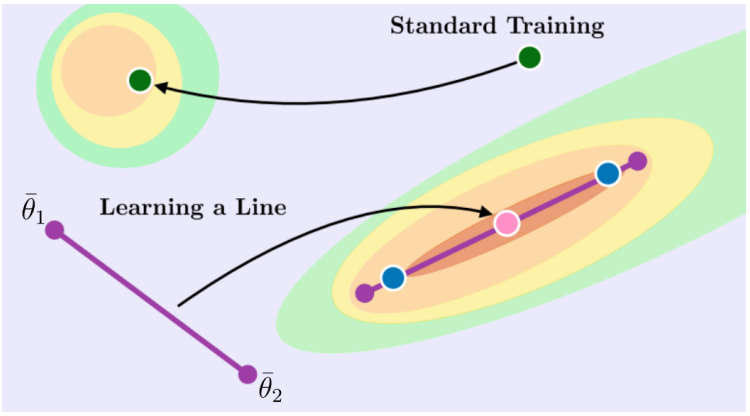


Figure 5.3 – Illustration adapted from Wortsman et al. (2021) showing the training of a line of neural networks parameterized by its two anchors $\bar{\theta}_1$ and $\bar{\theta}_2$, compared to training a single neural network.

5.3 Neural Network Subspaces

Following the work of Wortsman et al. (2021), we define now the Neural Network Subspaces and add preliminary important remarks.

5.3.1 Neural Network Convex Hull

Let us consider two NN, each defined by a unique set of d parameters within the d-dimensional space R^d, symbolized as θ_1 and θ_2. These networks delineate the boundaries of a segment in parameter space composed of all their convex combinations. Given Θ the space of all parametrized NN, this segment $\bar{\Theta} \subseteq \Theta$ is denoted mathematically as $\bar{\Theta} = \{\theta : \theta = \alpha \cdot \theta_1 + (1-\alpha) \cdot \theta_2, \alpha \in [0, 1]\}$, the combination $\alpha \cdot \theta_1 + (1 - \alpha) \cdot \theta_2$ being applied element-wise. In this context, the NNs define the boundaries of the subspace and are thus referred to as *anchors* throughout this thesis, denoted by $\bar{\theta}$. This segment exemplifies the simplest form of a subspace, but the generalization to n NN is formally trivial. The subspace is then the *convex hull* of these n NN:

Definition 5.1: Neural Network Convex Hull
Given n neural networks each defined by a set of parameters $\{\theta_i\}_{i=1}^n$ in a d-dimensional space R^d, the convex hull of these networks $\bar{\Theta}$, is the set of all their convex combinations. Mathematically, it is defined as

$\bar{\Theta} = \{\theta : \theta = \sum_{i=1}^{n} \alpha_i \theta_i, \ \sum_{i=1}^{n} \alpha_i = 1, \alpha_i \in [0, 1]\}$, where each combination is element-wise.

5.3.2 Learning objective

Consider the task of supervised learning where we aim to minimize a loss function $\mathcal{L}_(x, y; \theta)$ over a dataset. If we want all the policies of a subspace to be good at this task, the learning loss for a subspace could then be represented as

$$\mathcal{L}_{\text{subspace}}(x, y; \bar{\Theta}) := \int_{\Theta} \mathcal{L}(x, y; \theta) d\theta$$

In practice we can use similar optimization methods as for a single NN by optimizing this surrogate loss:

$$\mathbb{E}_{\alpha \sim p(\alpha)}[\mathcal{L}(x, y; \sum_{i=1}^{n} \alpha_i \cdot \bar{\theta}_i)] \tag{5.1}$$

where $p(\alpha)$ denotes a distribution over the simplex $[0, 1]^n$. When $\bar{\Theta} = \{\bar{\theta}_1, \bar{\theta}_2\}$ one can simply use the uniform distribution. This particular case is illustrated by Figure 5.3.

5.3.3 Other types of combinations

The use of *Bézier curves*, as opposed to simple convex combinations, has been experimented with varying degrees of success, as documented in (Wortsman et al. 2021) and discussed in Section 6.4.6. This approach requires at least three anchors and involves the use of Bézier parametric curves based on Bernstein polynomials, constrained to the interval $[0, 1]$, with the anchor parameters serving as the control points. For $N = 3$, it is defined by:

$$\bar{\Theta} = \left\{(1 - \alpha)^2 \, \bar{\theta}_1 \ + \ 2\,(1 - \alpha)\,\alpha \, \bar{\theta}_2 \ + \ \alpha^2 \, \bar{\theta}_3, \quad \forall \alpha \in [0, 1]\right\} \tag{5.2}$$

5.3.4 Computational remarks

Practically, Equation 5.1 can be minimized using any standard first-order gradient descent method, such as Adam (Kingma and Ba 2017), a variation of SGD (Robbins and Monro 1951) always used in our experiments, without significant computational overhead. Specifically, during inference, the only extra step in each layer involves averaging the weights and biases before the usual forward operation. During back-propagation, thanks to the linearity of differentiation, gradients are simply scale and used to update neural network anchor weights in accordance with the corresponding coefficient α_i. While this approach indeed increases memory usage, it does not necessitate additional memory for the backward pass, often the primary computational bottleneck (T. Chen et al. 2015).

```
import torch
import torch.nn as nn

class Linear(nn.Module):
    def __init__(self, n_anchors, in_features, out_features,
   bias = True):
        super().__init__()
        self.n_anchors = n_anchors
        self.in_features = in_features
        self.out_features = out_features
        self.is_bias = bias

        #initializing anchors
        anchors = [nn.Linear(in_features, out_features, bias=
   self.is_bias) for _ in range(n_anchors)]
        self.anchors = nn.ModuleList(anchors)

    def forward(self, x, alpha):
        # forwarding on all anchors
        xs = [anchor(x) for anchor in self.anchors]
        xs = torch.stack(xs,dim=-1)

        # averaging outputs with the convex combination
        alpha = torch.stack([alpha] * self.out_features, dim
   =-2)
        xs = (xs * alpha).sum(-1)
        return xs
```

Listing 5.1 – Python code to define the class of a subspace NN linear layer `n_anchors` using Pytorch.

During this thesis, we figured out that subspace methods involving up to 8 anchors incur only a 10% to 20% increase in training time compared to training a single policy. From an implementation perspective, as demonstrated in the code above, creating such anchors using Pytorch (Paszke et al. 2019) is quite straightforward. A particularly useful trick for CNN architectures: using the `groups` argument in the Pytorch `torch.nn.Conv2d` function and adjusting `in_channels` and `out_channels` accordingly. For instance, for `groups =2` changes the convolution operation, as if two convolutional layers were operating in parallel on separate halves of the input channels and then concatenating their outputs. One only has to weight average and sum the output to get a CNN subspace.

5.4 Neural Network Subspaces of Policies

This thesis contributes to the exploration of neural network subspaces of policies within the realm of RL, offering insights into previously unexamined aspects. Our first contribution in the field consists in adapting the NN subspace to the subspace of policies. In what follows, we present our formalization and discuss the underlying challenges.

We define policies π_θ parametrized by neural networks θ, where $\bar{\Theta}$ represents the set of such policies $\bar{\Pi} := \{\pi_\theta\}_{\theta \in \bar{\Theta}}$. By utilizing two anchor points $\bar{\theta}_1$, $\bar{\theta}_2$ and a reward function $R : \tau \to \mathbb{R}$ aimed at maximizing returns (with τ denoting a trajectory), we introduce a surrogate objective. This objective focuses on maximizing the average performance of policies within the subspace induced by $\bar{\theta}_1$ and $\bar{\theta}_2$, as detailed in:

$$\max_{\bar{\theta}_1, \bar{\theta}_2} \; \mathbb{E}_{\alpha \sim \mathcal{U}([0:1])} \left[\mathbb{E}_{\tau \sim \pi_{\alpha\bar{\theta}_1 + (1-\alpha)\bar{\theta}_2}} \left[R(\tau) \right] \right] \tag{5.3}$$

This equation will play a role in the objective discussed in Chapter 6. Its extension to n anchors, enabling the transformation of Equation 5.1 into a Reinforcement Learning objective, is straightforward. However, a critical question arises regarding the choice of distribution for sampling

within larger simplexes, especially when dealing with a greater number of anchors (*i.e.* more than two).

5.4.1 Sampling within the subspace

Sampling possibilities in this context are more varied than in traditional supervised learning, as combinations can be changed during a trajectory. It's even conceivable to use distinct combinations for rolling out and updating the policy, akin to the flexibility offered by off-policy algorithms (see our algorithm developed in Chapter 7). This opens up numerous possibilities and significantly enhances rollout diversity during training. Rather than sampling from a singular policy repeatedly, sampling occurs across various policies, similar to population methods that use multiple policies to increase diversity and robustness (see end of Section 2.3.2). Sampling can also be employed post-training during inference, serving as an efficient means to narrow down the search space compared to the full d-dimensional parameter space Θ. In this regard, the subspace strikes a balance between traditional optimization and evolutionary algorithms, which typically perturb parameters randomly. The subspace, however, channels the search within an area that has been tailored for a specific task.

The Dirichlet distribution is particularly advantageous for its ability to sample n positive random variables that sum to 1. Defined by parameters $a = (a_1, ..., a_n)$, which act as concentration parameters, it modulates the probability distribution across the n variables, rendering it a flexible distribution over a convex hull—ideal for subspace sampling. Different sampling strategies can be employed depending on the tuning of parameter a, as illustrated in Figure 5.4. We will use the flat Dirichlet distribution where $a = (1, ..., 1)$, known for its uniform distribution over the simplex $[0, 1]^n$. In Chapter 7, we will also experiment with a peaked distribution $a = (\frac{1}{n}, ..., \frac{1}{n})$, favorable for sampling around the anchors. The potential for integrating *active learning* strategies (Settles 2009; Fang et al. 2017), which could adapt the distribution parameters based on task-specific information or obtained rewards, and this during an episode, is also an intriguing prospect.

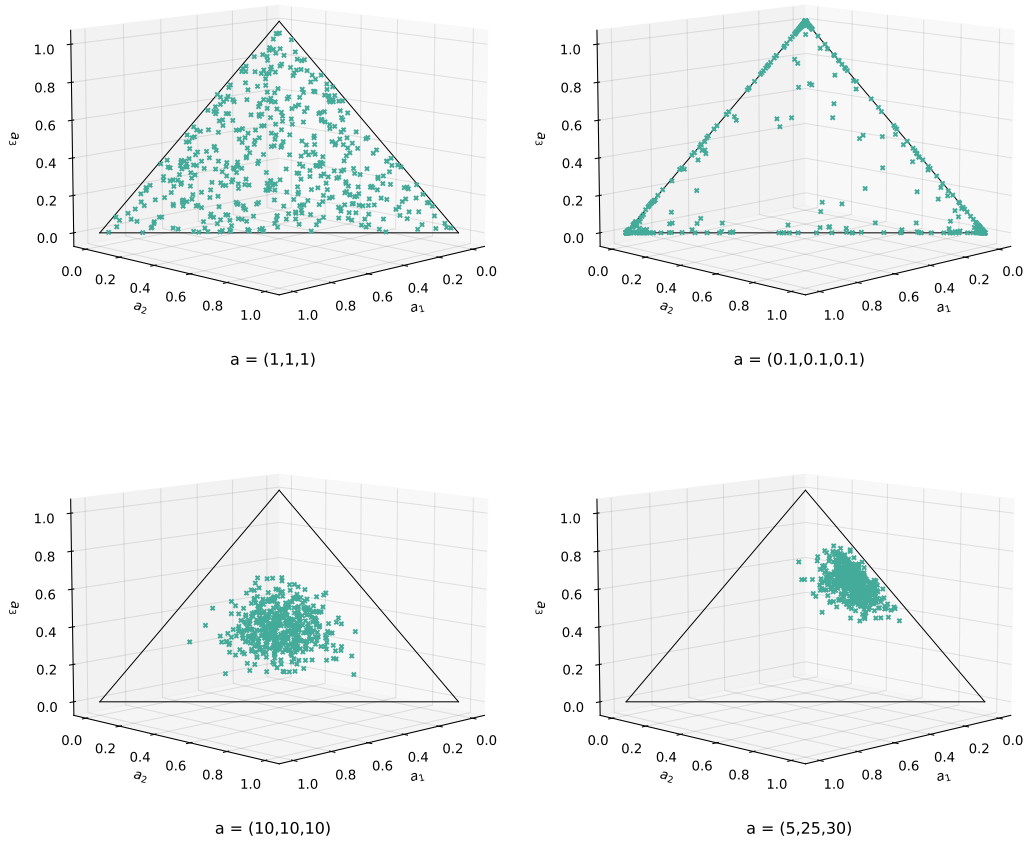


Figure 5.4 – 3D visualization of 1000 random samples using various Dirichlet distribution sampling strategies, with a representing the distribution's parameters. The "triangle" symbolizes the simplex defined by vertices $(1, 0, 0), (0, 1, 0), (0, 0, 1)$.

5.4.2 Freezing anchors

One potential issue is the deterioration of performance when training all anchors on a sequence of tasks. An intriguing strategy involves selectively freezing certain anchors while optimizing others. This approach, detailed in Chapter 7, allows for the preservation of learned behaviors in some anchors, particularly when they have adequately adapted to specific tasks, while continuing to evolve other anchors. This selective freezing can lead to a nuanced balance in the learning process, where certain stable aspects of the policy are maintained, while others are dynamically adjusted. One can even consider partial freezing strategies, akin to those in Mallya and

Lazebnik 2018, where only specific parts of a neural network are frozen. Such methods can result in interesting architectural combinations, mixing the concept of *modularity* like in Veniat et al. 2020 with the flexibility of subspace optimization. This approach opens up new avenues for exploring efficient and adaptive policy development in CRL.

5.4.3 Other types of combinations

Exploring alternative combination methods can significantly enhance the expressivity of the subspace. By extending the dimension of the combination variable α, it's possible to access a broader array of diverse policies. *Layer-wise combination* emerges as an obvious extension of the classical combinations. Here, distinct combinations are applied to each layer of the NN. A specific instance of this is the mixing of only the final layers between neural networks, as explored in Chapter 4. This strategy is particularly intriguing as it allows for the use of subspace methods in the final layer, while potentially employing different methodologies for the earlier layers or the *backbone* of the network, a concept we experiment in Section 7.4 on Continual World scenario. While not extensively tested, the idea of extending this to an *element-wise combination*, where the number of coefficients equals the number of parameters, presents an interesting avenue. Though this would significantly increase the number of parameters, effectively creating a new neural network to optimize, the potential for enhanced adaptability and performance is noteworthy.

5.5 Conclusion

This chapter has laid the groundwork for the novel framework of *subspaces of policies*, anchored in the principle of mode connectivity. We have demonstrated not only a potentially powerful framework—supported by robust theory—for training a subspace of policies but also identified various properties that are advantageous, such as the relatively low inference cost and the diverse methods available for sampling within the subspace. Moving forward, we will expand on these foundational elements to devise innovative methods applicable across a range of scenarios. We will progress

to a few-shot setting using an simple training method in Chapter 6, including an effective policy selection technique to identify the optimal point within a subspace. Chapter 7 sets its sights higher, seeking to iteratively build a subspace of policies to augment knowledge within a continual reinforcement learning framework.

CHAPTER 6

LEARNING A LINE OF POLICIES

This work is based on our paper *Learning a Subspace of Policies for Online Adaptation in Reinforcement Learning*[1] (Gaya et al. 2023) (ICLR). As we saw in Chapter 2, existing RL techniques work quite well in the classical setting where one have to maximize a unique reward function. Yet, considering that the environment at train time and the environment at test time are similar is unrealistic in many practical applications. As an example, when a gamer learns to play a video game, they often practice within a specific virtual environment, mastering certain levels with consistent patterns and predictable challenges. However, at competition time, we expect the gamer to be able to generalize their skills to new levels, unfamiliar gameplay mechanics, and unexpected in-game events. This requires the player to adapt their strategies and reflexes to a variety of unforeseen situations, paralleling the adaptability we aim to achieve in deep reinforcement learning agents. All in all, the challenges is to learn a policy that adapts itself to *unseen environments*. Different techniques have been proposed in the literature (Section 2.3) to automatically adapt the learned policy to the test environment. In the very large majority of works, the model has access to *multiple* training environments (meta-RL setting). Therefore, the training algorithm can identify which variations (or invariants) may occur at test time and how to adapt quickly to similar variations. But this setting may still be unrealistic for concrete applications: for instance, it assumes that the gamer will have had the chance to play against all the other competitors in practice rounds before the actual tournament begins.

In this chapter, we thus address this few-shot adaptation setting - hard to tackle - in which the agent is trained over one single environment and has to perform well on different unseen test environments; preventing us from using the meta-RL approaches discussed in Section 2.3.2. It introduces our innovative approach, detailed in our ICLR publication *Learning a Subspace*

1. You can find the appendix in the arxiv paper here. Code is available here.

of Policies for Online Adaptation in Reinforcement Learning (Gaya et al. 2021), marking the first practical exploration of policy subspaces. It is sectioned into five areas: Section 6.1 outlines the contextual challenges and the specific setting we are addressing; Section 6.2 describes the methodological approach; Section 6.3 details the experimental protocol we have employed; and Section 6.4 provides a thorough analysis of the results. The final Section 6.5 offers our concluding remarks on the study and pave the way for Chapter 7.

6.1 Context & Motivations

We consider the setting depicted in Section 2.3.1 where an agent is trained on a MDP, and has only K "free" episodes to adapt and be tested on another mdp. Note that we are interested in methods that adapt quickly to new a test environment and we will consider small values of K in our experiments. In this Chapter, for sake of simplicity, K will refer to the number of policies evaluated during adaptation since each policy may be evaluated over more than a single episode when facing stochastic environments.

A natural way to address this setting is by initially learning a single policy using any RL algorithm and fine-tuning it at test time over the test environment (see (a) in Figure 6.1), but this can be costly in terms of environment interactions. Because of the number of interactions required to make them work, this also excludes many methods derived from MAML (Finn et al. 2017) and described in Section 2.3. On the other side of the spectrum, one can learn a lot of policies and select it at test time, but this requires much more in terms of interactions, memory and computation time (see (b) in Figure 6.1). Drawing upon the insights from Chapter 5, we propose to learn a *Line of Policies* in the parameter space. Each particular point in this subspace corresponds to specific parameter values, and thus to a particular policy. At test time, one can rollout trajectories from multiple policies just as if we had trained much more than two (see (c) in Figure 6.1). Indeed, this subspace is learned by adapting a classical RL algorithm (without additional computation cost), such that an infinite continuum of policies is learned, each policy having different parameters. The policies thus capture and process information differently, and react differently to variations of the training environment.

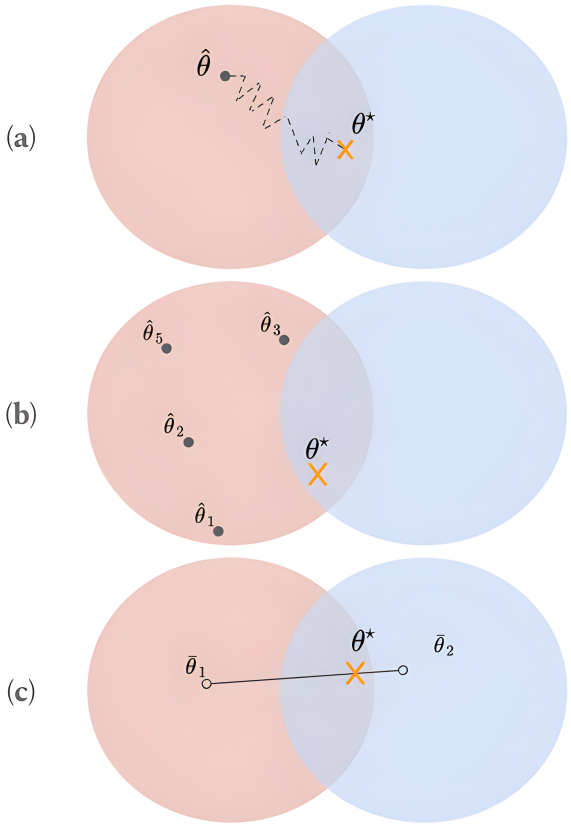


Figure 6.1 – Illustration of policy adaptation strategies in reinforcement learning. (a) represents the traditional approach of learning a single policy and fine-tuning it over the test environment, which may lead to high interaction costs. (b) depicts the strategy of learning multiple policies and selecting the best at test time, incurring high interaction, memory, and computational costs. (c) introduces our proposed method of learning a *Line of Policies* within the parameter space.

6.2 Line of Policies (LoP)

6.2.1 Intuition

To illustrate our idea, let us consider a toy example where the train environment contains states with correlated and redundant features, in such a way that multiple subsets of state features can be used to compute good actions to execute. Traditional RL algorithms will discover one policy π_{θ^*} that is optimal w.r.t the environment. This policy will typically use the state features in a particular way to decide the optimal action at each step. If some features become noisy (at test time) while, unluckily, π_{θ^*} particularly

relies on these noisy features, the performance of the policy will drastically drop. Now, let us consider that, instead of learning just one optimal policy, we also learn a second optimal policy $\pi_{\theta^{*'}}$, but enforcing $\theta^{*'}$ to be different than θ^*. This second policy may tend to make use of various features to compute actions. We thus obtain two policies instead of one, and we have more chances that at least one of these policies is efficient at test time. Identifying which of these two policies is the best for the test environment (i.e., adaptation) can simply be done by evaluating each policy over few episodes, keeping the best one. The model we present is built on top of this intuition, extending this example to an infinite set of policies and to variable environment dynamics.

We study the approach of *learning a subspace of policies in the parameter space*, and the use of such a model for online few-shot adaptation in reinforcement learning. Studying the structure of the parameter space has seen a recent surge of interest through the *mode connectivity* concept (see Chapter 5) and obtain good results in generalization, but it has never been involved in the RL setting. As intuitively illustrated in the previous paragraph, we expect that, given a variation of the training environment, having access to a subspace of policies that process information differently instead of a single policy will facilitate the adaptation. As a result, our method is very simple, does not need any extra hyper-parameter to tune and achieve good performance.

In the case of $N = 2$, the subspace of policies corresponds to a simple segment in the parameter space defined by $\bar{\theta}_1$ and $\bar{\theta}_2$ as extremities. $\bar{\theta}_1$ and $\bar{\theta}_2$ are combined through a single scalar value $\alpha \in [0; 1]$ as follows: $\theta = \alpha\bar{\theta}_1 + (1 - \alpha)\bar{\theta}_2$. One could maximize the objective function stated in Equation 6.1, by rolling out trajectories from random policies along the line of the subspace and optimize the anchors accordingly. But this lacks of a regularization term to encourage *functional diversity*.

