A Minimalist Approach to Deep Multi-task Learning

Vitaly Kurin

Magdalen College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Michaelmas 2022

Abstract

Multi-task learning is critical for real-life applications of machine learning. Modern approaches are characterised by algorithmic complexity, often unjustified, leading to impractical solutions. In contrast, this thesis demonstrates that a minimalistic alternative is possible, showing the attractiveness of simple methods. *'In defence of the Unitary Scalarisation for Deep Multi-task Learning'* motivates the rest of the thesis, showing that none of the more complex multi-task optimisers outperforms the simple per-task gradient summation when compared on fair grounds. Furthermore, it proposes a novel look at multi-task optimisers from the regularisation standpoint. The rest of this thesis focuses on deep reinforcement learning, a general framework for sequential decision-making. In particular, we look at the setting when observations (inputs to the model) are represented as graphs, i.e., collections of interconnected nodes. In *'Scaling GNNs to High-Dimensional Continuous Control'* and *'The Role of Morphology in Graph-Based Incompatible Control'*, we learn a single control policy for agents of different morphology by representing the observation set elements as graphs and deploy graph neural networks (including transformers). In the former chapter, we devise a simple method to scale graph networks by freezing some parts of the network to stabilise learning and prevent overfitting. In the latter chapter, we show that graph connectivity might be suboptimal for the downstream task demonstrating that less-constrained transformers perform significantly better without having the graph connectivity information. Finally, in the *'Generalisable Branching Heuristic for a SAT Solver'*, we apply multi-task reinforcement learning to Boolean satisfiability, a fundamental problem in academia and industrial applications. We demonstrate that Q-learning, a staple reinforcement learning algorithm equipped with graph neural networks for function approximation, can learn a generalisable branching heuristic.

We hope our findings will steer the further development of the field: creating more complex benchmarks, adding assumptions on task similarities and a model capacity, and exploring other objective functions rather than focusing on the average performance across the tasks.

Acknowledgements

First and foremost, I would like to thank my advisor, Shimon Whiteson, for guiding me through my DPhil, teaching me to clearly explain myself in writing, and fostering an independent researcher in me. Thank you, Shimon!

I am extremely grateful to Xiaowen Dong and Thomas Dietterich for carefully examining my thesis, providing constructive feedback and encouraging comments. I was a bit nervous before the viva, but I ended up having fun.

I would like to say a big thank you to Wendy Poole, our doctoral programme administrator, for doing everything possible so that I could focus solely on my research. She helped me with everything varying from booking plane tickets to providing moral support and fighting for funding for my studies. Thank you, Wendy, you are the best!

Samsung R&D Institute UK funded my studies in Oxford through the AIMS CDT programme, and I am incredibly grateful to Brian Kyungmin Song, Marwan Boustany, and all the others who organised this stipend. I greatly appreciate that funding came with no strings attached, and I had total freedom in pursuing my scientific interests throughout my DPhil.

I have been a proud member of the WhiRL lab in the Robert Hooke building next to the dinosaur museum in Oxford. I am grateful to all these fantastic people with whom we collaborated, had lunches or just shared a laugh or two: Greg Farquhar, Jakob Foerster, Supratik Paul, Maximilian Igl, Christian Schroeder de Witt, Tabish Rashid, Shangtong Zhang, Luisa Zintgraf, Anuj Mahajan, Tim Rocktaeschel, Bei Peng, Mingfei Sun, Jelena Luketina, Kristian Hartikainen, Matt Smith, Matt Fellows, Tarun Gupta, Risto Vuorio, Jacob Beck, Benjamin Ellis and Zheng Xiong. I would like to say a separate thanks to Wendelin Boehmer, who was extremely helpful to me during my first years at Oxford providing feedback and discussing research. During my time at the WhiRL lab, I had an opportunity to supervise brilliant Master's students: Jeffrey Mak and Charles Blake. I found our discussions engaging and fruitful and was constantly surprised by their drive, independence and intellect.

I made good friends throughout my doctoral studies at Oxford. I do not think I would have made this journey to the end without Alessandro De Palma and Panagiotis Tigkas. We had fascinating conversations while walking around Oxford.

We suffered together and had fun. I learnt a lot from you and hope I was able to teach you something as well. You are absolutely amazing and I will greatly miss you. I was lucky enough to collaborate with Alessandro and learned much from him. Without exaggeration, our collaboration saved me from quitting. I am extremely grateful to Oleh Stupak, my friend and neighbour in Castle Mill, who has been a source of infinite thought-provoking discussions and constant encouragement, which was so needed during the pandemic in 2020-2021. Henry Kenlay is another person who helped me a lot along the way. I am immensely thankful to him for all the discussions and infinite moral support. I will never forget our walks in Port Meadow complaining about life, research and everything else. Thank you, Henry! Our friendship means a lot to me. Finally, I am glad I met Vadim Kaushanskii during my time at Oxford, who was doing his DPhil at the Mathematical Institute at the time. I thank him for our walks around Oxford and for helping me out when I most needed it.