6.2.2 Increasing functional diversity

Indeed, one possible effect when optimizing $\mathcal{L}_{\text{subspace}}(\bar{\theta}_1, \bar{\theta}_2)$ from Equation 6.1 is to reach a solution where $\bar{\theta}_1$ and $\bar{\theta}_2$ are similar. In that case, all the policies would have the same parameters value, and will thus all achieve the same performance at test-time, producing no functional diversity. Since

Algorithm 6.1 PPO on a Line of Policies

Initialize: $\bar{\theta}_1, \bar{\theta}_2$, ϕ (Critic), n batch size

for $k = 0, 1, 2...$ **do**

  Sample $\alpha_1, ..., \alpha_n \sim \mathcal{U}_{[0,1]}$

  Define $\theta_{\alpha_i} \leftarrow \alpha_i \bar{\theta}_1 + (1 - \alpha_i)\bar{\theta}_2$

  Sample trajectories $\{\tau_i\}_1^n$ using $\{\pi_{\theta_{\alpha_i}}\}$

  Update $\bar{\theta}_1$ and $\bar{\theta}_2$ to maximize: $\frac{1}{n}\sum_{i=1}^{n} \widehat{\mathcal{L}}_{PPO}\left(\theta_{\alpha_i}\right) - \beta \cdot C\left(\bar{\theta}_1, \bar{\theta}_2\right)$

  Update ϕ to minimize: $\frac{1}{n}\sum_{i=1}^{n} \widehat{\mathcal{L}}_{MSE}\left(\phi, \alpha_i\right)$

end

Figure 6.2 – The adaptation of the PPO Algorithm with the LoP model. The different with the standard PPO algorithm is that: a) trajectories are sampled using multiple policies θ_{α_i} b) The policy loss is augmented with the auxiliary loss, and c) The critic takes the values α_i as an input.

we want to encouragithge the policies to process information, we force the anchor policies to have different parameters. This is implemented through the use of *cosine similarity*, a measure of similarity between two vectors. (Fort et al. 2020) found that models trained separately tend to have weight vectors that are not similar in direction, which they measured using cosine similarity close to 0. This is in contrast to models that are trained together, following the same path. To replicate this effect, we follow Wortsman et al. (2021) and add a regularization term C to our training goal that promotes a cosine similarity of 0 between $\bar{\theta}_1$ and $\bar{\theta}_2$: $C(\bar{\theta}_1, \bar{\theta}_2) = \frac{\langle \bar{\theta}_1, \bar{\theta}_2 \rangle^2}{\|\bar{\theta}_1\|^2 \|\bar{\theta}_2\|^2}$ The final optimization loss is then:

$$\mathcal{L}(\bar{\theta}_1, \bar{\theta}_2) = \mathbb{E}_{\alpha \sim \mathcal{U}[0:1]}\left[\mathbb{E}_{\tau \sim \pi_{\alpha\bar{\theta}_1 + (1-\alpha)\bar{\theta}_2}}\left[R\left(\tau\right)\right]\right] - \beta \cdot C(\bar{\theta}_1, \bar{\theta}_2) \tag{6.1}$$

where β is an hyper-parameter (see Section 6.4 for a discussion) that weights the auxiliary term.

One good property of this loss (in comparison to a method introducing an intrinsic reward) is that it does not modify the reward objective of the learned policies and thus does not encourage the model to learn policies that are sub-optimal at train time. In Section 6.4, we show that adding this auxiliary loss does not modify the performance of the model over the training environment, and does not need to be balanced by using any additional hyper-parameters.

We provide in Figure 6.2 the adapted version of the clipped PPO algorithm (see Section 2.1.1 for details about PPO) for learning a subspace of policies. In comparison to the classical approach, the batch of trajectories is first acquired by multiple policies sampled following $p(\alpha)$ (line 2-3). Then the PPO objective is optimized taking into account the policies used when sampling trajectories (line 4). At last, the critic is updated (line 5), taking as an input the α value so that it can make robust estimations of the expected reward for all the policies in the subspace. Adapting off-policy algorithms would be similar. Additional details are provided in appendix. Note that, for environments with discrete actions, we have made the same adaptation based on the A2C algorithm since A2C has less hyper-parameters than PPO and is easier to tune, with similar results.

6.2.3 K-shot adaptation

Given a subspace of policies $\bar{\Theta}$, different methods can be achieved to find the best policy over the test environment. For instance, it could be done by optimizing the distribution $p(\alpha)$ at test time. In this chapter, we use the same yet effective K-shot adaptation technique than S. Kumar et al. 2020 and Osa et al. 2021: we sample K episodes using different policies defined by different values of α that are uniformly spread over $[0, 1]$. In our example, it means that we evaluate policies uniformly distributed within the pentagon to identify a good test policy (blue star). Note that, when the environment is deterministic, only one episode per value of α needs to be executed to find the best policy, which leads to a very fast adaptation.

Note that the models proposed in (Osa et al. 2021; S. Kumar et al. 2020) shares some similarities with our approach with two differences: i) the auxiliary loss is based on an additional neural network used to enforce diversity in the behaviour of the policies. Moreover, in S. Kumar et al. 2020, this term is integrated to the reward while in (Osa et al. 2021), the auxiliary loss can be used only with continuous actions.

6.3 Experimental Protocol

6.3.1 Environments

We perform experiments in 6 different environments. Implementations based on the SaLinA (Denoyer et al. 2021) library together with train and test environments will be released upon acceptance. For each environment, we consider one train environment on which we trained the different methods, and multiple variations of the training environment for evaluation resulting in 50 test environments in total. Note that the complete experiments correspond to hundred of trained policies, and dozens of thousands of policy evaluations. For simple control environments (i.e., CartPole, Pendulum and AcroBot from the Openai Gym suite (Brockman et al. 2016)), we introduce few variations of the physics constant at test-time, for instance by varying the mass of the cart, the length of the pole. For complex control environments (i.e., HalfCheetah and Ant using the BRAX library (C Daniel Freeman et al. 2021a), we both use variations of the physics (e.g., gravity), variations of the agent shape (e.g., changing the size of the leg, or of the foot) and sensor alterations. At last, in MiniGrid (Chevalier-Boisvert et al. 2018) and ProcGen (Cobbe et al. 2019) we perform experiments where the agent is trained in one particular levels, but is evaluated in other levels (single levels on MiniGrid, and set of 10 levels in ProcGen). Note that ProcGen is a pixel-based environment where the architecture of the policy is much more complex than in control environments. Toy experiments on a simple Maze 2d are given in Figure 6.6 (left) to show the nature of the policies learned by the different methods.

6.3.2 Comparison with other methods

We compare our approach **LoP** [2] with different state-of-the-art methods: a) The **Single** approach is just a single policy learned on the train environment, and evaluated on the test ones. b) The **DIAYN+R**(reward) method is an extension of DIAYN (Eysenbach et al. 2018) where a set of discrete policies

2. We consider the LoP-A2C and the LoP-PPO models for environments with respectively discrete and continuous actions. LoP-PPO could be also used in the discrete case but requires more hyper-parameter tuning.

is learned using a weighted sum between the DIAYN reward and the task reward:

$$R_{DIAYN+R}(s,a) = r(s,a) + \beta \log p(z|s) \tag{6.2}$$

Critically, this model requires to choose a discriminator architecture to compute $\log p(z|s)$ and modifies the train reward by defining an intrinsic reward that may drastically change the behavior of the policies at train time. c) At last, we also compare with the model proposed in (Osa et al. 2021) denoted **Lc** (Latent-conditioned) that works only for continuous actions. This model is also based on a continuous z variable sampled uniformly at train time, but only uses an auxiliary loss without changing the reward. This auxiliary loss is defined through the joint learning of a density estimation model $\log P(z|s,a)$ where back-propagation is made over the action a. As in DIAYN+R, this model needs to carefully define a good neural network architecture for density estimation. Since Lc cannot be used with environment that have discrete actions, we have adapted DIAYN+R (called **DIAYN+R Cont.**) using a continuous z variable (instead of a discrete one) and a density estimation model $\log P(z|s)$ as in Osa et al. 2021. Note that we do not compare to (S. Kumar et al. 2020) for the exact same reason as the one identified in (Osa et al. 2021): SMERL assumes that the reward is known over the complete trajectories which results in unnatural adaptation of on-policy RL algorithms like PPO. Moreover, preliminary experiments with SMERL does not demonstrate any advantage against DIAYN+R correctly tuned. We also conducted experiments where K independent policies are learned, the best one being selected over each test environment. This approach obtains lower performance than the proposed baseline and needs K more training samples making it unrealistic in most of the environments.

6.3.3 Architecture and selection procedure

As network architectures, we use multi-layer perceptrons (MLP) with ReLU units for both the policy and the critic. For DIAYN+R $\log P(z|s,...)$ is also modeled by a MLP with a soft-max output. For Lc and DIAYN+R Cont., $\log P(z|s,...)$ is modeled by a MLP that computes the mean of a Gaussian distribution with a fixed variance. For these baselines, z is concatenated with the environment observation as an input for the policy and the critic models.

To choose the hyper-parameters of the different methods, let us remind that test environments cannot be used at train time for doing hyper-parameters search and/or model selection which makes this setting particularly difficult. Therefore, we rely on the following procedure: a grid-search over hyper-parameters is made, learning a single policy over the train environment. The best value of the hyper-parameters is then selected as the one that provides the best policy at train time. These hyper-parameters are then used for all the different baselines. Concerning the β value, for LoP, we report test results for $\beta = 1.0$ while, for Lc and DIAYN+R, we use the best value of β on test environments. This corresponds to an optimistic evaluation of the baseline performances; aiming at showing that our method is much more efficient since it does not need such a beta-tuning ($\beta = 1.0$ giving good performance in the majority of cases). Said otherwise, we compare our model in the less favorable case where baselines have been unrealistically tuned.

For the adaptation step, each policy is evaluated over 10 episodes for stochastic environments or 1 single episode for deterministic environments. We repeat this procedure over 10 different training seeds, and report the reward over the different test environments together with standard deviation.

6.4 Results & Analysis

6.4.1 Performance

We report the test performance of the models on different environments in Table 6.4.1. In all the environments, the adaptive models perform better than learning a single policy over the train environment which is not surprising. In most of the cases, LoP is able to achieve a better performance than other methods. For instance, on HalfCheetah where we evaluate the different methods over 16 variations of the train environments, LoP achieves an average reward of 10589 while Lc and DIAYN+R obtain respectively 9547 and 9680. Some examples of the discovered that behaviors in Ant and HalfCheetah [3] for the different methods, and for different values of α are

3. Videos available here

(a) environments with discrete action space

	CartPole	Acrobot	Pendulum	Minigrid	toy maze
Nb. Test Env.	6	6	3	6	4
Single Policy	143.4	-99.7	-52.7	0.169	-83.2
LoP	149.9	**-93.2**	**-28.9**	**0.447**	**-33.6**
DIAYN+R	**168.1**	-97.0	-47.1	0.248	-42.2
DIAYN+R L_2	156.1	-93.6	-44.0	0.443	-

(b) environments with continuous action space

	Brax HalfCheetah	Brax Ant
Nb. Test Env.	16	15
Single Policy	7697	3338
LoP	**10589**	**4031**
DIAYN+R	9680	3759
Lc	9547	4020

Table 6.1 – Average cumulated reward of the different models over multiple testing environments, averaged over 10 training seeds (higher is better). For DIAYN and Lc, results are reported for testing 10 policies for LoP, Lc, and DIAYN+R using 10 episodes per policy for stochastic environments and 1 episode per policy on deterministic ones. Performance is evaluated using the deterministic policy.

illustrated in Figures 6.8 and 6.10. This outlines that learning models that are optimal on the train task reward, but with different parameter values, allows us to discover policies react differently to variations of the training environment. It seems to be a better approach than encouraging policies to have a different behaviors (i.e., generating different state distributions) at train time. Same conclusions can be drawn in most of the environments, including MiniGrid where LoP is able to explore large mazes while being trained only on small ones. On the ProcGen environment, where the observation is an image processed through a complex ConvNet architecture, enforcing functional diversity (DIAYN+R) does not allow to learn good policies while the LoP model is able to better generalize to unseen levels. Note that the performance at train time is the same for all the different approaches reported (see the Table 6.3 for instance) but quickly decreases in DIAYN for larger values of β while it stays stable for LoP where the best results are obtained for $\beta = 1.0$.

Interestingly, in CartPole, DIAYN+R performs quite well. Indeed, when analyzing the learned policies, it seems to be a specific case where it is possible to obtain optimal policies that are diverse w.r.t the states they are sampling (by moving the cart more or less on the right/left while maintaining the pole vertical).

We have also performed experiments where test environments have the same dynamics as the training environment, but with defective sensors (i.e., some features at test time have a null value). The fact that LoP behaves also well confirms the effectiveness of our approach to different types of variations, including noisy features on which baselines methods were not applied in previous publications.

6.4.2 Sensitivity to hyper-parameters

	$K=$	5	10	20	Train Perf.
LoP	$\beta = 0.1$	3905	3991	4164	7659
	$\beta = 1.$	**4035**	**4031**	**4174**	7630
	$\beta = 10$	3998	4012	4145	7670
DIAYN	$\beta = 0.1$	3558	3833	3949	7739
	$\beta = 1.$	3451	3759	2878	5388
	$\beta = 10$	3356	3400	3109	4430
Lc	$\beta = 0.1$	3909	4020	4150	7767
	$\beta = 1.$	3820	3947	4126	7650
	$\beta = 10$	3870	3945	4108	7710

Figure 6.3 – Performance of the models **at train time** that shows that for LoP β is not hurting train performance while it is DIAYN+R. Standard error deviation is reported in Appendix Table 7 for each environment. We also report the performace at train time that shows that a too high value of β hurts DIAYN+R performance while is less critical in LoP.

One important characteristic of LoP is that it can be used with $\beta = 1.0$ and does not need to define any classifier architecture as opposed to DIAYN+R and Lc. Indeed, as shown in Table 6.3, the training performance of DIAYN drastically depends on a good tuning of β. Lc, which is less sensible, needs to use a correct classifier architecture as in DIAYN. LoP is simple to tune since the cosine term is usually easy to satisfy and our approach, at convergence, always reaches a 0 value on this term when $\beta > 0.0$. It

is also interesting to note than, on the BRAX environments, the number of environment interactions needed to train LoP is similar than the one needed to train a single policy and LoP comes with a very small overhead in comparison to classical methods.

6.4.3 Sensitivity to the budget

			SmallFeet	TinyFriction	BigGravity
A	LoP	K=5	8283	**10425**	**10464**
		K=10	8805	**10662**	**10578**
		K=20	8794	**10734**	**10807**
	DIAYN+R	K=5	7580	9132	8989
		K=10	7580	9132	8989
		K=20	8255	10003	9766
	Lc	K=5	8186	9521	9360
		K=10	8186	9661	9488
		K=20	8107	9661	9506
	BoP (N=3)	K=5	6775	7867	7878
		K=10	6660	7840	8026
		K=20	6996	7963	8015
	CoP (N=3)	K=5	**8996**	9468	9287
		K=10	**9210**	9523	9568
		K=20	**9155**	9979	9695

Figure 6.4 – Ablation study on the number of policies K used at test time on 3 HalfCheetah environment variations together with the performance of the BoP and CoP variants (see Section 6.4.6).

One interesting property is the number of policies (and thus of episodes) to test over a new environment to get a good performance. For LoP and Lc, given a trained model, one can evaluate as many policies (i.e., different values of α) as desired. For DIAYN+R, testing more policies also means training more policies which is expensive and less flexible. Table 6.4 (right) provides the reward of the different methods when testing K policies on different HalfCheetah settings: as expected, the performance of DIAYN+R tends to decrease when K is large since the model has difficulties to learn too many diverse policies. For LoP and Lc, spending more episodes to evaluate more policies naturally leads to a better performance: these two models provide a better way to deal with the exploration-exploitation trade-off at

test time. Lc also needs to define an additional neural network architecture to model $\log P(\alpha|s,a)$ while LoP does not, making our approach simpler.

6.4.4 Functional diversity

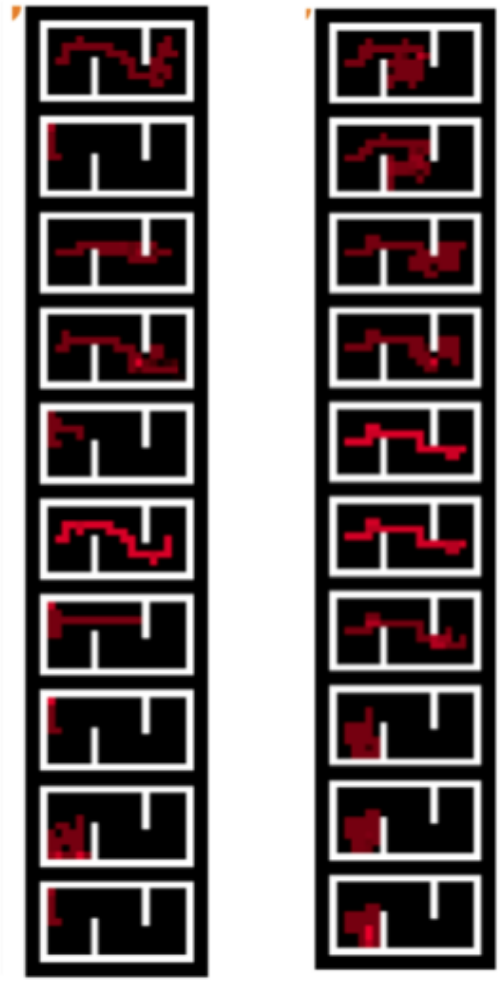


Figure 6.5 – Trajectories generated by $K = 10$ policies on an unseen maze (objective is to go from left to right) for DIAYN+R (left column with best β value) and LoP (right column with $\beta = 1.0$), illustrating the diversity obtained with DIAYN+R and LoP.

To better understand the nature of the policies discovered by the different approaches, we have made a qualitative study in which we analyze i) the robustness of the methods to corrupted observations, ii) the functional diversity induced by the different models, and iii) the specificity of the

different learned policies to particular test environments. First, LoP is more robust to input feature corruption (see Appendix Table 7 for the results) and we conjecture that it is because the diversity in the parameter space allows this model to learn policies that does not take into account the same input features equally.

We also measure the functional diversity induced by the different models by training *a posteriori* a classifier that aims at recovering which policy (i.e which value of α) has generated particular trajectories (see Figure 6.6. On LoP with $K = 5$, such a classifier obtains a 82% accuracy at validation time showing that the 5 policies are quite diverse, but less than the DIAYN+R policies where the classifier reaches a 100% accuracy which is logical knowing the auxiliary loss introduced by DIAYN which enforces this type of diversity. It is interesting to note that with the trajectories generated in the test environments with LoP policies, the accuracy of the classifier is reaching 87 %: when LoP is facing new environments, it tends to generate more diverse policies. We think that it is due to the fact that, since the policies have different parameter values, they react differently to states that have not been encountered at train time. At last, examples of policies on a simple maze2d are given in Figure 6.5 which illustrate the diversity of the discovered policies.

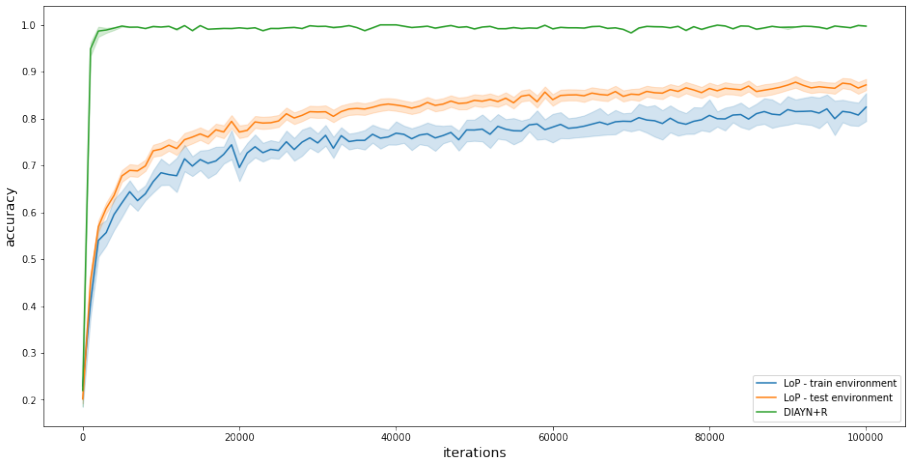


Figure 6.6 – We trained small discriminators over a dataset (100,000 environment interactions) of trajectories obtained with the learned policies of LoP and DIAYN+R when K=5. For each environment, for each seed, we trained a single discriminator and averaged the results. While the discriminators trained on DIAYN+R reach 100% accuracy rapidly on both train and test environments, they learn slower for LoP, with a slight advantage for the test environment, validating the fact that the diversity induced by the cosine similarity on the weights is more visible in variations of the environment rather than the environment on which the model has been trained. We evaluated the discriminator on a validation dataset (also 100,000 environment interactions) resulting in 100% accuracy for DIAYN in both train and test environments. For LoP, we obtained 82% accuracy on the training environment, and 87% on the test environments. The discriminator architecture consists in a neural network of two hidden layers of size 16, taking the unprocessed states as an input and outputting the predicted policy used (like in DIAYN).

6.4.5 Policies used at test time

We wanted to examine whether one policy within the subspace outperforms the others at test time. Indeed, this is something that has been analyzed by Wortsman et al. 2021 in supervised learning, with clearer evidence that the midpoint always outperforms the other points in the line segment. To do so, we plotted histograms illustrating which policy is used on each test environment with LoP and DIAYN+R (see Figure 6.7). This analysis shows that a) anchor policies are little used in the test environments and b) different test environments make use of different policies (i.e., different z). We hypothesize that it is a border effect of the training objective we

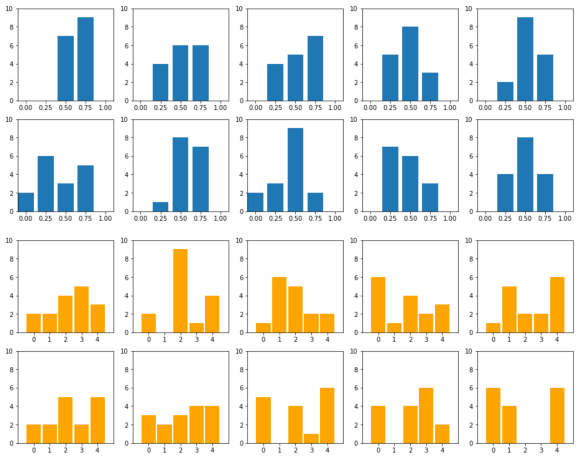


Figure 6.7 – Number of times each policy (over $K = 5$ tested policies) is chosen over the 16 test environments of the HalfCheetah setting for each of the 10 seeds. Blue is LoP and orange is DIAYN+R. Different policies are used for different test environments showing the interest of learning a subspace of policies. Note that in LoP, the anchor policies are rarely chosen.

defined : as we aim to optimize the whole line segment at train time (see Algorithm 6.2), by randomly sampling a z value at each epoch, the middle of the line segment should benefit from both anchor policies updates while these anchor policies benefit from the update of themselves.

6.4.6 Beyond a Line of Policies

While LoP is based on the learning of $N = 2$ anchor parameters, it is possible to combine more than two anchor parameters. We study two approaches combining $N = 3$ anchor parameters (that can be extended to $N = 3$): a) the first approach is a convex combination of policies (CoP) where α is sampled following a Dirichlet distribution. (b) The second approach is a Bézier combination (BoP). The only change between LoP and these models resides in the way we combine the anchor policies. For CoP, it is just the generalization of LoP for $N > 2$. BoP, makes use of a Bézier parametric curve (see Section 5.4.3) Concerning the policies α evaluated at test time, BoP uses the same strategy as LoP by testing values that are uniformly distributed in $[0; 1]$. For CoP, we opted for sampling K policies using a Dirichlet distribution over $[0, 1]^3$.

The results are presented in Table 6.4 over multiple HalfCheetah environments. It can be seen that these two strategies are not so efficient. LoP is thus a good trade-off between the number of parameters to train and the performance (Note that BoP and CoP need more samples to converge), at least given the particular neural network architectures we have used in this paper. We also performed an in-depth analysis of the evolution of the reward when K is increasing for LoP and CoP in Halfcheetah test environment (Figure 6.9). While we expected CoP to outperform LoP when K is high, the best reward becomes stable when K=20 for both methods, and in most test environments, CoP is not able to reach the same best reward as LoP.

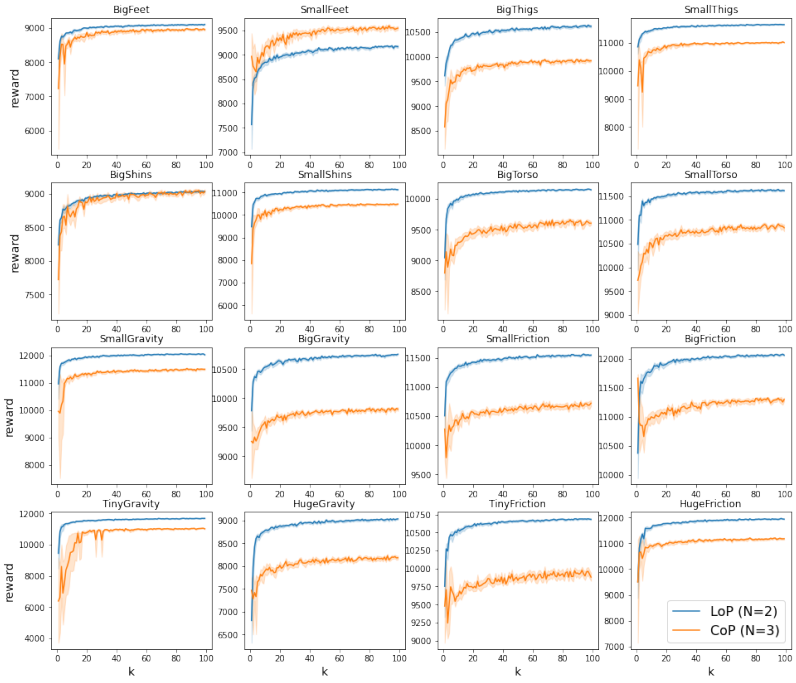


Figure 6.9 – Evolution of the best reward obtained with respect to K for LoP (N=2) and CoP (N=3) for each Halfcheetah test environment. We ran the K-shot evaluation for each K from K=1 to K=100 using the method described above : we simply sample K random coefficients using the uniform distribution over $[0, 1]$ for LoP and the Dirichlet distribution over $[0, 1]^3$ for CoP. Results are averaged over 10 run for each K, and over the 10 models we learned for each method.

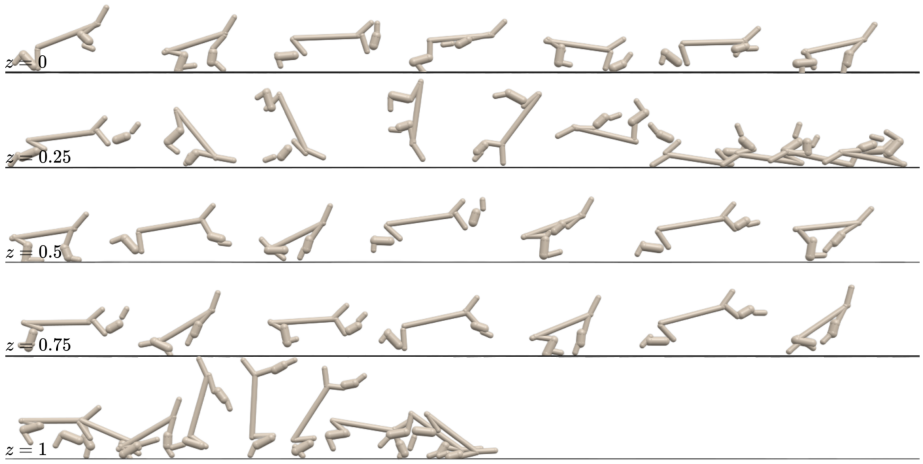


Figure 6.8 – Qualitative example of LoP trajectories on HalfCheetah "BigShins" test environment (5-shot setting). The best reward is obtained for $\alpha = 0.75$.

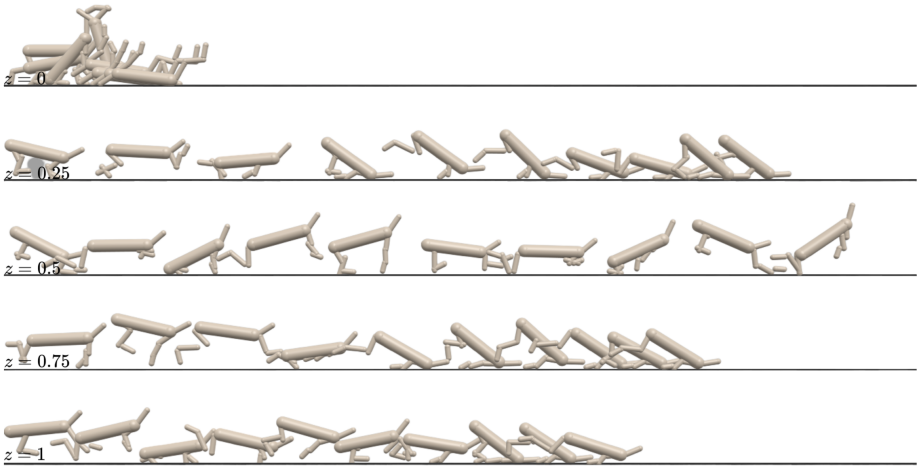


Figure 6.10 – Extreme case: when torso radius and mass are increased by 50%. Only one policy is able to adapt without falling down ($\alpha = 0.5$).

6.5 Conclusion

The approach we have presented in this chapter is deceptively straightforward, yet it yields policies resilient to the fluctuations of training environments. Unlike other methods, the Line of Policies (LoP) avoids the need for fine-tuning or the introduction of complex architectures to manage diversity. However, this is not without drawbacks:

- Efforts to create a subspace with a higher number of anchors (specifically, CoP and BoP incorporating three policies) did not bear fruit. These approaches often underperformed compared to LoP, despite representing a 50% increase in the number of parameters.
- The diversity achieved through LoP is recognized as finite, as illustrated in Figure 6.6. Even when augmented with a regularization term like cosine similarity, the optimization targets a singular reward, which inherently limits diversity. The method shows limited diversity of skills in practice.

Addressing these issues points us towards expanding this suite of strategies into the domain of Continual Reinforcement Learning. Such exploration would lead to a deeper understanding of the constructed subspace

and the investigation of different auxiliary losses to better control the subspace's shape. These prospects will be the focus of the next chapter.

CHAPTER 7

BUILDING A SUBSPACE OF POLICIES

We now advance the discussion of policy robustness and diversity from Chapter 6 into the realm of CRL, guided by insights from our paper *Building a Subspace of Policies for Continual Learning*[1] (Gaya et al. 2023) (ICLR Spotlight). The ability to continuously acquire new knowledge and skills is crucial for autonomous agents. As we saw in Section 2.4.3, existing methods are typically based on either fixed-size models that struggle to learn a large number of diverse behaviors, or growing-size models that scale poorly with the number of tasks. In this chapter, we aim to strike a better balance between an agent's *size* and *performance* by designing a method that grows adaptively depending on the task sequence. We introduce Continual Subspace of Policies (CSP), a new approach that incrementally builds a subspace of policies for training a reinforcement learning agent on a sequence of tasks. The subspace's high expressivity allows CSP to perform well for many different tasks while growing sublinearly with the number of tasks. Our method does not suffer from forgetting and displays positive transfer to new tasks. CSP outperforms a number of popular baselines on a wide range of scenarios from two challenging domains, Brax (locomotion) and Continual World (manipulation). Interactive visualizations of the subspace can be found at csp.

Drawing on the foundational concepts introduced in Chapter 5 and the first experiments in Chapter 6, we will first transition from the setting of few-shot adaptation to a more ambitious CRL framework (Section 7.1). We will then delve into our method, which entails an iterative and adaptive construction of a subspace of policies (Section 7.2). A significant portion of our discussion will be dedicated to the experimental protocol and the design of engaging scenarios that are conducive to comparing our methods (Section 7.3). We will dissect both the quantitative and qualitative outcomes

1. You can find the appendix in the arxiv paper here. Code is available here.

of our experiments (Section 7.4). The final Section 7.5 offers our concluding remarks on the study.

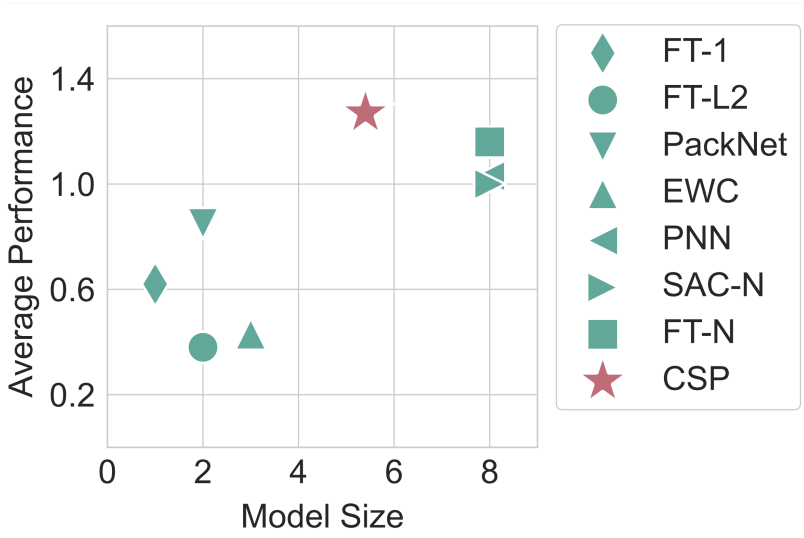


Figure 7.1 – Trade-off between model performance and size for a number of methods, on a sequence of 8 tasks from HalfCheetah (see Appendix Figure 6 for trade-offs on other scenarios)

7.1 Context & Motivations

Developing autonomous agents that can continuously acquire new knowledge and skills is a major challenge in machine learning, with broad application in fields like robotics or dialogue systems. In the past few years, there has been growing interest in the problem of training agents on sequences of tasks. As we saw in Section 2.4, current methods either use *fixed-size* models that struggle to learn a large number of diverse behaviors (Hinton et al. 2006; Rusu et al. 2016a; Z. Li and Hoiem 2018; Bengio and LeCun 2007; Kaplanis et al. 2019; Traoré et al. 2019; Kirkpatrick et al. 2017; Schwarz et al. 2018; Mallya and Lazebnik 2018), or *growing-size* models that scale poorly with the number of tasks (Berseth et al. 2018; Cheung et al. 2019; Wortsman et al. 2020). In this Chapter, we introduce an *adaptive-size* model which strikes a better balance between **performance** and **size**, two crucial properties of continual learning systems (Veniat et al. 2020), thus scaling better to long task sequences.

Taking inspiration from the mode connectivity literature (Chapter 5), we propose **Continual Subspace of Policies** (**CSP**), a new CRL approach that incrementally builds a subspace of policies. Instead of learning a single policy, CSP maintains an entire subspace of policies defined as a convex hull in parameter space. The vertices of this convex hull are called anchors,

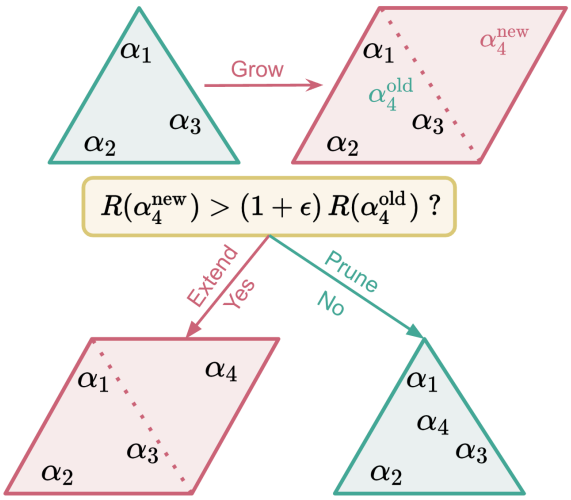


Figure 7.2 – Illustration of CSP.

with each anchor representing the parameters of a policy. This subspace captures a large number of diverse behaviors, enabling good performance on a wide range of settings. At every stage of the CRL process, the best found policy for a previously seen task is represented as a single point in the current subspace (*i.e.* unique convex combination of the anchors), which facilitates cheap storage and easy retrieval of prior solutions. If a new task shares some similarities with previously seen ones, a good policy can often be found in the current subspace without increasing the number of parameters. On the other hand, if a new task is very different from previously seen ones, CSP extends the current subspace by adding another anchor, and learns a new policy in the extended subspace. In this case, the pool of candidate solutions in the subspace increases, allowing CSP to deal with more diverse tasks in the future. The size of the subspace is increased only if this leads to performance gains larger than a given threshold, allowing users to specify the desired trade-off between performance and size (*i.e.* number of parameters or memory cost).

Ultimately, CSP iteratively learns a subspace of policies in the continual RL setting. Figure 7.2 illustrates our method: (1) At every stage during training, the subspace is a simplex defined by a set of anchors,*i.e.* vertices (top right . Any policy (*i.e.* point) in this simplex can be represented as a convex combination α of the anchor parameters.α_i defines the best policy in the subspace for task i. (2) When the agent encounters a new task, CSP tentatively *grows* the subspace by adding a new anchor. (3) Bottom-left: If the new task i is very different from previously seen ones, a better policy

α_i^{new} can usually be learned in the new subspace. In this case, CSP *extends* the subspace by keeping the new anchor. (4) If the new task bears some similarities to previously seen ones, a good policy α_i^{old} can typically be found in the old subspace. In this case, CSP *prunes* the subspace by removing the new anchor. The subspace is extended only if it improves performance relative to the old subspace by at least some threshold ϵ.

7.2 Continual Subspace of Policies (CSP)

Our work builds on Chapter 6 by leveraging a subspace of policies. However, instead of using the subspace to train on a single environment and adapt to new ones at test time, we use it to efficiently learn tasks sequentially in the continual RL setting. This requires designing a new approach for learning the subspace, as detailed below. We propose to adaptively construct a subspace of policies for the continual RL setting in which an agent learns tasks sequentially. We call our method **Continual Subspace of Policies** (**CSP**). A pseudo-code is available in Figure 7.3.

Our model builds a sequence of subspaces $\Theta_1, ..., \Theta_N$, one new subspace after each training task, with each subspace extending the previous one. Note that each subspace Θ_j is a collection of at most j anchors (or neural networks) *i.e.* $|\Theta_j| \leq j, \ \forall\, j \in 1...N$ since the subspace grows sublinearly with the number of tasks (as explained below). Hence, the number of anchors m of a subspace Θ_j is not the same as the number of tasks j used to create Θ_j. The learned policies are represented as single points in these subspaces. At each stage, CSP maintains both a set of anchors defining the current subspace, as well as the weights α corresponding to the best policies found for all prior tasks. The best found policy for task i after training on the first j tasks, $\forall\, i \leq j$, is denoted as $\pi_i^j(a|s)$ and can be represented as a point in the subspace Θ_j with a weight vector denoted α_i^j such that $\pi_i^j(a|s) = \pi(a|s, [\alpha_i^j, \Theta_j])$, where $|\alpha_i^j| = |\Theta_j| \leq j$.

Given a set of anchors Θ_j and a set of previously learned policies $\{\alpha_1^j, ..., \alpha_j^j\}$, updating our model on the new task t_{j+1} produces a new subspace Θ_{j+1} and a new set of weights $\{\alpha_1^{j+1}, ..., \alpha_{j+1}^{j+1}\}$. There are two possible cases. One possibility is that the current subspace already contains a good policy for t_{j+1}, so we just need to find the weight vector corresponding to a policy which performs well on t_{j+1}. The other possibility is that the current

subspace does not contain a good policy for t_{j+1}. In this case, the algorithm produces a new subspace by adding one anchor to the previous one (see next section), and converts the previous policies to be compatible with this new subspace.

To achieve this, CSP operates in two phases:

1. *Grow* the current subspace by adding a new anchor and learning the best possible policy for task $j+1$ in this subspace (where the previous j anchors are frozen).
2. Compare the quality of this policy with the best possible policy expressed in the previous subspace. Based on this comparison, decide whether to *extend* the subspace to match the new one or *prune* it back to the previous one.

We now describe in detail the two phases of the learning process, namely how we grow the subspace and how we decide whether to extend or prune it (see Figure 7.3 for a pseudo-code).

7.2.1 Grow the Subspace

Given the current subspace Θ_j composed of $m \leq j$ anchors (with j being the number of tasks seen so far), a new subspace $\tilde{\Theta}_{j+1}$ is built as follows. First a new anchor denoted θ_{j+1} is added to the set of anchors such that $\tilde{\Theta}_{j+1} = \Theta_j \bigcup \{\theta_{j+1}\}$. With all previous anchors frozen, we train the new anchor by sampling α values from a Dirichlet distribution parameterized by a vector with size $j+1$, $Dir\big(\mathcal{U}(j+1)\big)$. The new anchor θ_{j+1} is updated by maximizing the expected return obtained by interacting with the new task $j+1$ using policies defined by the sampled α's:

$$\theta_{j+1} = \arg\max_{\theta} \mathbb{E}_{\alpha \sim Dir, \tau \sim \pi(a|s,[\alpha, \Theta_j \bigcup \{\theta\}])} \left[R_{j+1}(\tau) \right] \tag{7.1}$$

where $R_{j+1}(\tau)$ is the return obtained on task $j+1$ throughout trajectory τ which was generated using policy $\pi(a|s, [\alpha, \tilde{\Theta}_{j+1})]$. Note that the anchor is trained such that not only one but all possible values of α tend to produce a good policy. To do so, we sample different α per episode. The resulting subspace thus aims at containing as many good policies as possible for task $j+1$.

Algorithm 7.1 Continual Subspace of Policies (CSP)

Input: $\theta_1,\dots,\theta_j$ (previous anchors), ϵ (threshold)
Initialize: W_ϕ (subspace critic), $\mathcal{B}$ (replay buffer)
Initialize: $\theta_{j+1} \leftarrow \frac{1}{j}\sum_{i=1}^{j}\theta_i$ (new anchor) ; `// Grow the Subspace`
for $i = 1, ..., B$ **do**
  Sample $\alpha \sim Dir\big(\mathcal{U}(j+1)\big)$
  Set policy parameters $\theta_\alpha \leftarrow \sum_{i=1}^{j+1}\alpha_i\theta_i$
  for $l = 1, ..., K$ **do**
    Collect and store (s, a, r, s', α) in $\mathcal{B}$ by sampling $a \sim \pi_{\theta_\alpha}(\cdot|s)$
  end
  if *time to update* **then**
    Update $\pi_{\theta_{j+1}}$ and W_ϕ using the SAC algorithm and the replay buffer $\mathcal{B}$
  end
end
Use $\mathcal{B}$ and W_ϕ to estimate: ; `// Extend or Prune the Subspace`

$$\alpha^{\text{old}} \leftarrow \underset{(\alpha,0) \text{ with } \alpha\in\mathbb{R}_+^m, \|\alpha\|_1=1}{\arg\max} W_\phi(\alpha)$$

$$\alpha^{\text{new}} \leftarrow \underset{\alpha\in\mathbb{R}_+^{m+1}, \|\alpha\|_1=1}{\arg\max} W_\phi(\alpha)$$

if $W_\phi(\cdot, \alpha^{\text{new}}) > (1+\epsilon)\cdot W_\phi(\cdot, \alpha^{\text{old}})$ **then**
  Return: $\theta_1, \dots, \theta_j, \theta_{j+1}, \alpha^{\text{new}}$; `// Extend`
else
  Return: $\theta_1, \dots, \theta_j, \alpha^{\text{old}}$; `// Prune`
end

Figure 7.3 – Detailed algorithm of csp

7.2.2 Extend or Prune the Subspace

To decide if the new anchor is kept, we propose to simply compare the best possible policy for task $j+1$ in the new subspace with the best possible policy for the same task in the previous subspace (*i.e.* without using the new anchor). Each policy could be evaluated via Monte-Carlo (MC) estimates by doing additional rollouts in the environment and recording the average performance. However, this typically requires a large number (*e.g.* millions) of interactions which may be impractical with a limited budget. Thus, we propose an alternative procedure to make this evaluation sample efficient.

For each task j and corresponding subspace Θ_j, our algorithm also learns a Q-function $Q(s, a, \alpha)$ which is trained to predict the expected return on task j for all possible states, actions, and all possible α's in the corresponding subspace. This Q-function is slightly different from the classical ones in RL since it takes as an additional input the vector α. This Q-function is reset for every new task. Our algorithm is based on SAC (Haarnoja et al. 2018a), so for each task, we collect a replay buffer of interactions $\mathcal{B}$ which contains all states and actions seen by the agent while training on that task. Thus, the Q-function $Q(s, a, \alpha)$ can help us directly estimate the quality $W(\alpha)$ of the policy represented by the weight vector α in the new subspace which can be computed as the average over all states and actions in the replay buffer:

$$W(\alpha) = \mathbb{E}_{s,a\sim\mathcal{B}} Q(s, a, \alpha). \tag{7.2}$$

It is thus possible to compute the value of α corresponding to the best policy in the extended subspace (denoted $\alpha^{\text{new}} \in \mathbb{R}_+^{m+1}, \|\alpha^{\text{new}}\|_1 = 1$):

$$\alpha^{\text{new}} = \underset{\alpha\in\mathbb{R}_+^{m+1},\|\alpha\|_1=1}{\arg\max} W(\alpha), \tag{7.3}$$

as well as the value of α corresponding to the best policy in the previous subspace (denoted $\alpha^{\text{old}} \in \mathbb{R}_+^{m}, \|\alpha^{\text{old}}\|_1 = 1$):

$$\alpha^{\text{old}} = \underset{(\alpha,0) \text{ with } \alpha\in\mathbb{R}_+^{m},\|\alpha\|_1=1}{\arg\max} W(\alpha). \tag{7.4}$$

In practice, α^{new} and α^{old} are estimated by uniformly sampling a number of α's in the corresponding subspace as well as a number of states and actions from the buffer.