I have met a lot of amazing people in the last five years, and these people made this journey more enjoyable and fun: Joao Mesias and Ana Vencá, Kyriakos Shiarlis, Sudhanshu Kasewa, Bryn Elesedy, Anna Gautier, Alexander Mitchell, Rhydian Windsor, Yuki Asano, Ada Alevizaki, Siddhant Gangapurwala, Bernardo Pérez Orozco, Tim Rudner, Adam Golinski, Shuyu Lin, Kyriakos Polymenakos, Nantas Nardelli, Rodrigo Rivera Castro, Kyunghyun Cho, Cristina Pinneri, Despoina Paschalidou, David Stutz, Jack Parker-Holder, Michael Bronstein, Pasquale Minervini, Tal Kachman, and Andrey Kravchenko.

I received my first undergraduate degree in Economics in 2012, and while I have not yet found a proper application to the knowledge I got there, these four years were one of the best years of my life. Not only did I find my future wife there, but I also met a bunch of amazing people and spent four years of my life shoulder to shoulder with them. I am grateful to you all: Alisa Yakusheva, Stas Kabaev, Ilya Shahurin, Misha Ivanov, Kirill Vetrov, Misha Teplovodskii, Masha Petrunina, Alisa Kolesnikova, Fedor Chuvilkin, Maxim Sofronov, Dayana Shuptinova, Masha Romanenko, Karina Guylazyan, Kolya Schwitz, Misha Gorbachev, Anne-Marie Forget, Pasha Pankratov, Ilya Kuznetsov and Maxim Fedotov. I feel fortunate to have the last two, Ilya and Maxim, as my close friends.

Throughout my studies, I had an opportunity to intern at several terrific industrial labs and meet a lot of magnificent people. At NVIDIA, I was lucky to work in the applied deep learning team, which fostered my love of applied research. I am incredibly grateful to Bryan Catanzaro for inviting me, giving a problem to work on and mentoring me along the way. I thank Saad Godil for endless everyday discussions, constant feedback, and fostering a pragmatic researcher in me. I thank Andrew Tao and Guy Peled for being extremely helpful with the infrastructure. I

am grateful to Rafael Valle for being a true friend, providing moral support and having fun together. I thank Rajarshi Roy, Robert Kirby, Yogesh Mahajan, Alex Aiken, Mohammad Shoyebi, Sungwon Kim and Ryan Prenger, and the rest of the ADLR team for exciting discussions and just being amazing people.

At the end of 2020, I had an excellent opportunity to work at Facebook AI Research in London and met a bunch of extraordinary people there. I am indebted to Tim Rocktaeschel for unlimited support and mentoring, and I thank my colleagues there: Edward Grefenstette, Heinrich Kuettler, Roberto Calandra, Mikayel Samvelyan, and Sainbayar Sukhbaatar, who all gave invaluable feedback and helped me grow.

I was privileged enough to work at Microsoft Research in Cambridge twice: first as a visitor in 2017 and then as an intern in 2021. I would love to thank Sebastian Nowozin for inviting me, a master's student with no research track record at the time, giving me a problem to tackle and providing constant feedback. In addition, I was lucky enough to have Katja Hofmann co-supervising me in 2017, and I learnt a lot from her. I thank Ryota Tomioka, Miltos Allamanis, Tom Ellis, Diane Bouchacourt, Morgan Funtowicz, David Bignell, Martin Kukla, Manon Knoertzer, Yordan Zaykov, Botond Cseke and Alex Spengler for exciting discussions and making MSR a great place to be at. In 2021, I was fortunate to work with Ryota Tomioka and to learn from him. I am highly grateful to Ryota for mentoring and helping me grow as a researcher. I am delighted to have worked closely with Andrew Fitzgibbon, who helped me develop an engineering approach to research, which I genuinely appreciate. I am lucky to have interacted with Simon Peyton Jones, Alan Lawrence, Tom Ellis, Anna Mitenkova, Katja Hofmann, Matthew Jones, Ricky Loynd, and Jaroslaw Rzepecki throughout my internship at MSR, who provided invaluable feedback for my work and shaped me as a researcher.

I have dreamt of working at DeepMind ever since I read the DQN paper in 2014. In summer 2022, that dream came true. I interned in the Deep Learning team, closely working with Yaroslav Ganin, Aaron van den Oord, and Petar Veličković. I met so many great people there, and I am forever grateful to them for engaging discussions, advice, chess and foosball matches, and more. This is the list of these great people: Isabela Albuquerque, Jannik Kossen, Michael Figurnov, Lisa Schutt, Leo Berrada, Nate Kushman, Yujia Li, Igor Babushkin, Felix Gimeno, Ilan Price, Peter Battaglia, Nikolai Savinov, Sasha Vezhnevets, Catalin Ionescu, George Papamakarios, Matko Bosnjak, Andrew Dudzik, Charles Blundell, Kyriacos Nikiforou, Yulia Rubanova, Shakir Mohammed, Oriol Vinyals, Andrea Tachetti, Michael Shaarshmidt, Feryal Behbahani, Roman Ring, Oleg Rybkin, Evgeny Nikishin, Borja Ibarz, Razvan Pascanu, Alex Vitvitsky, Marin Vlastelica, and Robert Csordas.