The quality of the new subspace and the previous one can thus be evaluated by comparing $W(\alpha^{\text{new}})$ and $W(\alpha^{\text{old}})$. If $W(\alpha^{\text{new}}) > (1 + \epsilon) \cdot W(\alpha^{\text{old}})$, the subspace is extended to the new subspace (*i.e.* the one after the grow phase): $\Theta_{j+1} = \tilde{\Theta}_{j+1}$. Otherwise, the subspace is pruned back to the old subspace (*i.e.* the one before the growth phase): $\Theta_{j+1} = \Theta_j$. Note that, if the subspace is extended, the previously learned policies have to be mapped in the new subspace such that $\alpha_i^{j+1} \in \mathbb{R}_+^{j+1}$ *i.e.* $\forall\ i \leq j,\ \alpha_i^{j+1} := (\alpha_i^j, 0)$ and $\alpha_{j+1}^{j+1} := \alpha^{\text{new}}$. If the subspace is not extended, then old values can be kept *i.e.* $\forall\ i \leq j,\ \alpha_i^{j+1} := \alpha_i^j$ and $\alpha_{j+1}^{j+1} := \alpha^{\text{old}}$.

After finding the best α for the current task, the replay buffer and Q-function are reinitialized. Hence, the memory cost for training the Q-function is constant and there is no need to store data from previous tasks unlike other CRL approaches (Rolnick et al. 2019). The policy $\pi_{\theta_{j+1}}$ and value function W_ϕ are updated using SAC (Haarnoja et al. 2018a). See Appendix C for more details.

7.2.3 Training and using the critic

The subspace critic W_ϕ plays a central role in our method. Compared to the vanilla SAC critic, we only add the convex combination α as an input (concatenated with the states and actions). In this way, it is optimized not only to evaluate the future averaged return on (s, a) pairs of a single policy, but an infinity of policies, characterized by the convex combination α.

At the end of each task, the subspace critic has the difficult task to estimate by how much the new anchor policy θ_{j+1} improves the performance of the current subspace Θ_j. To do so, one has to find the best combination of policies in the last subspace Θ_j, calling it α^{old} and the one in the current subspace Θ_{j+1}, calling it α^{new}. In practice, we found that sampling 1024 random (s, a) pairs from the replay buffer at the end of the task allows to have an accurate estimation of the best policies. This part does not require any new interaction with the environment.

However, we found that rolling out the top-k α in the new and former subspaces helps to find the best combination. In practice, we found that setting $k := 8$ with one rollout per combination is sufficient. In tasks that have $1M$ interactions and an episode horizon of 1000, it requires to allocate 0.8% of the budget to the purpose of finding a good policy in the subspace, which does not significantly impact the training procedure.

7.2.4 Sampling Policies from the Subspace

Yet, it is important to allow the critic to estimate α^{old}. During training, we noticed that sampling with a simple flat Dirichlet distribution (i.e. a uniform distribution over the simplex induced by the current subspace) is not enough to make the critic able to accurately estimate the performance

of the last subspace (indeed, the chances of sampling a policy in the last subspace are almost surely 0.). This is why we decided to sample in both the current and the last subspace. The distribution we use is then a mixture of two Dirichlet (equal chances of sampling in the last subspace and in the current subspace). We did not perform an ablation to see if balancing the mixture would increase performances.

We also tried to sample with a peaked distribution (concentration of the Dirichlet equal to the inverse of the number of the anchors) to see if it increased performances. In some cases the new subspace is able to find good policies faster with this distribution. It can be a good trade off between always choosing the last anchor and sampling uniformly.

7.2.5 Implementation

Our Pytorch implementation of CSP uses the `nn.ModuleList` object to store anchor networks. The additional computational cost compared to a single network is negligible during both training and inference as it is mentioned in Wortsman et al. (2021).

7.2.6 Scalability

By having access to an infinite number of policies, the subspace is highly expressive so it can capture a wide range of diverse behaviors. This enables positive transfer to many new tasks without the need for training additional parameters. As a consequence, the number of parameters scales *sublinearly* with the number of tasks. The speed of growth is controlled by the threshold ϵ which defines how much performance we are willing to give up for decreasing the number of parameters (by the size of one policy network). Practitioners can set the threshold to trade-off performance gains for memory efficiency (*i.e.* the higher the ϵ the more performance losses are tolerated to reduce memory costs). In practice, we noticed that setting $\epsilon = 0.1$ allows good performance and a limited growth of parameters.

As the agent learns to solve more and more tasks, we expect the subspace to grow more slowly (or stop growing entirely) since it already contains many useful behaviors which *transfer* to new tasks. On the other hand, if

the agent encounters a task that is significantly different than previous ones and all other behaviors in the subspace, the subspace still has the *flexibility* to grow and incorporate entirely new skills. The number of anchors in the final subspace is adaptive and depends on the sequence of tasks. The longer and more diverse the task sequence, the more anchors are needed. This property is important for real-world applications with open-ended interactions where it's unlikely to know a priori how much capacity is required to express all useful skills.

7.3 Experimental Protocol

7.3.1 Designing scenarios

We evaluate CSP on 18 CRL scenarios containing 35 different RL tasks, from two continuous control domains, *locomotion* in **Brax** (C Daniel Freeman et al. 2021a) and *robotic manipulation* in **Continual World** (CW, Wołczyk et al. (2021)), a challenging CRL benchmark. For CW, we run experiments on both the proposed sequence of 10 tasks (CW10), and on all 8 triplet sequences (CW3). Each task in these sequences has a *different reward* function. The goal of these experiments is to compare our approach with popular CRL methods on a well-established benchmark.

For Brax, we create new CRL scenarios based on 3 subdomains: **HalfCheetah**, **Ant**, and **Humanoid**. Each scenario has 8 tasks and each task has *specific dynamics*. We use a budget of 1M interactions for each task. The goal of these experiments is to perform an in-depth study to separately evaluate capabilities specific to CRL agents such as *forgetting, transfer, robustness, and compositionality* (Tables 7.5a, 7.5b for more information). For each of the 4 capabilities, we create 2 CRL scenarios, one based on HalfCheetah and one based on Ant. To further probe the effectiveness of our approach, we also create one CRL scenario with 4 tasks on the challenging Humanoid domain. Here we use a budget of 2M interactions for each task.

The CRL scenarios are created following the protocol introduced in Wołczyk et al. (2021) which proposes a systematic way of generating task sequences that test CRL agents along particular axes. We studied the relationship between these changes as follows: we learn a new task with a

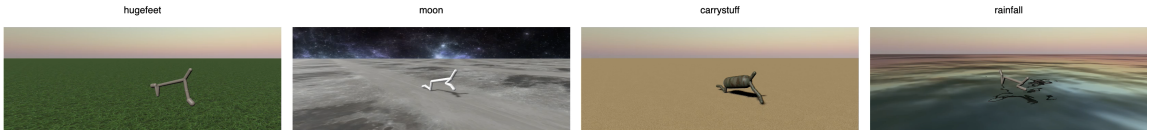


Figure 7.4 – Rendering example of our Halfcheetah forgetting scenario. Click here for interactive visualizations.

policy that has been pre-trained on a former task. We drew forgetting and transfer tables for each pair of tasks (see Figure 7.5b). With this information, we designed 5 types of scenarios representing a particular challenge in continual learning. Note that all tasks have a budget of $1M$ interactions each and repeat their loop 2 times (i.e. $8M$ interactions in total) except for the Humanoid scenario that contains 4 tasks with $2M$ interactions. While Wołczyk et al. (2021) focus on transfer and forgetting, we also probe robustness (*e.g.* adapting to environment perturbations such as action reversal), and compositionality (*e.g.* combining two previously learned skills to solve a new task). Each task in a sequence has different dynamics which are grounded in quasi-realistic situations such as increased or decreased gravity, friction, or limb lengths. here are the 5 types of scenarios [2] we designed, each representing a particular challenge in continual learning:

1. **Forgetting Scenarios** are designed such that a single policy tends to forget the former task when learning a new one.

   - Halfcheetah: `hugefeet → moon → carrystuff → rainfall`
   - Ant: `normal → hugefeet → rainfall → moon`

2. **Transfer Scenarios** are designed such that a single policy has more difficulties to learn a new task after having learned the former one, rather than learning it from scratch.

   - Halfcheetah: `carrystuff_hugegravity → moon → defectivesensor → hugefeet_rainfall`
   - Ant: `nofeet_1_3 → nofeet_2_4 → nofeet_1_2 → nofeet_3_4`

2. Note that all tasks have a budget of 1M interactions each and repeat their loop 2 times (i.e. 8M interactions in total) except for the Humanoid scenario that contains 4 tasks with 2M interactions

3. **Robustness Scenarios** alternate between a normal task and a very different distraction task that disturbs the whole learning process of a single policy (we simply inverted the actions). While this challenge looks particularly simple from a human perspective (a simple -1 vector applied on the output is fine to find an optimal policy in a continual setting), we figured out that the Fine-tuning policies struggle to recover good performances (the final average reward actually decreases).

   - Halfcheetah: `normal → inverted_actions → normal → inverted_actions`
   - Ant: `normal → inverted_actions → normal → inverted_actions`

4. **Compositional Scenarios** present two first tasks that will be useful to learn the last one, but a very different distraction task is put at the third place to disturb this forward transfer. The last task is indeed a combination of the two first tasks in the sense that it combines their particularities. For example, if the first task is "moon" and the second one is tinyfeet, the last one will combine moon's gravity and feet morphological changes.

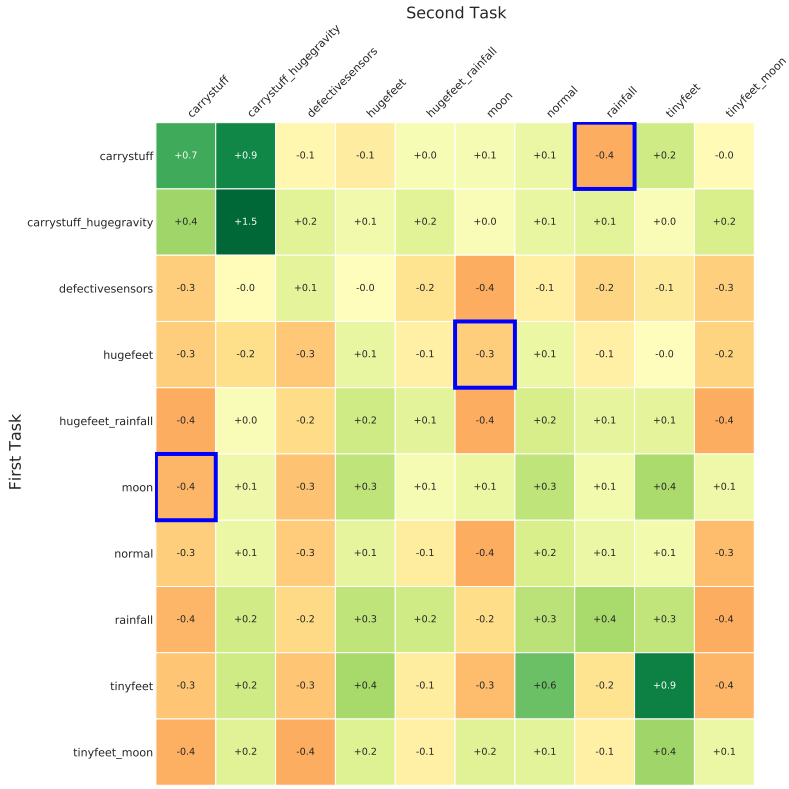
   - Halfcheetah: `tinyfeet → moon → carrystuff_hugegravity → tinyfeet_moon`
   - Ant: `nofeet_2_3_4 → nofeet_1_3_4 → nofeet_1_2 → nofeet_3_4`

5. **Humanoid scenario** is an additional scenario built with the challenging environment Humanoid to test our method in higher dimensions. Here is the detailed sequence of the scenario.

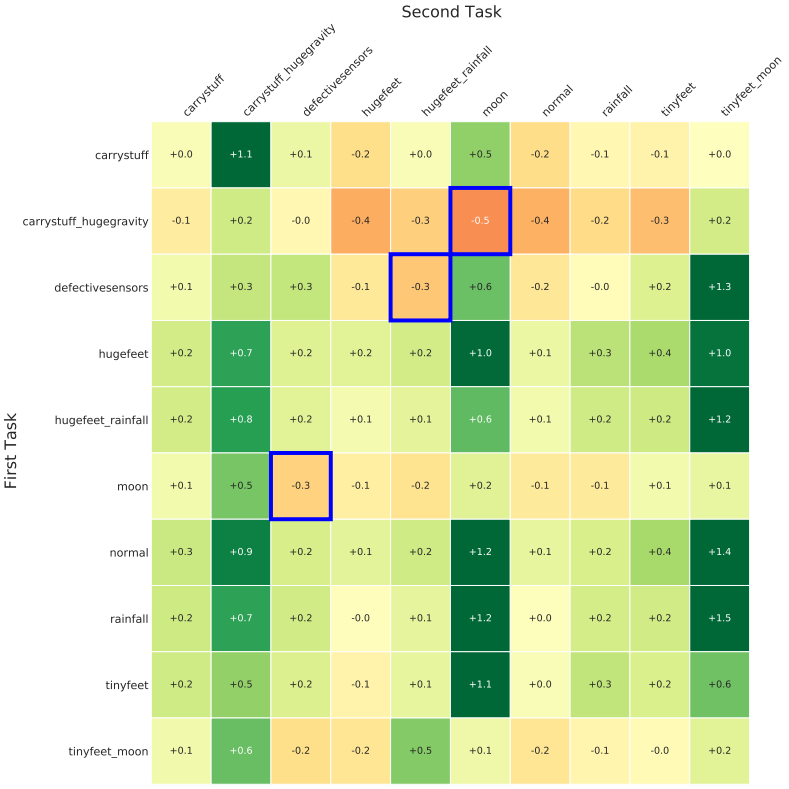
   - Humanoid: `normal → moon → carrystuff → tinyfeet`

7.3.2 Comparisons

We compare CSP with a number of popular CRL baselines described in Section 2.4.3 such as PNN (Rusu et al. 2016b), EWC (Kirkpatrick et al. 2017), PACKNET (Mallya and Lazebnik 2018), FT-1 which finetunes a single model on the entire sequence of tasks, and FT-L2 which is like FT-1 with an additional L_2 regularization applied during finetuning. We also compare with SAC-N which trains one model for each task from scratch.



(a) Forgetting table



(b) Transfer table

Figure 7.5 – Forgetting (a) and Transfer (b) tables of our Halfcheetah tasks. The pairs selected to create our scenario are highlighted in blue. Results are averaged over 3 seeds using a classical RL algorithm.

Method	HalfCheetah (4 scenarios)		Ant (4 scenarios)		Humanoid (1 scenario)	
	Performance	Model Size	Performance	Model Size	Performance	Model Size
FT-1	0.62 ± 0.29	1.0	0.52 ± 0.26	1.0	0.71 ± 0.07	1.0
FT-L2	0.38 ± 0.15	2.0	0.78 ± 0.20	2.0	0.68 ± 0.28	2.0
PACKNET	0.85 ± 0.14	2.0	1.08 ± 0.21	2.0	0.96 ± 0.21	2.0
EWC	0.43 ± 0.24	3.0	0.55 ± 0.24	3.0	0.94 ± 0.01	3.0
PNN	1.03 ± 0.14	8.0	0.98 ± 0.31	8.0	0.98 ± 0.26	4.0
SAC-N	1.00 ± 0.15	8.0	1.00 ± 0.38	8.0	1.00 ± 0.29	4.0
FT-N	1.16 ± 0.20	8.0	0.97 ± 0.20	8.0	0.65 ± 0.46	4.0
CSP (ours)	$\mathbf{1.27 \pm 0.27}$	5.4 ± 1.3	$\mathbf{1.11 \pm 0.17}$	3.9 ± 0.8	$\mathbf{1.76 \pm 0.19}$	3.4 ± 0.3
CSP-ORACLE	1.88 ± 0.19		1.24 ± 0.07		1.98 ± 0.22	

Table 7.1 – Aggregated results across all Brax scenarios from HalfCheetah, Ant, and Humanoid. These scenarios were designed to test forgetting, transfer, compositionality, and robustness. CSP performs as well as or better than the strongest baselines, while having a much lower model size and thus memory cost. CSP's performance is also not too far from that of CSP-ORACLE indicating that it can use the critic to find good policies in the subspace without requiring millions of interactions.

While SAC-N avoids forgetting, it cannot transfer knowledge across tasks. Finally, we compare with a method called FT-N which combines the best of both SAC-N and FT-1. Just like SAC-N, it stores one model per task after training on it and just like FT-1, it finetunes the previous model to promote transfer. However, FT-N and SAC-N scale poorly (*i.e.* linearly) with the number of tasks in terms of both memory and compute, which makes them unfeasible for real-world applications. Note that our method is not directly comparable with CLEAR (Rolnick et al. 2019) since we assume no access to data from prior tasks. Storing data from all prior tasks (as CLEAR does) is unfeasible for long task sequences due to prohibitive memory costs. All methods use SAC (Haarnoja et al. 2018a) as a base algorithm. The means and standard deviations are computed over 10 seeds unless otherwise noted. See Appendix A for more details about our protocol, baselines, and hyperparameters, respectively.

7.4 Results & Analysis

7.4.1 Performance on Brax

Table 7.1 shows the aggregated results across all scenarios from HalfCheetah, Ant, and Humanoid. The results are normalized using SAC-N's performance.

CSP performs as well as or better than the strongest baselines which grow linearly with the number of tasks (*i.e.* PNN, SAC-N, FT-N), while maintaining a much smaller size and thus memory cost. In fact, CSP's size grows sublinearly with the number of tasks. At the same time, CSP is significantly better than the most memory-efficient baselines that have a relatively small size (*i.e.* FT-1, FT-L2, EWC, PackNet). Naive methods like FT-1 have low memory costs and good transfer, but suffer from catastrophic forgetting. In contrast, methods that aim to reduce forgetting such as FT-L2 or EWC do so at the expense of transfer. The only competitive methods in terms of performance are the ones where the number of parameters increases at least linearly with the number of tasks, such as PNN, SAC-N, and FT-N. These methods have no forgetting because they store the models trained on each task. SAC-N has no transfer since it trains each model from scratch, while FT-N promotes transfer as it finetunes the previous model. However, due to their poor scalability, these methods are unfeasible for more challenging CRL scenarios with long and varied task sequences.

In contrast, CSP doesn't suffer from forgetting since the best found policies for all prior tasks can be cheaply stored in the form of convex combinations of the subspace's anchors (*i.e.* vectors rather than model weights). In addition, due to having access to a large number of policies (*i.e.* all convex combinations of the anchors), CSP has good transfer to new tasks (that share some similarities with prior ones). This allows CSP to achieve strong performance while its memory cost grows sublinearly with the number of tasks. Nevertheless, CSP's performance is not too far from that of CSP-ORACLE, indicating that the critic can be used to find good policies in the subspace without requiring millions of interactions. See Section 7.2.3 for more details about the use of the critic.

Method	CSP (ours)	PackNet	EWC	FT-L2	FT-1
Performance	0.81 ± 0.06	$\mathbf{0.83 \pm 0.02}$	0.66 ± 0.03	0.48 ± 0.05	0.10 ± 0.01
# Heads	$\mathbf{5.3 \pm 1.6}$	10	10	10	10

Table 7.2 – Results on the CW10 benchmark for CSP and other popular baselines including the state-of-the-art PackNet (results taken from Wołczyk et al. (2021)). CSP performs almost as well as PackNet while using about half the number of heads, and thus having a much lower memory cost.

7.4.2 Performance on Continual World

Table 7.2 shows results on CW10 (Wołczyk et al. 2021), a popular CRL benchmark. The baselines used in Wołczyk et al. (2021) use a separate linear layer for each task on top of a common network. The authors recognize this as a limitation since it scales poorly with the number of tasks. For a fair comparison, we implement CSP using a similar protocol where the anchors are represented by linear heads, so the number of heads grows adaptively depending on the performance threshold ϵ. Instead of model size, we compare the final number of heads, since this now determines a method's scalability with the number of tasks.

CSP performs almost as well as PackNet which is the current state-of-the-art on CW10 (Wołczyk et al. 2021), and is significantly better than all other baselines. At the same time, CSP uses about half the number of heads, thus being more memory efficient especially as the number of tasks increases. Note that PackNet suffers from a major limitation, namely that it requires prior knowledge of the total number of tasks in order to allocate resources. If this information is not correctly specified, PackNet is likely to fail due to either not being expressive enough to handle many tasks or being too inefficient while learning only a few tasks (Wołczyk et al. 2021). This makes PackNet unfeasible for real-world applications where agents can face task sequences of varying lengths (including effectively infinite ones). In contrast, CSP grows adaptively depending on the sequence of tasks, so it can handle both short and long sequences without any modification to the algorithm. See Appendix D2 for additional results on Continual World, including CW3.

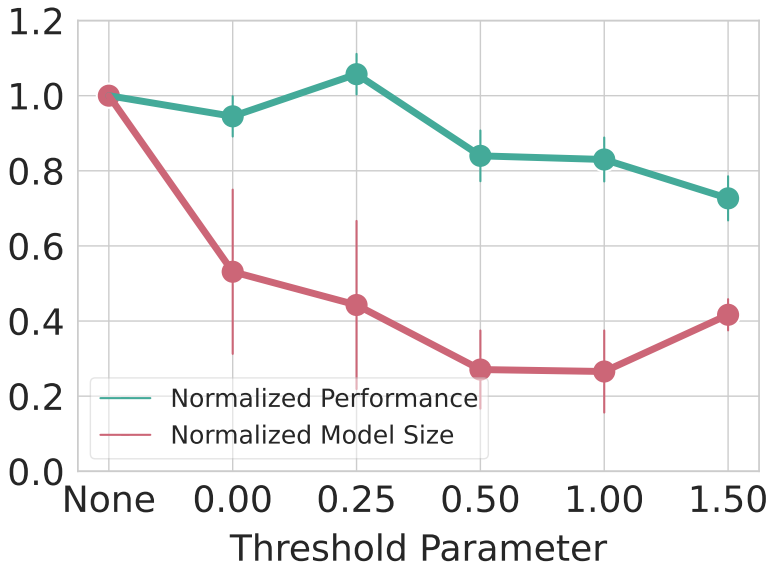


Figure 7.6 – Performance of CSP w.r.t. the Threshold

To summarize, CSP is competitive with the strongest CRL baselines, while having a lower memory cost since its size grows sublinearly with the number of tasks. Thus, CSP maintains a good balance between model size and performance, which allows it to scale well to long task sequences.

7.4.3 Varying the Threshold

Figure 7.6 shows how performance and size vary with the threshold ϵ used to decide whether to extend the subspace or not. Both the performance and the size are normalized with respect to CSP-LINEAR which always extends the subspace. As expected, as ϵ increases, performance decreases, but so does the size. Note that *performance is still above* 70% *even as the size is cut by more than* 70%, relative to CSP-LINEAR which trains a new set of parameters (*i.e.* anchor) for each task. Thus, CSP *can drastically reduce memory costs without significantly hurting performance*. Practitioners can set the threshold to trade-off performance gains for memory efficiency *i.e.* the higher the ϵ the more performance losses are tolerated in order to reduce memory costs. In practice, we found $\epsilon = 0.1$ to offer a good trade-off between performance and model size.

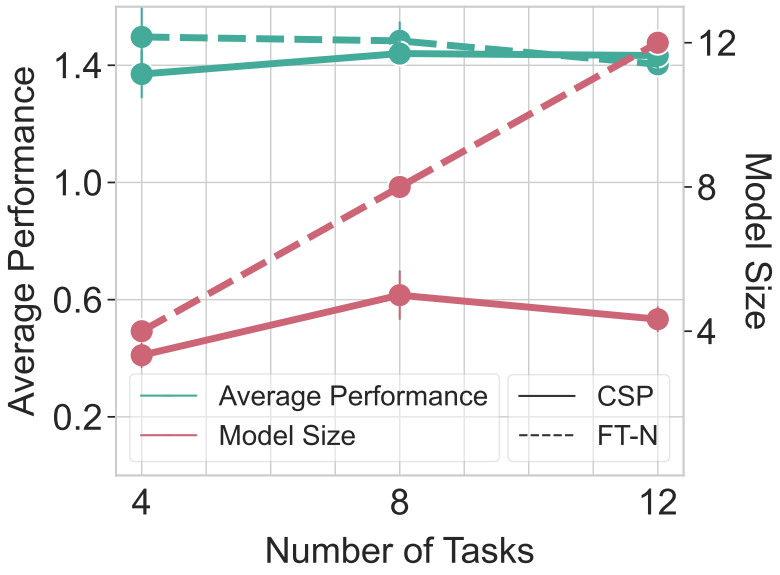


Figure 7.7 – Performance of CSP and FT-N w.r.t. the number of Tasks

7.4.4 Scalability

Figure 7.7 shows how performance and size scale with the number of tasks in the sequence (on HalfCheetah's compositionality scenario) for both CSP and FT-N which is our strongest baseline on this scenario. CSP maintains both strong performance and small size (*i.e.* low memory cost) even as the number of tasks increases. In contrast, even if FT-N performs well on all these scenarios, its size grows linearly with the number of tasks, rendering it impractical for long task sequences.

7.4.5 Learning Efficiency

Figure 7.8 shows the average performance for three different budgets (*i.e.* number of interactions allowed for each task) on HalfCheetah's robustness scenario, comparing CSP with FT-1, FT-L2, and EWC. These results demonstrate that CSP can learn efficiently even with a reduced budget, while still outperforming these baselines. By keeping track of all convex combinations of previously learned behaviors, CSP enables good transfer to new tasks which in turn leads to efficient training on task sequences.

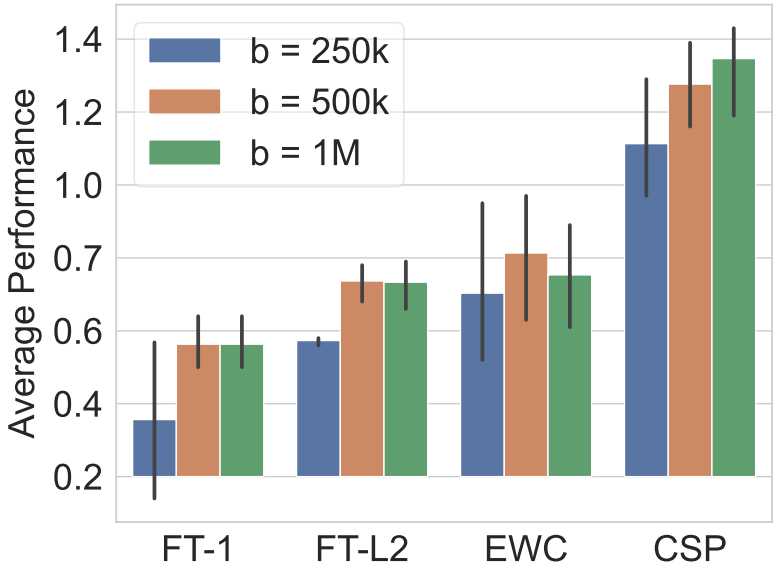


Figure 7.8 – Performance of multiple methods using different budgets.

7.4.6 Visualizing the Subspace

We also perform a number of ablations to understand how the different components of CSP influence performance. We first compare CSP with CSP-ORACLE which selects the best policy by sampling a large number of policies in the subspace and computing Monte-Carlo estimates of their returns. These estimates are expected to be more accurate than the critic's, so CSP-ORACLE can be considered an upper bound to CSP. However, CSP-ORACLE is less efficient than CSP since it requires significantly more interactions with the environment to find a policy for each task (*i.e.* millions).

The best way to visualize the reward and critic value landscapes of the subspaces is when there are 3 anchors (see Figure 7.9 and 7.10). To do saw, we draw 8192 evenly spaced points in the 3-dimensional simplex of $\mathbb{R}^3$, and average the return over 10 rollouts for the reward landscape, and 1024 pairs of (s, a) for the critic landscape. We used the short version of the Compositional Scenario of HalfCheetah to display the results.

We also want to understand how much performance we lose, if any, by not adding one anchor per task. To do this, we run an ablation with no threshold which always extends the subspace by adding one anchor for each new task. In addition, we vary the threshold ϵ used to decide whether

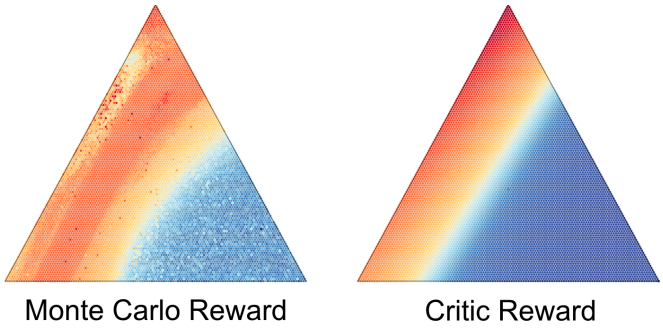


Figure 7.9 – Value of each policy in the subspace estimated using both Monte-Carlo simulations (left) and our critic's predictions (right), demonstrating both that the subspace is smooth and that the critic learns accurate estimates of the reward, which allows CSP to find good policies in the subspace

to extend the subspace based on how much performance is gained by doing so. This analysis can shed more light on the trade-off between performance gain and memory cost as the threshold varies.

7.4.7 Smoothness and Critic Accuracy.

Figure 7.9 shows a snapshot of a trained subspace, along with the expected reward of all policies in the subspace, for a given task. The expected reward is computed using both Monte-Carlo (MC) rollouts, as well as our critic's predictions using Equation 7.2. As illustrated, the learned Q-function has a similar landscape with the MC reward *i.e.* the subspace is smooth and the critic's estimates are accurate.

7.4.8 Diversity and Compositionality.

Figure 7.10 illustrates that the subspace contains behaviors composed of previously learned skills (*e.g.* walk on the moon, walk with tiny feet, walk on the moon with tiny feet). This allows CSP to reuse previously learned skills to find good policies for new tasks without the need for additional parameters. The figure also shows that for a given task, the policies in the subspace cover the whole spectrum of rewards, thus emphasizing the diversity of the behaviors expressed by the subspace.

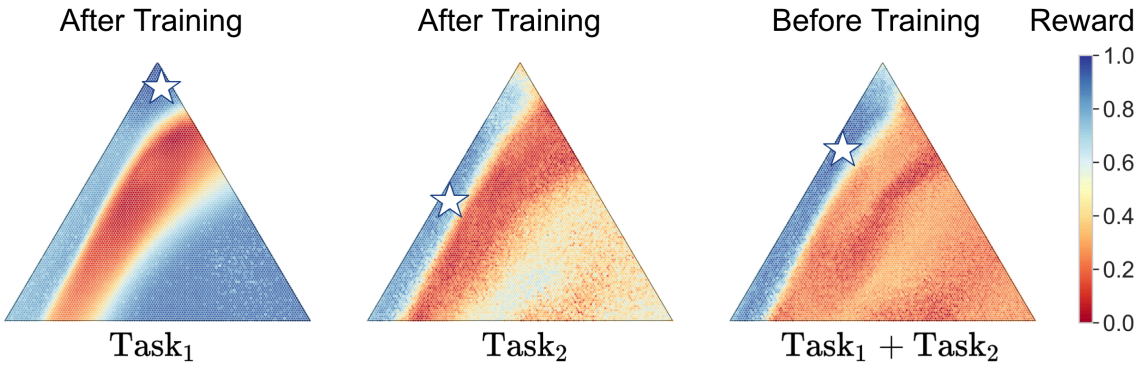


Figure 7.10 – Subspace for three different tasks, the third being a combination of the first two (*e.g.* walk on the moon (task 1) with tiny feet (task 2)), demonstrating that the subspace already contains a policy with high reward on the compositional task before being trained on it. The star represents the best policy in the subspace for the corresponding task. The subspace contains policies covering the whole spectrum of rewards, suggesting that it captures diverse behaviors. Click here to see interactive visualizations of the subspace.

7.4.9 Limitations

CSP prevents forgetting of prior tasks, promotes transfer to new tasks, and scales sublinearly with the number of tasks. Despite all these advantages, our method still has a number of limitations. While the subspace grows only sublinearly with the number of tasks, this number is highly dependent on the task sequence. In the worst case scenario, it increases linearly with the number of tasks. On the other hand, the more similar the tasks are, the lower the size of the subspace needed to learn good policies for all tasks. Thus, one important direction for future work is to learn a subspace of policies with a fixed number of anchors. Instead of training an additional anchor for each new task, one could optimize a policy in the current subspace on the new task while ensuring that the best policies for prior tasks don't change too much. This could be formulated as a constrained optimization problem where all the anchors defining the subspace are updated for each new task, but some regions of the subspace are regularized to not change very much. This would result in the subspace having different regions which are good for different tasks.

While the memory costs increase sublinearly with the number of tasks, the computational costs increase linearly with the number of tasks, in the

current implementation of CSP. This is because we train an additional anchor for each new task, which can be removed if it doesn't significantly improve performance. However, the computational costs can also be reduced if you have access to maximum reward on a task. This is typically the case for sparse reward tasks where if the agent succeeds, it receives a reward of 1 and 0 otherwise. In this case, there is no need to train one anchor per task. Instead, one can simply find the best policy in the current subspace and compare its performance with the maximum reward to decide whether to train an additional anchor or not. Hence, in this version of CSP (which is a minor modification of the current implementation) both memory and compute scale sublinearly with the number of tasks. However, this assumption doesn't always hold, so here we decided to implement the more general version of CSP.

In this work, we don't specifically leverage the structure of the subspace in order to find good policies. Hence, one promising research direction is to further improve transfer efficiency by leveraging the structure of the subspace to find good policies for new tasks. For example, this could be done by finding the convex combination of the anchors which maximizes return on a given task. Regularizing the geometry of the subspace to impose certain inductive biases could also be a fruitful direction for future work.

7.5 Conclusion

In this Chapter, we propose CSP, a new continual RL method which adaptively builds a subspace of policies to learn a sequence of tasks. CSP is competitive with the best existing methods while using significantly fewer parameters. Thus, it strikes a good balance between performance and memory cost which allows it to scale well to long task sequences. Our method is first to use a subspace of policies for continual RL, and thus opens up many interesting directions for future work. For example, one can assume a fixed size for the subspace and update all the anchors whenever the agent encounters a new task. Another promising direction is to leverage the structure of the subspace to meta-learn or search for good policies on a given task.

PART IV

Conclusion & Perspectives

CHAPTER 8

OTHER CONTRIBUTIONS

Alongside the main work of my thesis, my journey in the domain of deep learning has been enriched by additional contributions. I will share insights from working on *SaLinA* and *WorldSense*. My ongoing collaboration with Meta AI's Llama team represents a frontier of research that, due to its nascent stage, will be discussed in a limited capacity.

8.1 SaLinA

Everything is an Agent

SaLinA (Denoyer et al. 2021), developed as an extension of PyTorch, is a Python library designed to simplify the implementation of complex sequential learning models, including reinforcement learning algorithms. Its primary advantage lies in its simplicity and adaptability, allowing PyTorch users to easily understand and modify algorithms coded with SaLinA. Notably, the library can efficiently utilize multiple CPUs and GPUs, making it suitable for large-scale training scenarios. SaLinA's broad scope captures various settings such as model-based RL, batch RL, hierarchical RL, and multi-agent RL, making it accessible not only to RL practitioners but also to any deep learning programmer.

The strength of SaLinA compared to existing RL libraries lies in its ability to parallelize tasks and leverage multiple processing units effectively. This parallel execution capability is achieved through a unique architecture that enables agents to operate in different processes, speeding up their execution. This approach reduces the high adoption cost and complexity associated with traditional RL platforms. Moreover, the library's design facilitates the

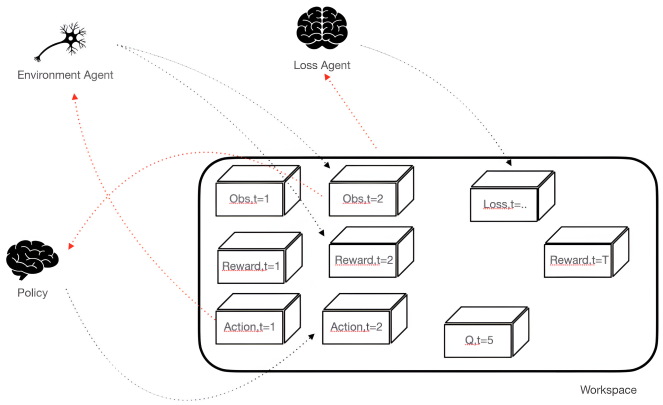


Figure 8.1 – SaLinA principles: Multiple agents read (black dotted lines) and write (red dotted lines) information into a workspace. This workspace corresponds to a common knowledge base which is iteratively updated by a diverse group of agents. Very similar to a blackboard in multi-agent systems, this architecture handles complex, ill-defined problems, where the solution is the sum of its parts.

easy development of model-based RL methods and other complex models, ensuring scalability and flexibility in various learning settings.

Another key principle in SaLinA is treating every component as an Agent, streamlining the process for both understanding and developing complex models. This approach has significantly simplified the implementation of experiments in Chapter 4, 7 and 6, particularly with the integration of the Brax environment. By treating elements such as environments, policies, and even data loaders as Agents, SaLinA facilitates a unified and flexible architecture. This design choice not only eases the learning curve for new programmers but also enables seamless transitions between different types of agents, such as replacing a environment agent (e.g. an agent executing a Gym environment) with a parametric agent (e.g. a neural network instantiating a policy). Consequently, the development of model-based RL methods and other intricate models becomes more intuitive and efficient. This uniformity in treating all components as Agents has been instrumental in the enhanced implementation and efficiency of the algorithms discussed in these chapters.

This work on SaLinA has been one of the most inspiring aspects of my thesis. It shifted my perspective on reinforcement learning (RL) from viewing it as a mere amalgamation of algorithms from machine learning and optimal control, to seeing it as a distinct problem of *sequential decision mak-*

ing. This change in viewpoint led me to scrutinize aspects often relegated to appendices in many papers, yet are central to the methods: How many policies to instantiate in parallel? How frequently should policies be refreshed in algorithms like SAC or PPO? How to make a replay buffer more effective? These questions, sometimes overlooked by the community, are crucial for the success of an experiment.

My contribution to SaLinA involved enhancing the library with benchmarks on continuous control tasks, including integration with the Brax environment (see `here`), and developing a continual learning module (see `here`). These additions were critical in extending the library's applicability to more diverse scenarios, particularly in the context of sequential reinforcement learning tasks. The continual learning module, linked to the concepts explored in chapter 6 of my thesis, provides a framework for benchmarking easily any continual learning methods on several Brax scenarios, thereby enriching SaLinA's capability in the domain of CRL.

8.2 WorldSense

This section outlines my contribution to the paper *WorldSense: A Benchmark for Grounded Language Understanding in Large Language Models* (Benchekroun et al. 2023), which evaluates the ability of LLMs to comprehend and interpret the world through textual data. My role was primarily focused on the fine-tuning and assessment of these models, particularly Llama2, to gauge their effectiveness in leveraging text to construct and reason about world states. The objective was to illuminate the capabilities and limitations of contemporary LLMs in tasks necessitating an understanding of real-world contexts.

LLMs have shown remarkable abilities, such as understanding and generating text that resembles human communication, executing complex reasoning tasks, and seemingly possessing a rudimentary understanding of the world solely based on textual input (Brown et al. 2020a). This perceived understanding introduces both potential and challenges in accurately determining the depth of LLMs' real-world concept comprehension and their capacity to develop consistent, coherent world representations. The WorldSense benchmark was created to confront these challenges by assessing

LLMs' abilities to conceive, manipulate, and reason about world states from textual descriptions.

Inspired by cognitive assessments conducted on animals and young children, WorldSense introduces a pioneering benchmark designed to investigate the implicit world models within LLMs (Gurnee and Tegmark 2023; J. Roberts et al. 2023). By avoiding the biases and memorization issues inherent in earlier benchmarks through the use of abstract schemata applied across various textual "skins," this benchmark provides a genuine test of an LLM's world understanding and reasoning capabilities, independent of domain and resistant to simple memorization tactics (Sainz et al. 2023; K. Zhou et al. 2023). It is structured around three problem types: grounded inferences, consistency detection, and completeness detection, each targeting different aspects of world understanding and reasoning.

Our experimental setup involved evaluating three leading chat LLMs: GPT3.5, GPT4, and Llama2-chat, against the challenges posed by WorldSense. We analyzed their responses for accuracy and any biases in their replies. The tests examined the models' proficiency in making grounded inferences, identifying inconsistencies, and evaluating the completeness of descriptions under various conditions, including basic controls designed to isolate specific reasoning challenges.

WorldSense serves as a significant step towards understanding the extent and nature of tacit world models in LLMs, circumventing previous benchmarks' limitations concerning bias and memorization, while addressing the challenges of translating between language and the non-verbal, multidimensional concept of the world (Bowman 2023). By focusing on linear order problems, which are cognitively simple yet widespread in language and relevant across multiple domains, *WorldSense* aims to provide a comprehensive assessment of LLMs' ability to learn and infer from text alone about linear relationships and transitive inferences, fundamental to both human and animal cognition (Lazareva and Wasserman 2010; Bryant and Trabasso 1971; Gillan 1981; W. A. Roberts and Phelps 1994; Von Fersen et al. 1991; Grosenick et al. 2007).

The results highlighted in Figure 8.2 demonstrate a nuanced picture of LLM capabilities and limitations. While larger models like GPT4 showed a somewhat better ability to perform grounded inferences and detect inconsistencies, all models struggled with the more complex tasks of consistency

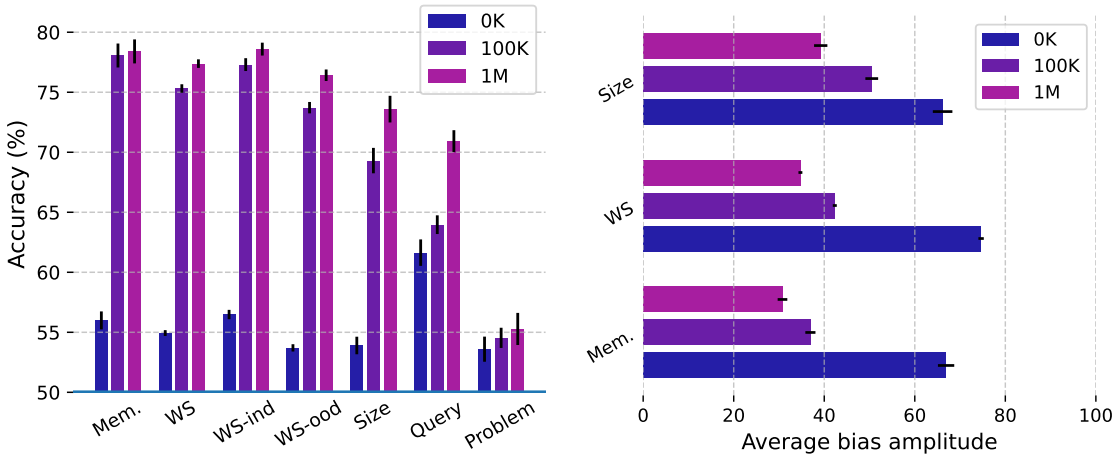


Figure 8.2 – **Finetuning results.** Left: Accuracy across Llama2_{70B} models finetuned on 0, 100K and 1M training examples, on the WS test set split into in-domain (WS-ind) and out-of-domain (WS-ood) subsets, plus a memorisation test set (Mem.), and the ood-size, ood-query and ood-problem generalisation test sets. Right: Bias amplitude on the memorisation, WS and length generalisation test sets. Chance levels for accuracies are 50%, error bars denote 95% confidence intervals for both plots.

and completeness detection. Notably, the experiments revealed a significant challenge in translating between textual descriptions and coherent world states—a core component of world understanding. My involvement in this project, particularly in running and evaluating the Llama2 experiments, focused on finetuning Llama2 models with an ablation study to understand how specific model adjustments affected performance across the different types of WorldSense problems.

This study underscores the complexity of evaluating LLMs' world understanding and highlights the innovative approach of WorldSense in providing a nuanced, challenging benchmark for future research. Despite advancements, the findings emphasize the need for continued efforts to enhance LLMs' abilities to form, manipulate, and reason about tacit world models.

8.3 Ongoing work with Llama's team

I am currently involved in the development of a new version of Llama with Meta AI's "Gen AI" team. The initial version, as cited in (Touvron et al.

2023a), focused on providing open-source foundational language models (referred to as "pre-trained" models), whereas the second iteration (Touvron et al. 2023b) introduced models that were also fine-tuned. A key concept in this paper was *rejection sampling*, which involves creating a pool of (prompts, generations) that received high rewards from the *reward model*, and then fine-tuning on these samples before proceeding to RLHF.

Part of my role is to refine this work, which includes developing better datasets and more advanced rejection sampling strategies, incorporating *self-refinement* techniques. Another aspect of my contribution is applying subspaces during the RLHF phase, drawing inspiration from the method developed in Chapter 4, to create a *diverse* model population.

CHAPTER 9

CONCLUSION & FUTURE DIRECTIONS

9.1 Conclusion

Throughout this manuscript, I have addressed three pivotal questions posed in the introduction. 1) Is deep reinforcement learning mature enough to tackle zero-shot generalization? Part II offers mixed insights: Chapter 3 (A Pathological Example), while highlighting significant advancements in ZSG, also uncovers substantial limitations that RL is not yet equipped to overcome. Chapter 4 (Weight Averaging for Multi-objective RL) introduces an appealing new strategy but leaves the question of user adaptation unresolved. This leads to the second question, 2) can we envision a simple and intuitive method to facilitate an agent's adaptation to a new task? In Part III, and particularly in Chapter 6 (Learning a Line of Policies), I demonstrate the feasibility of leveraging a new framework, *subspace of policies*, for effective adaptation in few-shot settings. Finally, this brings us to question 3) can this method be generalized in a more real-world context where an autonomous agent faces a sequence of tasks? Chapter 7 (Building a Subspace of Policies) in the most ambitious CRL setting of this thesis shows that subspaces benefit from being *built*, rather than merely used ad-hoc as a simple manifold of policies. This last method paves the way for numerous research directions, some of which I will introduce in the next section of this Chapter.

The exploration of subspaces in RL represents a promising frontier that challenges traditional paradigms and encourages a reconsideration of how we approach generalization and adaptability in AI models. The future of subspaces, as demonstrated through this research, lies in their potential to craft more dynamic, responsive, and efficient learning systems. By continuing to investigate and refine these methods, we open the door to AI that can more adeptly navigate the complexities of real-world tasks, paving the way

for advancements that could redefine the boundaries of machine learning capabilities.

Reflecting on my journey through this thesis, the learnings extend beyond the technical achievements and into the realm of personal growth and understanding. The process of dissecting and reconstructing the foundations of RL has not only contributed to the academic community but has also offered invaluable lessons in persistence, critical thinking, and the importance of creative approaches to problem-solving. This work underscores the significance of questioning assumptions and relentlessly pursuing a deeper comprehension of the mechanisms that drive deep learning.

9.2 Future Directions

The new framework we described in Part III opens the door to numerous applications. Some relate to the application to a specific problem (for example privacy and efficiency in *decentralized deep learning*); while others are still in the embryonic stage of research (for example, combining *modularity* with subspaces). Here are two areas that I am particularly interested in exploring in the future.

9.2.1 Subspaces as a framework for decentralized deep learning

Decentralized deep learning (DDL) emerges as a pivotal innovation in the domain of AI (Sun et al. 2021), primarily addressing the challenges of privacy (Tran et al. 2021) and efficiency (M. Li et al. 2014) in distributed computing environments. Leveraging technologies like federated learning and swarm learning, DDL enables collaborative model training across multiple edge devices without the necessity to centralize sensitive data (see Figure 9.1). This approach significantly enhances privacy protections, particularly vital in sectors such as healthcare (Poirot et al. 2019) and finance (Ratadiya et al. 2020) where data sensitivity is paramount.

In the field of RL, the integration of DDL principles holds transformative potential, particularly in robotics. In the field of RL, the integration of DDL principles holds transformative potential, particularly in robotics. For example B. Liu et al. (2019) developed a framework aimed at enhancing

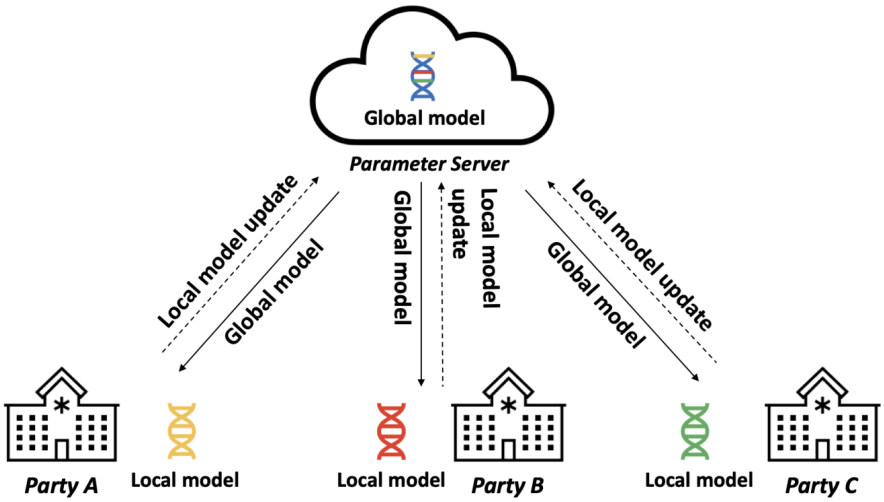


Figure 9.1 – Federated Learning Architecture: A cloud representing the global model receives updated parameters from local models coordinated by a server. This iterative process leads to convergence of the global model. Taken from Sun et al. (2021)

robot navigation in cloud robotic systems. This framework facilitates the fusion and transfer of experiences across robots, allowing them to effectively leverage prior knowledge and adapt swiftly to new environments. Leveraging federated and swarm learning approaches, DDL enables collaborative model training directly on devices, a paradigm shift that enhances privacy and computational efficiency. Extending these principles to video gaming, especially in client/server frameworks (Gambetta 2024), suggests that leveraging client-side resources for inference and optimization can significantly enhance gameplay experiences and efficiencies, by distributing the computational load and enabling real-time adaptations to game dynamics.

Weight averaging methods are well-suited to these specific settings and have been tested for DDL. However, these methods focus on memory and computational efficiency, privacy, and overlook the possibility of training and maintaining a whole subspace of policies. Kamp et al. (2019) discuss a *dynamic averaging protocol* to synchronize models when they diverge significantly, sending the average model to the client as needed, and the client model to the server when insightful. Enhancing this framework could involve sending not just a simple average model to the client but a more personalized weight combination, even during inference, possibly through active learning (P. Ren et al. 2021b). An idea could be to explore a bal-

ance between *specific* and *global* updates within the framework. Specifically, through active learning techniques (P. Ren et al. 2021b), one could dynamically adjust the model from the server to the client, tailoring the experience to the user's current performance, such as by modulating the difficulty of an opposing agent (e.g. a non-playable character that the player has to defeat). Globally, the aggregation of insights from all clients back to the server could serve to refine and enhance the overarching subspace of policies, illustrating a symbiotic enhancement of personalized and collective learning outcomes (a bit like the method proposed in Chapter 7).

9.2.2 Boosting neural architecture search with subspaces

Neural architecture search (NAS) (P. Ren et al. 2021a) represents a paradigm shift in the automated design of neural networks, optimizing architecture to achieve superior performance with minimal human intervention. NAS automates the process of selecting the best architecture from a vast space, utilizing techniques such as reinforcement learning (Zoph and Le 2016) and evolutionary algorithms (Y. Liu et al. 2021) to identify optimal structures for specific tasks. This approach has demonstrated significant advancements in areas ranging from image classification to sequence modeling, highlighting its potential to revolutionize how neural networks are conceived and optimized.

The integration of subspaces of policies into NAS could transform the search process itself, focusing on *combinations* of modules rather than discrete module selections. This approach enables a continuous and dynamic exploration of architectural configurations within a defined subspace, facilitating the identification of optimal structures through a blend of exploration and exploitation. By treating the search space as a continuous domain, it's possible to apply techniques like active learning and gradient-based optimization, allowing for real-time adjustments based on performance feedback. This method extends the flexibility of NAS, enabling the system to navigate through a richer landscape of architectural possibilities and dynamically adapt architectures in response to evolving tasks or environments. Such an approach could vastly increase the versatility, efficiency, and robustness of NAS-oriented models.

BIBLIOGRAPHY

Abel, David, Dilip Arumugam, Lucas Lehnert, and Michael L. Littman (2017). "Toward Good Abstractions for Lifelong Learning". In: (cit. on p. 35).

Agarap, Abien Fred (2018). "Deep learning using rectified linear units (relu)". In: *arXiv preprint* (cit. on p. 65).

Ahmed, Hadeer (2017). "Detecting opinion spam and fake news using n-gram analysis and semantic similarity". PhD thesis (cit. on p. 70).

Askell, Amanda, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Jackson Kernion, Kamal Ndousse, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, and Jared Kaplan (2021). "A General Language Assistant as a Laboratory for Alignment". In: *arXiv preprint* (cit. on p. 66).

Astrom, Karl J et al. (1965). "Optimal control of Markov processes with incomplete state information". In: *Journal of mathematical analysis and applications* 10.1, pp. 174–205 (cit. on p. 30).

Bai, Yuntao, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. (2022a). "Training a helpful and harmless assistant with reinforcement learning from human feedback". In: *arXiv preprint* (cit. on pp. 63, 70).

Bai, Yuntao, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. (2022b). "Constitutional AI: Harmlessness from AI Feedback". In: *arXiv preprint* (cit. on p. 63).

Bain, Michael and Claude Sammut (1995). "A Framework for Behavioural Cloning". In: *Machine Intelligence 15*. URL: https://api.semanticscholar.org/CorpusID:10738655 (cit. on pp. vii, viii).

Banerjee, Satanjeev and Alon Lavie (2005). "METEOR: An automatic metric for MT evaluation with improved correlation with human judgments". In: *ACL Workshop* (cit. on p. 72).

Barrett, Leon and Srini Narayanan (2008). "Learning all optimal policies with multiple criteria". In: *ICML* (cit. on pp. 22, 64, 66).

Beeching, Edward, Younes Belkada, Leandro von Werra, Sourab Mangrulkar, Lewis Tunstall, and Kashif Rasul (2023). *Fine-tuning 20B LLMs with RLHF on a 24GB consumer GPU*. https://huggingface.co/blog/trl-peft (cit. on p. 70).

Beltagy, Iz, Matthew E. Peters, and Arman Cohan (2020). *Longformer: The Long-Document Transformer*. arXiv: 2004.05150 [cs.CL] (cit. on pp. ix, 49, 56).

Benchekroun, Youssef, Megi Dervishi, Mark Ibrahim, Jean-Baptiste Gaya, Xavier Martinet, Grégoire Mialon, Thomas Scialom, Emmanuel Dupoux, Dieuwke Hupkes, and Pascal Vincent (2023). *WorldSense: A Synthetic Benchmark for Grounded Reasoning in Large Language Models*. arXiv: 2311.15930 [cs.CL] (cit. on pp. xiii, 145).

Bengio, Yoshua and Yann LeCun (2007). "Scaling Learning Algorithms Towards AI". In: *Large Scale Kernel Machines*. MIT Press (cit. on p. 119).

Berseth, Glen, Cheng Xie, Paul Cernek, and Michiel Van de Panne (2018). *Progressive Reinforcement Learning with Distillation for Multi-Skilled Motion Control*. arXiv: 1802.04765 [cs.LG] (cit. on p. 119).

Berseth, Glen, Zhiwei Zhang, Grace Zhang, Chelsea Finn, and Sergey Levine (2022). "CoMPS: Continual Meta Policy Search". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=PVJ6j87gOHz (cit. on p. 34).

Bommasani, Rishi, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. (2021). "On the opportunities and risks of foundation models". In: *arXiv preprint* (cit. on p. 61).

Bowman, Samuel R (2023). "Eight things to know about large language models". In: *arXiv preprint arXiv:2304.00612* (cit. on p. 146).

Bradbury, James, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, and Qiao Zhang (2018). *JAX: composable transformations of Python+NumPy programs*. Version 0.3.13. URL: http://github.com/google/jax (cit. on p. 15).

Brockman, Greg, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba (2016). *OpenAI Gym*. eprint: arXiv:1606.01540 (cit. on p. 103).

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. (2020a). "Language models are few-shot learners". In: *Advances in neural information processing systems* 33, pp. 1877–1901 (cit. on p. 145).

Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec

Radford, Ilya Sutskever, and Dario Amodei (2020b). "Language Models are Few-Shot Learners". In: *NeurIPS* (cit. on pp. v, 3).

Bryant, Peter E and Thomas Trabasso (1971). "Transitive inferences and memory in young children." In: *Nature* (cit. on p. 146).

Campbell, Murray, A Joseph Hoane Jr, and Feng-hsiung Hsu (2002). "Deep blue". In: *Artificial intelligence* 134.1-2, pp. 57–83 (cit. on pp. v, 3).

Castelletti, Andrea, Francesca Pianosi, and Marcello Restelli (2013). "A multiobjective reinforcement learning approach to water resources systems operation: Pareto frontier approximation in a single run". In: *Water Resources Research* (cit. on p. 23).

Chang, Ming-Wei, Lev-Arie Ratinov, Dan Roth, and Vivek Srikumar (2008). "Importance of Semantic Representation: Dataless Classification." In: *Aaai*. Vol. 2, pp. 830–835 (cit. on p. 19).

Chankong, Vira and Yacov Y Haimes (2008). *Multiobjective decision making: theory and methodology*. Courier Dover Publications (cit. on p. 22).

Chatzilygeroudis, Konstantinos, Vassilis Vassiliades, Freek Stulp, Sylvain Calinon, and Jean-Baptiste Mouret (2019). "A Survey on Policy Search Algorithms for Learning Robot Controllers in a Handful of Trials". In: *IEEE Transactions on Robotics* (cit. on p. 27).

Chen, Lili, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch (2021). *Decision Transformer: Reinforcement Learning via Sequence Modeling*. arXiv: 2106.01345 [cs.LG] (cit. on pp. v, 3, 18).

Chen, Sihao, Fan Zhang, Kazoo Sone, and Dan Roth (2021). "Improving Faithfulness in Abstractive Summarization with Contrast Candidate Generation and Selection". In: *NAACL* (cit. on p. 70).

Chen, Tianqi, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang (2015). "Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems". In: *arXiv preprint arXiv:1512.01274* (cit. on p. 91).

Cheung, Brian, Alex Terekhov, Yubei Chen, Pulkit Agrawal, and Bruno A. Olshausen (2019). "Superposition of many models into one". In: *ArXiv* abs/1902.05522 (cit. on pp. 34, 119).

Chevalier-Boisvert, Maxime, Lucas Willems, and Suman Pal (2018). *Minimalistic Gridworld Environment for OpenAI Gym*. https://github.com/maximecb/gym-minigrid (cit. on p. 103).

Choromanska, Anna, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun (2015). *The Loss Surfaces of Multilayer Networks*. arXiv: 1412.0233 [cs.LG] (cit. on pp. 5, 87).

Christiano, Paul F, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei (2017). "Deep reinforcement learning from human preferences". In: *NeurIPS* (cit. on p. 70).

Cideron, Geoffrey, Thomas Pierrot, Nicolas Perrin, Karim Beguir, and Olivier Sigaud (2020). "Qd-rl: Efficient mixing of quality and diversity in reinforcement learning". In: *arXiv preprint arXiv:2006.08505*, p. 36 (cit. on p. 27).

Cobbe, Karl, Christopher Hesse, Jacob Hilton, and John Schulman (2019). "Leveraging Procedural Generation to Benchmark Reinforcement Learning". In: *arXiv preprint arXiv:1912.01588* (cit. on p. 103).

Colas, Cédric, Tristan Karch, Olivier Sigaud, and Pierre-Yves Oudeyer (2022). "Autotelic Agents with Intrinsically Motivated Goal-Conditioned Reinforcement Learning: A Short Survey". In: *Journal of Artificial Intelligence Research* 74, pp. 1159–1199 (cit. on p. 27).

Cully, Antoine, Jeff Clune, Danesh Tarapore, and Jean-Baptiste Mouret (2015). "Robots that can adapt like animals". In: *Nature* 521, pp. 503–507 (cit. on p. 27).

Deleu, Tristan and Yoshua Bengio (2018). *The effects of negative adaptation in Model-Agnostic Meta-Learning*. arXiv: 1812.02159 [cs.LG] (cit. on p. 25).

Denoyer, Ludovic, Alfredo de la Fuente, Song Duong, Jean-Baptiste Gaya, Pierre-Alexandre Kamienny, and Daniel H. Thompson (2021). *SaLinA: Sequential Learning of Agents*. https://gitHub.com/facebookresearch/salina (cit. on pp. xii, 103, 143).

Dewey, Dan (2014). "Reinforcement Learning and the Reward Engineering Principle". In: *AAAI* (cit. on p. 77).

Díaz-Rodríguez, Natalia, Vincenzo Lomonaco, David Filliat, and Davide Maltoni (2018). *Don't forget, there is more than forgetting: new metrics for Continual Learning*. arXiv: 1810.13166 [cs.AI] (cit. on p. 33).

Dorigo, Marco and Marco Colombetti (1994). "Robot shaping: Developing autonomous agents through learning". In: *Artificial intelligence* (cit. on p. 77).

Draxler, Felix, Kambis Veschgini, Manfred Salmhofer, and Fred A. Hamprecht (2019). *Essentially No Barriers in Neural Network Energy Landscape*. arXiv: 1803.00885 [stat.ML] (cit. on p. 87).

Duan, Yan, Xi Chen, Rein Houthooft, John Schulman, and P. Abbeel (2016a). "Benchmarking Deep Reinforcement Learning for Continuous Control". In: *ICML* (cit. on p. 77).

Duan, Yan, John Schulman, Xi Chen, Peter L. Bartlett, Ilya Sutskever, and Pieter Abbeel (2016b). "RL 2: $FastReinforcementLearningviaSlowReinforcementLearning$". In: *CoRR* abs/1611.02779. arXiv: 1611.02779. URL: http://arxiv.org/abs/1611.02779 (cit. on p. 24).

Dulac-Arnold, Gabriel, Nir Levine, Daniel J Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester (2021). "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis". In: *Machine Learning* (cit. on p. 63).

Eysenbach, Benjamin, Abhishek Gupta, Julian Ibarz, and Sergey Levine (2018). "Diversity is All You Need: Learning Skills without a Reward Function". In: *CoRR* abs/1802.06070. arXiv: 1802.06070. URL: http://arxiv.org/abs/1802.06070 (cit. on pp. 26, 103).

Fang, Meng, Yuan Li, and Trevor Cohn (2017). "Learning how to active learn: A deep reinforcement learning approach". In: *arXiv preprint arXiv:1708.02383* (cit. on p. 93).

Filos, Angelos, Panagiotis Tigkas, Rowan McAllister, Nicholas Rhinehart, Sergey Levine, and Yarin Gal (2020). "Can autonomous vehicles identify, recover from, and adapt to distribution shifts?" In: *International Conference on Machine Learning*. PMLR, pp. 3145–3153 (cit. on p. 19).

Finn, Chelsea, Pieter Abbeel, and Sergey Levine (2017). "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *arXiv e-prints*, arXiv:1703.03400, arXiv:1703.03400. arXiv: 1703.03400 [cs.LG] (cit. on pp. 24, 25, 98).

Finn, Chelsea, Kelvin Xu, and Sergey Levine (2019). *Probabilistic Model-Agnostic Meta-Learning*. arXiv: 1806.02817 [cs.LG] (cit. on p. 25).

Forestier, Sébastien and Pierre-Yves Oudeyer (2016). "Modular Active Curiosity-Driven Discovery of Tool Use". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Daejeon, South Korea. URL: https://hal.archives-ouvertes.fr/hal-01384566 (cit. on p. 28).

Fort, Stanislav, Huiyi Hu, and Balaji Lakshminarayanan (2020). *Deep Ensembles: A Loss Landscape Perspective*. arXiv: 1912.02757 [stat.ML] (cit. on pp. 87, 101).

Frankle, Jonathan, Gintare Karolina Dziugaite, Daniel M. Roy, and Michael Carbin (2020). *Linear Mode Connectivity and the Lottery Ticket Hypothesis*. arXiv: 1912.05671 [cs.LG] (cit. on p. 88).

Freeman, C Daniel, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem (2021a). "Brax–A Differentiable Physics Engine for Large Scale Rigid Body Simulation". In: *arXiv preprint arXiv:2106.13281* (cit. on pp. 49, 103, 127).

Freeman, C. Daniel, Erik Frey, Anton Raichuk, Sertan Girgin, Igor Mordatch, and Olivier Bachem (2021b). *Brax - A Differentiable Physics Engine for Large Scale Rigid Body Simulation*. Version 0.9.3. URL: http://github.com/google/brax (cit. on pp. xii, 14, 15, 40, 52, 77, 85).

Fu, Justin, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine (2020). "D4rl: Datasets for deep data-driven reinforcement learning". In: *arXiv preprint arXiv:2004.07219* (cit. on pp. 18, 41, 43).

Furuta, Hiroki, Yusuke Iwasawa, Yutaka Matsuo, and Shixiang Shane Gu (2023). "A System for Morphology-Task Generalization via Unified Representation and Behavior Distillation". In: *The Eleventh International Conference on Learning Representations*. URL: https://openreview.net/forum?id=HcUf-QwZeFh (cit. on pp. 21, 41–43, 48).

Gambetta, Gabriel (2024). *Client-Server Game Architecture*. https://gabrielgambetta.com/client-server-game-architecture.html. Accessed: 2024-02-22 (cit. on p. 151).

Ganguli, Deep, Liane Lovitt, Jackson Kernion, Amanda Askell, Yuntao Bai, Saurav Kadavath, Ben Mann, Ethan Perez, Nicholas Schiefer, Kamal Ndousse, et al. (2022). "Red teaming language models to reduce harms: Methods, scaling behaviors, and lessons learned". In: *arXiv preprint* (cit. on p. 63).

Garipov, Timur, Pavel Izmailov, Dmitrii Podoprikhin, Dmitry Vetrov, and Andrew Gordon Wilson (2018). *Loss Surfaces, Mode Connectivity, and Fast Ensembling of DNNs*. arXiv: 1802.10026 [stat.ML] (cit. on p. 87).

Gaya, Jean-Baptiste, Thang Doan, Lucas Caccia, Laure Soulier, Ludovic Denoyer, and Roberta Raileanu (2023). *Building a Subspace of Policies for Scalable Continual Learning*. arXiv: 2211.10445 [cs.LG] (cit. on pp. xi, 97, 117).

Gaya, Jean-Baptiste, Laure Soulier, and Ludovic Denoyer (2021). "Learning a subspace of policies for online adaptation in Reinforcement Learning". In: *ArXiv* abs/2110.05169 (cit. on pp. xi, 98).

Ghosh, Dibya, Jad Rahme, Aviral Kumar, Amy Zhang, Ryan P Adams, and Sergey Levine (2021). "Why Generalization in RL is Difficult: Epistemic POMDPs and Implicit Partial Observability". In: *Advances in Neural Information Processing Systems*. Ed. by A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan. URL: https://openreview.net/forum?id=QWIvzSQaX5 (cit. on p. 20).

Gillan, Douglas J (1981). "Reasoning in the chimpanzee: II. Transitive inference." In: *Journal of Experimental Psychology: Animal Behavior Processes* 7.2, p. 150 (cit. on p. 146).

Glorot, Xavier and Yoshua Bengio (2010). "Understanding the difficulty of training deep feedforward neural networks". In: *Proceedings of the thirteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, pp. 249–256 (cit. on p. 85).

Goldberg, Yoav (2023). *Reinforcement Learning for Language Models*. https://gist.github.com/yoavg/6bff0fecd65950898eba1bb321cfbd81 (cit. on p. 62).

Gotmare, Akhilesh, Nitish Shirish Keskar, Caiming Xiong, and Richard Socher (2018). *Using Mode Connectivity for Loss Landscape Analysis*. arXiv: 1806.06977 [cs.LG] (cit. on p. 87).

Grosenick, Logan, Tricia S Clement, and Russell D Fernald (2007). "Fish can infer social rank by observation alone". In: *Nature* 445.7126, pp. 429–432 (cit. on p. 146).

Gu, Shixiang Shane, Ethan Holly, Timothy P. Lillicrap, and Sergey Levine (2017). "Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates". In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 3389–3396 (cit. on p. 17).

Gupta, Agrim, Linxi Fan, Surya Ganguli, and Li Fei-Fei (2022). "MetaMorph: Learning Universal Controllers with Transformers". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=Opmqtk_GvYL (cit. on pp. 21, 41).

Gurnee, Wes and Max Tegmark (2023). "Language Models Represent Space and Time". In: *arXiv preprint arXiv:2310.02207* (cit. on p. 146).

Haarnoja, Tuomas, Aurick Zhou, Pieter Abbeel, and Sergey Levine (2018a). "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor". In: *International conference on machine learning*. PMLR, pp. 1861–1870 (cit. on pp. vii, 17, 124, 125, 131).

Haarnoja, Tuomas, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, et al. (2018b). "Soft actor-critic algorithms and applications". In: *arXiv preprint arXiv:1812.05905* (cit. on p. 18).

Han, Isaac, Dong-Hyeok Park, and Kyung-Joong Kim (2021). "A new open-source off-road environment for benchmark generalization of autonomous driving". In: *IEEE Access* 9, pp. 136071–136082 (cit. on p. 19).

Hausman, Karol, Jost Tobias Springenberg, Ziyu Wang, Nicolas Heess, and Martin A. Riedmiller (2018). "Learning an Embedding Space for Transferable Robot Skills". In: *ICLR (Poster)*. OpenReview.net (cit. on p. 24).

Hayes, Conor F, Roxana Rădulescu, Eugenio Bargiacchi, Johan Källström, Matthew Macfarlane, Mathieu Reymond, Timothy Verstraeten, Luisa M Zintgraf, Richard Dazeley, Fredrik Heintz, et al. (2022). "A practical guide to multi-objective reinforcement learning and planning". In: *JAAMAS* (cit. on pp. 23, 63–65).

Hejna, Donald, Lerrel Pinto, and Pieter Abbeel (2020). "Hierarchically Decoupled Imitation For Morphological Transfer". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 4159–4171. URL: https://proceedings.mlr.press/v119/hejna20a.html (cit. on p. 21).

Hinton, Geoffrey E., Simon Osindero, and Yee Whye Teh (2006). "A Fast Learning Algorithm for Deep Belief Nets". In: *Neural Computation* 18, pp. 1527–1554 (cit. on pp. 35, 119).

Ho, Jonathan, Ajay Jain, and Pieter Abbeel (2020). *Denoising Diffusion Probabilistic Models*. arXiv: 2006.11239 [cs.LG] (cit. on pp. v, 3, 73).

Hong, Sunghoon, Deunsol Yoon, and Kee-Eung Kim (2022). "Structure-Aware Transformer Policy for Inhomogeneous Multi-Task Reinforcement Learn-

ing". In: *International Conference on Learning Representations*. URL: https://openreview.net/forum?id=fy_XRVHqly (cit. on p. 21).

Hu, Edward J, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen (2022). "LoRA: Low-Rank Adaptation of Large Language Models". In: *ICLR* (cit. on p. 70).

Hu, Jia Cheng, Roberto Cavicchioli, and Alessandro Capotondi (2022). "ExpansionNet v2: Block Static Expansion in fast end to end training for Image Captioning". In: *arXiv preprint* (cit. on p. 72).

Huang, Wenlong, Igor Mordatch, and Deepak Pathak (2020a). "One Policy to Control Them All: Shared Modular Policies for Agent-Agnostic Control". In: *Proceedings of the 37th International Conference on Machine Learning*. Ed. by Hal Daumé III and Aarti Singh. Vol. 119. Proceedings of Machine Learning Research. PMLR, pp. 4455–4464. URL: https://proceedings.mlr.press/v119/huang20d.html (cit. on p. 21).

Huang, Wenlong, Igor Mordatch, and Deepak Pathak (2020b). "One policy to control them all: Shared modular policies for agent-agnostic control". In: *International Conference on Machine Learning*. PMLR, pp. 4455–4464 (cit. on p. 45).

Humplik, Jan, Alexandre Galashov, Leonard Hasenclever, Pedro A. Ortega, Yee Whye Teh, and Nicolas Heess (2019). "Meta reinforcement learning as task inference". In: *arXiv e-prints*, arXiv:1905.06424, arXiv:1905.06424. arXiv: 1905.06424 [cs.LG] (cit. on p. 24).

Ioffe, Sergey and Christian Szegedy (2015). "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *ICML* (cit. on p. 65).

Irvine, Robert, Douglas Boubert, Vyas Raina, Adian Liusie, Vineet Mudupalli, Aliaksei Korshuk, Zongyi Liu, Fritz Cremer, Valentin Assassi, Christie-Carol Beauchamp, et al. (2023). "Rewarding Chatbots for Real-World Engagement with Millions of Users". In: *arXiv preprint* (cit. on p. 63).

Ishii, Shin, Wako Yoshida, and Junichiro Yoshimoto (2002). "Control of exploitation–exploration meta-parameter in reinforcement learning". In: *Neural networks* 15.4-6, pp. 665–687 (cit. on p. 13).

Izmailov, Pavel, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson (2018). "Averaging weights leads to wider optima and better generalization". In: *UAI* (cit. on p. 73).

Izmailov, Pavel, Dmitrii Podoprikhin, Timur Garipov, Dmitry Vetrov, and Andrew Gordon Wilson (2019). *Averaging Weights Leads to Wider Optima and Better Generalization*. arXiv: 1803.05407 [cs.LG] (cit. on p. 88).

Jamal, Muhammad Abdullah and Guo-Jun Qi (2019). "Task Agnostic Meta-Learning for Few-Shot Learning". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (cit. on p. 25).

Javed, Khurram and Martha White (2019). "Meta-Learning Representations for Continual Learning". In: *NeurIPS* (cit. on p. 35).

Kaelbling, Leslie Pack, Michael L Littman, and Andrew W Moore (1996). "Reinforcement learning: A survey". In: *Journal of artificial intelligence research* 4, pp. 237–285 (cit. on pp. 21, 39).

Kamp, Michael, Linara Adilova, Joachim Sicking, Fabian Hüger, Peter Schlicht, Tim Wirtz, and Stefan Wrobel (2019). "Efficient decentralized deep learning by dynamic model averaging". In: *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2018, Dublin, Ireland, September 10–14, 2018, Proceedings, Part I 18*. Springer, pp. 393–409 (cit. on p. 151).

Kaplanis, Christos, Murray Shanahan, and Claudia Clopath (2019). "Policy Consolidation for Continual Reinforcement Learning". In: *ArXiv* abs/1902.00255 (cit. on p. 119).

Karpathy, Andrej and Li Fei-Fei (2015). "Deep visual-semantic alignments for generating image descriptions". In: *CVPR* (cit. on p. 73).

Kessler, Samuel, Jack Parker-Holder, Philip Ball, Stefan Zohren, and Stephen J Roberts (2022). "Same State, Different Task: Continual Reinforcement Learning without Interference". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 36. 7, pp. 7143–7151 (cit. on p. 34).

Khetarpal, Khimya, Matthew Riemer, Irina Rish, and Doina Precup (2020). "Towards Continual Reinforcement Learning: A Review and Perspectives". In: *ArXiv* abs/2012.13490 (cit. on pp. 20, 30, 32, 34).

Kim, Taesup, Jaesik Yoon, Ousmane Dia, Sungwoong Kim, Yoshua Bengio, and Sungjin Ahn (2018). *Bayesian Model-Agnostic Meta-Learning*. arXiv: 1806.03836 [cs.LG] (cit. on p. 25).

Kingma, Diederik P. and Jimmy Ba (2017). *Adam: A Method for Stochastic Optimization*. arXiv: 1412.6980 [cs.LG] (cit. on p. 91).

Kirk, Hannah Rose, Bertie Vidgen, Paul Röttger, and Scott A Hale (2023). "Personalisation within bounds: A risk taxonomy and policy framework for the alignment of large language models with personalised feedback". In: *arXiv preprint* (cit. on pp. 23, 63).

Kirk, Robert, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel (Jan. 2023). "A Survey of Zero-shot Generalisation in Deep Reinforcement Learning". In: *Journal of Artificial Intelligence Research* 76, pp. 201–264. URL: http://dx.doi.org/10.1613/jair.1.14174 (cit. on pp. 3, 19, 41).

Kirkpatrick, James, Razvan Pascanu, Neil C. Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A. Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, Demis Hassabis, Claudia Clopath, Dharshan Kumaran, and Raia Hadsell (2017). "Overcoming catastrophic forgetting in neural networks". In: *Proceedings of the National Academy of Sciences* 114, pp. 3521–3526 (cit. on pp. xii, 20, 33, 35, 119, 129).

Kober, Jens, J Andrew Bagnell, and Jan Peters (2013). "Reinforcement learning in robotics: A survey". In: *The International Journal of Robotics Research* 32.11, pp. 1238–1274 (cit. on p. 17).

Koloskova, Anastasia, Tao Lin, Sebastian U Stich, and Martin Jaggi (2019). "Decentralized deep learning with arbitrary communication compression". In: *arXiv preprint arXiv:1907.09356* (cit. on p. 6).

Köpf, Andreas, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, et al. (2023). "OpenAssistant Conversations–Democratizing Large Language Model Alignment". In: *arXiv preprint* (cit. on p. 70).

Kovač, Grgur, Masataka Sawayama, Rémy Portelas, Cédric Colas, Peter Ford Dominey, and Pierre-Yves Oudeyer (2023). "Large Language Models as Superpositions of Cultural Perspectives". In: *arXiv preprint* (cit. on p. 63).

Kullback, Solomon and Richard A Leibler (1951). "On information and sufficiency". In: *The annals of mathematical statistics* 22.1, pp. 79–86 (cit. on p. 16).

Kumar, Saurabh, Aviral Kumar, Sergey Levine, and Chelsea Finn (2020). "One Solution is Not All You Need: Few-Shot Extrapolation via Structured MaxEnt RL". In: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*. Ed. by Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin. URL: https://proceedings.neurips.cc/paper/2020/hash/5d151d1059a6281335a10732fc49620e-Abstract.html (cit. on pp. 26, 102, 104).

Kurin, Vitaly, Maximilian Igl, Tim Rocktäschel, Wendelin Boehmer, and Shimon Whiteson (2021). *My Body is a Cage: the Role of Morphology in Graph-Based Incompatible Control*. arXiv: 2010.01856 [cs.LG] (cit. on pp. 21, 41, 43, 45, 46, 48).

Lakshminarayanan, Balaji, Alexander Pritzel, and Charles Blundell (2017). *Simple and Scalable Predictive Uncertainty Estimation using Deep Ensembles*. arXiv: 1612.01474 [stat.ML] (cit. on pp. 73, 87, 88).

Lambert, Nathan, Lewis Tunstall, Nazneen Rajani, and Tristan Thrush (2023). *HuggingFace H4 Stack Exchange Preference Dataset*. URL: https://huggingface.co/datasets/HuggingFaceH4/stack-exchange-preferences (cit. on p. 70).

Lange, Matthias De, Rahaf Aljundi, Marc Masana, Sarah Parisot, Xu Jia, Alevs Leonardis, Gregory G. Slabaugh, and Tinne Tuytelaars (2021). "A continual learning survey: Defying forgetting in classification tasks." In: *IEEE transactions on pattern analysis and machine intelligence* PP (cit. on p. 29).

Lange, Sascha, Thomas Gabel, and Martin Riedmiller (2012). "Batch reinforcement learning". In: *Reinforcement learning: State-of-the-art*. Springer, pp. 45–73 (cit. on p. 18).

Lazareva, Olga F and Edward A Wasserman (2010). "Nonverbal transitive inference: Effects of task and awareness on human performance". In: *Behavioural Processes* 83.1, pp. 99–112 (cit. on p. 146).

Lazaric, Alessandro (2012). "Transfer in reinforcement learning: a framework and a survey". In: *Reinforcement Learning*. Springer, pp. 143–173 (cit. on p. 24).

Lee, Kimin, Hao Liu, Moonkyung Ryu, Olivia Watkins, Yuqing Du, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, and Shixiang Shane Gu (2023). "Aligning Text-to-Image Models using Human Feedback". In: *arXiv preprint* (cit. on p. 74).

Lee, Soochan, Junsoo Ha, Dongsu Zhang, and Gunhee Kim (2020). "A neural dirichlet process mixture model for task-free continual learning". In: *arXiv preprint arXiv:2001.00689* (cit. on p. 35).

Levine, Sergey, Aviral Kumar, George Tucker, and Justin Fu (2020). *Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems*. arXiv: 2005.01643 [cs.LG] (cit. on pp. v, 3, 14, 18).

Li, Kaiwen, Tao Zhang, and Rui Wang (2020). "Deep reinforcement learning for multiobjective optimization". In: *IEEE-T-CYBERNETICS* (cit. on pp. 22, 64, 66).

Li, Mu, David G Andersen, Jun Woo Park, Alexander J Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J Shekita, and Bor-Yiing Su (2014). "Scaling distributed machine learning with the parameter server". In: *11th USENIX Symposium on operating systems design and implementation (OSDI 14)*, pp. 583–598 (cit. on p. 150).

Li, Zhizhong and Derek Hoiem (2018). "Learning without Forgetting". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 40, pp. 2935–2947 (cit. on pp. 35, 119).

Lin, Chin-Yew and Eduard Hovy (2003). "Automatic evaluation of summaries using n-gram co-occurrence statistics". In: *NAACL* (cit. on pp. 63, 72).

Lin, Tsung-Yi, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick (2014). "Microsoft coco: Common objects in context". In: *ECCV* (cit. on pp. 71, 72).

Littman, Michael L (1994). "Markov games as a framework for multi-agent reinforcement learning". In: *Machine learning proceedings 1994*. Elsevier, pp. 157–163 (cit. on p. 13).

Liu, Boyi, Lujia Wang, and Ming Liu (2019). "Lifelong federated reinforcement learning: a learning architecture for navigation in cloud robotic systems". In: *IEEE Robotics and Automation Letters* 4.4, pp. 4555–4562 (cit. on p. 150).

Liu, Evan Zheran, Aditi Raghunathan, Percy Liang, and Chelsea Finn (2021). "Decoupling Exploration and Exploitation for Meta-Reinforcement Learning without Sacrifices". In: *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*. Ed. by Marina Meila and

Tong Zhang. Vol. 139. Proceedings of Machine Learning Research. PMLR, pp. 6925–6935. URL: http://proceedings.mlr.press/v139/liu21s.html (cit. on p. 24).

Liu, Jiashuo, Zheyan Shen, Yue He, Xingxuan Zhang, Renzhe Xu, Han Yu, and Peng Cui (2021). "Towards out-of-distribution generalization: A survey". In: *arXiv preprint arXiv:2108.13624* (cit. on p. 20).

Liu, Yuqiao, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Kay Chen Tan (2021). "A survey on evolutionary neural architecture search". In: *IEEE transactions on neural networks and learning systems* (cit. on p. 152).

Liu, Ze, Han Hu, Yutong Lin, Zhuliang Yao, Zhenda Xie, Yixuan Wei, Jia Ning, Yue Cao, Zheng Zhang, Li Dong, Furu Wei, and Baining Guo (2022). "Swin Transformer V2: Scaling Up Capacity and Resolution". In: *CVPR* (cit. on p. 72).

Long, Weifan, Taixian Hou, Xiaoyi Wei, Shichao Yan, Peng Zhai, and Lihua Zhang (2023). "A Survey on Population-Based Deep Reinforcement Learning". In: *Mathematics* 11.10, p. 2234 (cit. on p. 83).

Lopez-Paz, David, Diane Bouchacourt, Levent Sagun, and Nicolas Usunier (2022). "Measuring and signing fairness as performance under multiple stakeholder distributions". In: *arXiv preprint* (cit. on p. 63).

Lopez-Paz, David and Marc'Aurelio Ranzato (2022). *Gradient Episodic Memory for Continual Learning*. arXiv: 1706.08840 [cs.LG] (cit. on p. 32).

Lu, Kevin, Aditya Grover, P. Abbeel, and Igor Mordatch (2021). "Reset-Free Lifelong Learning with Skill-Space Planning". In: *ArXiv* abs/2012.03548 (cit. on p. 35).

Makoviychuk, Viktor, Lukasz Wawrzyniak, Yunrong Guo, Michelle Lu, Kier Storey, Miles Macklin, David Hoeller, Nikita Rudin, Arthur Allshire, Ankur Handa, et al. (2021). "Isaac gym: High performance gpu-based physics simulation for robot learning". In: *arXiv preprint arXiv:2108.10470* (cit. on p. 14).

Mallya, Arun and Svetlana Lazebnik (2018). "Packnet: Adding multiple tasks to a single network by iterative pruning". In: *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 7765–7773 (cit. on pp. xii, 35, 94, 119, 129).

Mankowitz, Daniel Jaymin, Augustin Zídek, André Barreto, Dan Horgan, Matteo Hessel, John Quan, Junhyuk Oh, H. V. Hasselt, David Silver, and Tom Schaul (2018). "Unicorn: Continual Learning with a Universal, Off-policy Agent". In: *ArXiv* abs/1802.08294 (cit. on p. 35).

McCloskey, Michael and Neal J Cohen (1989). "Catastrophic interference in connectionist networks: The sequential learning problem". In: *Psychology of learning and motivation*. Vol. 24. Elsevier, pp. 109–165 (cit. on p. 34).

Michaud, Eric J, Adam Gleave, and Stuart Russell (2020). "Understanding learned reward functions". In: *arXiv preprint* (cit. on p. 63).

Mnih, Volodymyr, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Tim Harley, Timothy P. Lillicrap, David Silver, and Koray Kavukcuoglu (2016). "Asynchronous Methods for Deep Reinforcement Learning". In: *Proceedings of the 33rd International Conference on International Conference on Machine Learning - Volume 48*. ICML'16. New York, NY, USA: JMLR.org, pp. 1928–1937 (cit. on pp. vii, 14).

Mnih, Volodymyr, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller (2013). *Playing Atari with Deep Reinforcement Learning*. arXiv: 1312.5602 [cs.LG] (cit. on p. 17).

Monier, Louis, Jakub Kmec, Alexandre Laterre, Thomas Pierrot, Valentin Courgeau, Olivier Sigaud, and Karim Beguir (2020). *Offline Reinforcement Learning Hands-On*. arXiv: 2011.14379 [cs.LG] (cit. on p. 19).

Mundt, Martin, Yongjun Hong, Iuliia Pliushch, and Visvanathan Ramesh (2020). "A Wholistic View of Continual Learning with Deep Neural Networks: Forgotten Lessons and the Bridge to Active and Open World Learning". In: *ArXiv* abs/2009.01797 (cit. on p. 29).

Murray, Naila, Luca Marchesotti, and Florent Perronnin (2012). "AVA: A large-scale database for aesthetic visual analysis". In: *CVPR* (cit. on p. 74).

Nadal, Marcos and Anjan Chatterjee (2019). "Neuroaesthetics and art's diversity and universality". In: *Wiley Interdisciplinary Reviews: Cognitive Science* (cit. on p. 63).

Nagabandi, Anusha, Chelsea Finn, and Sergey Levine (2018). "Deep online learning via meta-learning: Continual adaptation for model-based RL". In: *arXiv preprint arXiv:1812.07671* (cit. on p. 34).

Nekoei, Hadi, Akilesh Badrinaaraayanan, Aaron Courville, and Sarath Chandar (2021). "Continuous coordination as a realistic scenario for lifelong learning". In: *International Conference on Machine Learning*. PMLR, pp. 8016–8024 (cit. on p. 29).

Ng, Andrew Y, Daishi Harada, and Stuart Russell (1999). "Policy invariance under reward transformations: Theory and application to reward shaping". In: *ICML* (cit. on p. 77).

Ngo, Richard, Lawrence Chan, and Soren Mindermann (2022). "The alignment problem from a deep learning perspective". In: *arXiv preprint* (cit. on p. 61).

Nichol, Alex, Joshua Achiam, and John Schulman (2018). *On First-Order Meta-Learning Algorithms*. arXiv: 1803.02999 [cs.LG] (cit. on p. 25).

OpenAI (2023). "GPT-4 Technical Report". In: *arXiv preprint* (cit. on p. 62).

Oquab, Maxime, Leon Bottou, Ivan Laptev, and Josef Sivic (2014). "Learning and transferring mid-level image representations using convolutional neural networks". In: *CVPR* (cit. on p. 61).

Osa, Takayuki, Voot Tangkaratt, and Masashi Sugiyama (2021). *Discovering Diverse Solutions in Deep Reinforcement Learning*. arXiv: 2103.07084 [stat.ML] (cit. on pp. 26, 102, 104).

Ouyang, Long, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul Christiano, Jan Leike, and Ryan Lowe (2022). *Training language models to follow instructions with human feedback*. arXiv: 2203.02155 [cs.CL] (cit. on pp. 15, 63).

Ovadya, Aviv (2023). "Generative CI through Collective Response Systems". In: *arXiv preprint* (cit. on p. 63).

Pan, Alexander, Kush Bhatia, and Jacob Steinhardt (2022). "The Effects of Reward Misspecification: Mapping and Mitigating Misaligned Models". In: *ICLR* (cit. on p. 63).

Pan, Alexander, Chan Jun Shern, Andy Zou, Nathaniel Li, Steven Basart, Thomas Woodside, Jonathan Ng, Hanlin Zhang, Scott Emmons, and Dan Hendrycks (2023). "Do the Rewards Justify the Means? Measuring Trade-Offs Between Rewards and Ethical Behavior in the MACHIAVELLI Benchmark". In: *ICML* (cit. on p. 63).

Papineni, Kishore, Salim Roukos, Todd Ward, and Wei-Jing Zhu (2002). "Bleu: a method for automatic evaluation of machine translation". In: *ACL* (cit. on pp. 63, 72).

Pareto, Vilfredo (1964). *Cours d'économie politique*. Librairie Droz (cit. on pp. 23, 64).

Parisi, German Ignacio, Ronald Kemker, Jose L. Part, Christopher Kanan, and Stefan Wermter (2019). "Continual Lifelong Learning with Neural Networks: A Review". In: *Neural networks : the official journal of the International Neural Network Society* 113, pp. 54–71 (cit. on pp. 29, 34).

Parker-Holder, Jack, Aldo Pacchiano, Krzysztof M Choromanski, and Stephen J Roberts (2020). "Effective diversity in population based reinforcement learning". In: *Advances in Neural Information Processing Systems* 33, pp. 18050–18062 (cit. on p. 28).

Pasunuru, Ramakanth and Mohit Bansal (2019). "Continual and Multi-Task Architecture Search". In: *ArXiv* abs/1906.05226 (cit. on p. 35).

Paszke, Adam, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala (2019). "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc,

E. Fox, and R. Garnett. Curran Associates, Inc., pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf (cit. on p. 92).

Peng, Xue Bin, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel (2017). "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *arXiv e-prints*, arXiv:1710.06537, arXiv:1710.06537. arXiv: 1710.06537 [cs.RO] (cit. on p. 19).

Perez, Christian F., Felipe Petroski Such, and Theofanis Karaletsos (2020). *Generalized Hidden Parameter MDPs Transferable Model-based RL in a Handful of Trials*. arXiv: 2002.03072 [cs.LG] (cit. on p. 20).

Pierrot, Thomas, Valentin Macé, Felix Chalumeau, Arthur Flajolet, Geoffrey Cideron, Karim Beguir, Antoine Cully, Olivier Sigaud, and Nicolas Perrin-Gilbert (2022). "Diversity policy gradient for sample efficient quality-diversity optimization". In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1075–1083 (cit. on p. 27).

Pinto, André Susano, Alexander Kolesnikov, Yuge Shi, Lucas Beyer, and Xiaohua Zhai (2023). "Tuning computer vision models with task rewards". In: *arXiv preprint* (cit. on pp. 62, 70).

Platanios, Emmanouil Antonios, Abulhair Saparov, and Tom Mitchell (2020). *Jelly Bean World: A Testbed for Never-Ending Learning*. arXiv: 2002.06306 [cs.LG] (cit. on p. 29).

Poirot, Maarten G, Praneeth Vepakomma, Ken Chang, Jayashree Kalpathy-Cramer, Rajiv Gupta, and Ramesh Raskar (2019). "Split learning for collaborative deep learning in healthcare". In: *arXiv preprint arXiv:1912.12115* (cit. on p. 150).

Pomerleau, Dean A (1988). "Alvinn: An autonomous land vehicle in a neural network". In: *Advances in neural information processing systems* 1 (cit. on p. 18).

Powers, Sam, Eliot Xing, Eric Kolve, Roozbeh Mottaghi, and Abhinav Gupta (2022). "Cora: Benchmarks, baselines, and metrics as a platform for continual reinforcement learning agents". In: *Conference on Lifelong Learning Agents*. PMLR, pp. 705–743 (cit. on p. 29).

Powers, Sam, Eliot Xing, Eric Kolve, Roozbeh Mottaghi, and Abhinav Kumar Gupta (2021). "CORA: Benchmarks, Baselines, and Metrics as a Platform for Continual Reinforcement Learning Agents". In: *ArXiv* abs/2110.10067 (cit. on p. 34).

Pugh, Justin K, Lisa B Soros, and Kenneth O Stanley (2016). "Quality diversity: A new frontier for evolutionary computation". In: *Frontiers in Robotics and AI* 3, p. 40 (cit. on p. 27).

Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever (2018). "Improving Language Understanding by Generative Pre-Training". In: (cit. on p. 62).

Ramé, Alexandre, Guillaume Couairon, Mustafa Shukor, Corentin Dancette, Jean-Baptiste Gaya, Laure Soulier, and Matthieu Cord (2023). *Rewarded soups: towards Pareto-optimal alignment by interpolating weights fine-tuned on diverse rewards*. arXiv: 2306.04488 [cs.LG] (cit. on pp. x, 61).

Ratadiya, Pratik, Khushi Asawa, and Omkar Nikhal (2020). "A decentralized aggregation mechanism for training deep learning models using smart contract system for bank loan prediction". In: *arXiv preprint arXiv:2011.10981* (cit. on p. 150).

Reddy, Gautam, Antonio Celani, Terrence J Sejnowski, and Massimo Vergassola (2016). "Learning to soar in turbulent environments". In: *Proceedings of the National Academy of Sciences* 113.33, E4877–E4884 (cit. on p. 13).

Reed, Scott, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas (2022). *A Generalist Agent*. arXiv: 2205.06175 [cs.AI] (cit. on pp. ix, 7, 41, 47).

Ren, Hang, Aivar Sootla, Taher Jafferjee, Junxiao Shen, Jun Wang, and Haitham Bou-Ammar (2022). "Reinforcement Learning in Presence of Discrete Markovian Context Evolution". In: *arXiv preprint arXiv:2202.06557* (cit. on p. 34).

Ren, Pengzhen, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Xiaojiang Chen, and Xin Wang (2021a). "A comprehensive survey of neural architecture search: Challenges and solutions". In: *ACM Computing Surveys (CSUR)* 54.4, pp. 1–34 (cit. on pp. 6, 152).

Ren, Pengzhen, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang (2021b). "A survey of deep active learning". In: *ACM computing surveys (CSUR)* 54.9, pp. 1–40 (cit. on pp. 6, 151, 152).

Rennie, Steven J, Etienne Marcheret, Youssef Mroueh, Jerret Ross, and Vaibhava Goel (2017). "Self-critical sequence training for image captioning". In: *CVPR* (cit. on pp. 62, 70, 72).

Robbins, Herbert and Sutton Monro (1951). "A stochastic approximation method". In: *The annals of mathematical statistics*, pp. 400–407 (cit. on p. 91).

Roberts, Jonathan, Timo Lüddecke, Sowmen Das, Kai Han, and Samuel Albanie (2023). "GPT4GEO: How a Language Model Sees the World's Geography". In: *arXiv preprint arXiv:2306.00020* (cit. on p. 146).

Roberts, William A and Maria T Phelps (1994). "Transitive inference in rats: A test of the spatial coding hypothesis". In: *Psychological Science* 5.6, pp. 368–374 (cit. on p. 146).

Rolnick, David, Arun Ahuja, Jonathan Schwarz, Timothy P. Lillicrap, and Greg Wayne (2019). "Experience Replay for Continual Learning". In: *NeurIPS* (cit. on pp. 125, 131).

Rombach, Robin, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer (2022). "High-resolution image synthesis with latent diffusion models". In: *CVPR* (cit. on p. 73).

Rumelhart, David E, Geoffrey E Hinton, and Ronald J Williams (1986). "Learning representations by back-propagating errors". In: *nature* 323.6088, pp. 533–536 (cit. on p. 13).

Rusu, Andrei A., Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell (2016a). "Policy Distillation". In: *CoRR* abs/1511.06295 (cit. on pp. 35, 119).

Rusu, Andrei A., Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell (2016b). "Progressive Neural Networks". In: *ArXiv* abs/1606.04671 (cit. on pp. xii, 35, 129).

Sainz, Oscar, Jon Ander Campos, Iker García-Ferrero, Julen Etxaniz, Oier Lopez de Lacalle, and Eneko Agirre (2023). "Nlp evaluation in trouble: On the need to measure llm data contamination for each benchmark". In: *arXiv preprint arXiv:2310.18018* (cit. on p. 146).

Santara, Anirban, Abhishek Naik, Balaraman Ravindran, Dipankar Das, Dheevatsa Mudigere, Sasikanth Avancha, and Bharat Kaul (2017). "Rail: Risk-averse imitation learning". In: *arXiv preprint arXiv:1707.06658* (cit. on p. 18).

Santurkar, Shibani, Esin Durmus, Faisal Ladhak, Cinoo Lee, Percy Liang, and Tatsunori Hashimoto (2023). "Whose opinions do language models reflect?" In: *arXiv preprint* (cit. on p. 63).

Schulman, John, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz (2015). "Trust region policy optimization". In: *International conference on machine learning*. PMLR, pp. 1889–1897 (cit. on p. 16).

Schulman, John, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel (2018). *High-Dimensional Continuous Control Using Generalized Advantage Estimation*. arXiv: 1506.02438 [cs.LG] (cit. on p. 16).

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017a). *Proximal Policy Optimization Algorithms*. arXiv: 1707.06347 [cs.LG] (cit. on pp. vii, viii, 14, 16).

Schulman, John, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov (2017b). *Proximal Policy Optimization Algorithms*. arXiv: 1707.06347 [cs.LG] (cit. on p. 78).

Schwarz, Jonathan, Wojciech M. Czarnecki, Jelena Luketina, Agnieszka Grabska-Barwinska, Yee Whye Teh, Razvan Pascanu, and Raia Hadsell (2018). "Progress & Compress: A scalable framework for continual learning". In: *ArXiv* abs/1805.06370 (cit. on pp. 35, 119).

Schwarz, Jonathan, Siddhant Jayakumar, Razvan Pascanu, Peter E Latham, and Yee Teh (2021). "Powerpropagation: A sparsity inducing weight reparameterisation". In: *Advances in Neural Information Processing Systems*. Ed. by M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan. Vol. 34. Curran Associates, Inc., pp. 28889–28903. URL: https://proceedings.neurips.cc/paper/2021/file/f1e709e6aef16ba2f0cd6c7e4f52b9b6-Paper.pdf (cit. on p. 35).

Settles, Burr (2009). "Active learning literature survey". In: (cit. on p. 93).

Shannon, Claude Elwood (1948). "A mathematical theory of communication". In: *The Bell system technical journal* 27.3, pp. 379–423 (cit. on p. 18).

Shukor, Mustafa, Corentin Dancette, Alexandre Rame, and Matthieu Cord (2023). "UnIVAL: Unified Model for Image, Video, Audio and Language Tasks". In: *Transactions on Machine Learning Research Journal* (cit. on p. 75).

Siebenborn, Max, Boris Belousov, Junning Huang, and Jan Peters (2022). *How Crucial is Transformer in Decision Transformer?* arXiv: 2211.14655 [cs.LG] (cit. on p. 19).

Silver, David, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas baker, Matthew Lai, Adrian Bolton, Yutian Chen, Timothy P. Lillicrap, Fan Hui, L. Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis (2017). "Mastering the game of Go without human knowledge". In: *Nature* 550, pp. 354–359 (cit. on pp. v, 3).

Sodhani, Shagun, Franziska Meier, Joelle Pineau, and Amy Zhang (2021). "Block Contextual MDPs for Continual Learning". In: *arXiv preprint arXiv:2110.06972* (cit. on p. 35).

Stiennon, Nisan, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano (2020). "Learning to summarize with human feedback". In: *NeurIPS* (cit. on pp. 64, 70).

Sun, Yuwei, Hideya Ochiai, and Hiroshi Esaki (2021). "Decentralized deep learning for multi-access edge computing: A survey on communication efficiency and trustworthiness". In: *IEEE Transactions on Artificial Intelligence* 3.6, pp. 963–972 (cit. on pp. 150, 151).

Sutton, Richard S (1990). "Integrated architectures for learning, planning, and reacting based on approximating dynamic programming". In: *Machine learning proceedings 1990*. Elsevier, pp. 216–224 (cit. on p. 13).

Sutton, Richard S. and Andrew G. Barto (1998). *Reinforcement Learning: An Introduction*. Second. The MIT Press. URL: http://incompleteideas.net/book/the-book-2nd.html (cit. on pp. 11–14).

Sutton, Richard S., David A. McAllester, Satinder P. Singh, and Yishay Mansour (2000). "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *NIPS* (cit. on p. 15).

Taori, Rohan, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto (2023). *Stanford Alpaca: An Instruction-following LLaMA model*. https://github.com/tatsu-lab/stanford_alpaca (cit. on pp. 62, 64, 70).

Taylor, Matthew E. and Peter Stone (Dec. 2009). "Transfer Learning for Reinforcement Learning Domains: A Survey". In: *J. Mach. Learn. Res.* 10, pp. 1633–1685 (cit. on p. 24).

Teh, Yee Whye, Victor Bapst, Wojciech Marian Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu (2017). "Distral: Robust Multitask Reinforcement Learning". In: *CoRR* abs/1707.04175. arXiv: 1707.04175. URL: http://arxiv.org/abs/1707.04175 (cit. on p. 24).

Tessler, Chen, Shahar Givony, Tom Zahavy, Daniel Mankowitz, and Shie Mannor (2017). "A deep hierarchical approach to lifelong learning in minecraft". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 31. 1 (cit. on p. 34).

Todorov, Emanuel, Tom Erez, and Yuval Tassa (2012). "MuJoCo: A physics engine for model-based control". In: *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, pp. 5026–5033 (cit. on pp. 15, 49, 77).

Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample (2023a). "LLaMA: Open and Efficient Foundation Language Models". In: *arXiv preprint arXiv:2302.13971* (cit. on pp. 64, 69, 147).

Touvron, Hugo, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom (2023b). *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv: 2307.09288 [cs.CL] (cit. on p. 148).

Trabucco, Brandon, Mariano Phielipp, and Glen Berseth (2022). "AnyMorph: Learning Transferable Polices By Inferring Agent Morphology". In: *Proceed-*

ings of the 39th International Conference on Machine Learning. Ed. by Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato. Vol. 162. Proceedings of Machine Learning Research. PMLR, pp. 21677–21691. URL: https://proceedings.mlr.press/v162/trabucco22b.html (cit. on p. 21).

Tran, Anh-Tu, The-Dung Luong, Jessada Karnjana, and Van-Nam Huynh (2021). "An efficient approach for privacy preserving decentralized deep learning models based on secure multi-party computation". In: *Neurocomputing* 422, pp. 245–262 (cit. on p. 150).

Traoré, Kalifou René, Hugo Caselles-Dupré, Timothée Lesort, Te Sun, Guanghang Cai, Natalia Díaz Rodríguez, and David Filliat (2019). "DisCoRL: Continual Reinforcement Learning via Policy Distillation". In: *ArXiv* abs/1907.05855 (cit. on p. 119).

Vamplew, Peter, Richard Dazeley, Cameron Foale, Sally Firmin, and Jane Mummery (2018). "Human-aligned artificial intelligence is a multiobjective problem". In: *Ethics and Information Technology* (cit. on p. 64).

Vamplew, Peter, John Yearwood, Richard Dazeley, and Adam Berry (2008). "On the limitations of scalarisation for multi-objective reinforcement learning of pareto fronts". In: *AJCAIA* (cit. on p. 66).

Vapnik, Vladimir N (1999). "An overview of statistical learning theory". In: *TNN* (cit. on p. 61).

Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin (2017). "Attention is All you Need". In: *NeurIPS* (cit. on pp. v, 3, 65).

Vedantam, Ramakrishna, C Lawrence Zitnick, and Devi Parikh (2015). "Consensus-based image description evaluation". In: *ICCV* (cit. on pp. 63, 72).

Veniat, Tom, Ludovic Denoyer, and Marc'Aurelio Ranzato (2020). "Efficient continual learning with modular networks and task-driven priors". In: *arXiv preprint arXiv:2012.12631* (cit. on pp. 95, 119).

Völske, Michael, Martin Potthast, Shahbaz Syed, and Benno Stein (2017). "Tl; dr: Mining reddit to learn automatic summarization". In: *ACL Workshop* (cit. on p. 70).

Von Fersen, Lorenzo, Clive D Wynne, Juan D Delius, and John E Staddon (1991). "Transitive inference formation in pigeons." In: *Journal of Experimental Psychology: Animal Behavior Processes* 17.3, p. 334 (cit. on p. 146).

Wang, Guanzhi, Yuqi Xie, Yunfan Jiang, Ajay Mandlekar, Chaowei Xiao, Yuke Zhu, Linxi Fan, and Anima Anandkumar (2023). "Voyager: An open-ended embodied agent with large language models". In: *arXiv preprint arXiv:2305.16291* (cit. on pp. v, 3).

Wang, Peng, An Yang, Rui Men, Junyang Lin, Shuai Bai, Zhikang Li, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang (2022). "OFA: Unifying Architectures, Tasks, and Modalities Through a Simple Sequence-to-Sequence Learning Framework". In: *CoRR* (cit. on pp. 75, 77).

Wang, Tingwu, Renjie Liao, Jimmy Ba, and Sanja Fidler (2018). "NerveNet: Learning Structured Policy with Graph Neural Networks". In: *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net. URL: https://openreview.net/forum?id=S1sqHMZCb (cit. on pp. 21, 41).

Wang, Yizhong, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi (2022). "Self-Instruct: Aligning Language Model with Self Generated Instructions". In: *arXiv preprint* (cit. on p. 70).

Wang, Zhechao, Qiming Fu, Jianping Chen, Yunzhe Wang, You Lu, and Hongjie Wu (2023). *Reinforcement Learning in Few-Shot Scenarios: A Survey*. URL: https://doi.org/10.1007/s10723-023-09663-0 (cit. on p. 24).

Watkins, Christopher JCH and Peter Dayan (1992). "Q-learning". In: *Machine learning* 8, pp. 279–292 (cit. on p. 13).

Werra, Leandro von, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, and Nathan Lambert (2020). *TRL: Transformer Reinforcement Learning*. https://github.com/lvwerra/trl (cit. on p. 70).

Williams, Ronald J (1992). "Simple statistical gradient-following algorithms for connectionist reinforcement learning". In: *Reinforcement learning* (cit. on p. 72).

Wilson, Aaron, Alan Fern, Soumya Ray, and Prasad Tadepalli (2007). "Multi-Task Reinforcement Learning: A Hierarchical Bayesian Approach". In: *Proceedings of the 24th International Conference on Machine Learning*. ICML '07. Corvalis, Oregon, USA: Association for Computing Machinery, pp. 1015–1022. URL: https://doi.org/10.1145/1273496.1273624 (cit. on p. 24).

Wołczyk, Maciej, Michal Zajkac, Razvan Pascanu, Lukasz Kuci'nski, and Piotr Milo's (2021). "Continual World: A Robotic Benchmark For Continual Reinforcement Learning". In: *ArXiv* abs/2105.10919 (cit. on pp. xii, 29, 31, 32, 34, 35, 127, 128, 133).

Wolf, Thomas, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush (2020). "Transformers: State-of-the-Art Natural Language Processing". In: *EMNLP* (cit. on pp. 69, 70).

Won, Jungdam, Deepak Gopinath, and Jessica Hodgins (2020). "A scalable approach to control diverse behaviors for physically simulated characters". In: *TOG* (cit. on p. 23).

Wortsman, Mitchell, Maxwell Horton, Carlos Guestrin, Ali Farhadi, and Mohammad Rastegari (2021). "Learning Neural Network Subspaces". In: *ArXiv* abs/2102.10472 (cit. on pp. x, 88–90, 101, 111, 126).

Wortsman, Mitchell, Gabriel Ilharco, Samir Yitzhak Gadre, Rebecca Roelofs, Raphael Gontijo-Lopes, Ari S. Morcos, Hongseok Namkoong, Ali Farhadi, Yair Carmon, Simon Kornblith, and Ludwig Schmidt (2022). "Model soups: averaging weights of multiple fine-tuned models improves accuracy without increasing inference time". In: *ICML* (cit. on p. 65).

Wortsman, Mitchell, Vivek Ramanujan, Rosanne Liu, Aniruddha Kembhavi, Mohammad Rastegari, Jason Yosinski, and Ali Farhadi (2020). "Supermasks in Superposition". In: *ArXiv* abs/2006.14769 (cit. on pp. 34, 119).

Wu, Shuang, Jian Yao, Haobo Fu, Ye Tian, Chao Qian, Yaodong Yang, QIANG FU, and Yang Wei (2023). "Quality-Similar Diversity via Population Based Reinforcement Learning". In: *The Eleventh International Conference on Learning Representations*. URL: https://openreview.net/forum?id=bLmSMXbqXr (cit. on p. 28).

Wu, Zeqiu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, and Hannaneh Hajishirzi (2023). "Fine-Grained Human Feedback Gives Better Rewards for Language Model Training". In: *NeuriPS* (cit. on pp. 64, 66).

Xie, Annie, James Harrison, and Chelsea Finn (2020). "Deep reinforcement learning amidst lifelong non-stationarity". In: *arXiv preprint arXiv:2006.10701* (cit. on p. 34).

Xie, Enze, Lewei Yao, Han Shi, Zhili Liu, Daquan Zhou, Zhaoqiang Liu, Jiawei Li, and Zhenguo Li (2023). "DiffFit: Unlocking Transferability of Large Diffusion Models via Simple Parameter-Efficient Fine-Tuning". In: *arXiv preprint arXiv:2304.06648* (cit. on p. 74).

Xu, Jiazheng, Xiao Liu, Yuchen Wu, Yuxuan Tong, Qinkai Li, Ming Ding, Jie Tang, and Yuxiao Dong (2023). "ImageReward: Learning and Evaluating Human Preferences for Text-to-Image Generation". In: *arXiv preprint* (cit. on pp. 63, 73).

Xu, Ju and Zhanxing Zhu (2018). "Reinforced Continual Learning". In: *NeurIPS* (cit. on p. 35).

Xu, Mengdi, Wenhao Ding, Jiacheng Zhu, Zuxin Liu, Baiming Chen, and Ding Zhao (2020). "Task-agnostic online reinforcement learning with an infinite mixture of gaussian processes". In: *Advances in Neural Information Processing Systems* 33, pp. 6429–6440 (cit. on p. 34).

Yang, Chuanyu, Kai Yuan, Qiuguo Zhu, Wanming Yu, and Zhibin Li (2020). "Multi-expert learning of adaptive legged locomotion". In: *Science Robotics* (cit. on p. 23).

Yang, Runzhe, Xingyuan Sun, and Karthik Narasimhan (2019). "A Generalized Algorithm for Multi-Objective Reinforcement Learning and Policy Adaptation". In: *NeurIPS* (cit. on p. 23).

Yen, Gary G and Zhenan He (2013). "Performance metric ensemble for multiobjective evolutionary algorithms". In: *TEVC* (cit. on pp. 70, 72).

Yu, Licheng, Patrick Poirson, Shan Yang, Alexander C Berg, and Tamara L Berg (2016). "Modeling context in referring expressions". In: *ECCV* (cit. on p. 75).

Zhang, Shu, Xinyi Yang, Yihao Feng, Can Qin, Chia-Chih Chen, Ning Yu, Zeyuan Chen, Huan Wang, Silvio Savarese, Stefano Ermon, et al. (2023). "HIVE: Harnessing Human Feedback for Instructional Visual Editing". In: *arXiv preprint* (cit. on p. 62).

Zhao, Wenshuai, Jorge Peña Queralta, and Tomi Westerlund (2020). "Sim-to-real transfer in deep reinforcement learning for robotics: a survey". In: *2020 IEEE symposium series on computational intelligence (SSCI)*. IEEE, pp. 737–744 (cit. on pp. 19, 21).

Zhou, Kun, Yutao Zhu, Zhipeng Chen, Wentong Chen, Wayne Xin Zhao, Xu Chen, Yankai Lin, Ji-Rong Wen, and Jiawei Han (2023). "Don't Make Your LLM an Evaluation Benchmark Cheater". In: *arXiv preprint arXiv:2311.01964* (cit. on p. 146).

Zoph, Barret and Quoc V Le (2016). "Neural architecture search with reinforcement learning". In: *arXiv preprint arXiv:1611.01578* (cit. on p. 152).