I would like to thank old friends from my Master's studies at RWTH, who kept supporting me after we got scattered around the world: Valentin Belonogov, Lavinia Banu, Ilya Kostrikov, Ilia Kulikov and Lucas Beyer, who should finally acknowledge that I was the inspiration for the name of his ViT model (ViTaly Kurin).

For many people, the road to graduation is not a walk in the park, and I am among these people. It is common knowledge that doctoral students suffer from mental issues, and I am not an exception, and I thought about quitting multiple times. Psychotherapy is one of the things that helped me persevere. I am infinitely grateful to Fedor Chuvilkin, who recommended therapy, and my therapist Natalia, who guided me out of a deep rut of mental suffering and helped me feel the joy of life again. Cycling is another thing that helped me graduate. I am a proud member of the Richmond Park Velo cycling club, and I cannot thank these amazing people enough: Richard, Salvo, Marco, Anlyn, Lex, Marelise, Adam, Kuntal, Dave, Nic, Wes, Geoff, Sonia, Dev, and the rest of the club for making it such a great place. Cycling long distances helped me to build stamina, learn to turn my head off and boost my self-confidence. I will never forget cycling with Jakob Foerster, almost 400km from Manchester to London, in two days, after which I could barely sit.

I want to say a separate thanks to Alexey Donov, whom I have known for over twenty years and who has been a role model. He is the reason why I got into computer science and helped me a lot along the way. Sadly, our interactions are only virtual nowadays, but I appreciate that I can always speak to you, and you will listen. I am grateful to Alexey for teaching me constantly to ask the 'why' question and trying to get to the root of the problem. I thank Mikhail Shelemetyev for introducing me to Linux back in 2006, which bolstered my interest and curiosity in computers. I am forever indebted to people who helped me grow as a software engineer, especially Konstantin Esipov, who offered me my first programming job and helped me develop. My software engineering time was lots of fun, and it made a massive impact on me as a researcher. I had a great time working with Misha Zadorozhny, Oleg Kuzmenko, Artur Rakhmatullin, Nikita Gaevoy, Pavel Ivanov, Oksana Gorobets and Dmitry Makhotin, thank you all!

Finally, I would like to thank my family. Alla and Stanislav Romanov, thank you for believing in me and supporting me morally and financially along the way. I am incredibly grateful to my grandfather Viktor Golyshev and my grandmother Ludmila Golysheva, who, sadly, passed away during the pandemic, and whom we all miss so much, for raising me and developing a serious attitude to learning in me. I am immensely grateful to my wife Alisa and son Iurii for putting up with my schedule and sacrificing their time and energy to make it easier for me to do research. I am fortunate to have you two in my life. Thank you! Finally, Alisa was extremely helpful in training my patience and perseverance by not agreeing to go

out with me for more than a year back in 2008. I believe that these qualities and my natural stubbornness have been critical for my graduation, as well as the creativity I developed while devising witty plans to attract the interest of the woman I love.

Contents

List of Figures

Abbreviations

CNN . . . . . Convolutional Neural Network

DDPG . . . . . Deep Deterministic Policy Gradient

DQN . . . . . Deep Q-Networks

GNN . . . . . Graph Neural Network

IMTL . . . . . Impartial multi-task learning

MDP . . . . . Markov Decision Process

MGDA . . . . . Multiple-Gradient Descent Algorithm

MLP . . . . . Multilayer Perceptron

MPNN . . . . . Message Passing Neural Network

MTL . . . . . Multi-Task learning

MTRL . . . . . Multi-task Reinforcement Learning

PPO . . . . . Proximal Policy Optimisation

RL . . . . . Reinforcement Learning

RLW . . . . . Random Loss Weighting

SAC . . . . . Soft Actor-Critic

SAT . . . . . Boolean Satisfiability

SF . . . . . Snowflake

SMP . . . . . Shared Modular Policies

SMTO . . . . . Specialised Multi-Task Optimiser

TD3 . . . . . Twin Delayed Deep Deterministic policy gradient

Discard everything that does not spark joy

— Marie Kondo's *The Life-Changing Magic of Tidying*

1 Introduction

Contents

1.1 Motivation

This thesis considers Multi-Task learning (MTL), meaning the study of problems arising when dealing with multiple machine learning tasks. Typically, MTL implies training and testing a single model on a discrete fixed set of tasks. However, we also study related problems such as generalisation and transfer in this work.

There are multiple reasons to consider MTL varying from imitating nature [TSG21] to purely pragmatic rationale, such as reducing computational or memory complexity. One compelling reason to use MTL is to leverage similarities among multiple

problems to obtain better models [Car97b]. Better models are those that yield higher performance, are more sample efficient, more generalisable or more robust during training. MTL significantly impacts practical applications, where data collection comes with a cost, and a mistake might lead to disastrous consequences. In some cases, it is even impossible not to use MTL, e.g., combinatorial optimisation or mathematical reasoning tasks. Mastering one particular instance, in these cases, is not meaningful since *solving the problem* implies solving all of its instances, not memorising several cases.

More concretely, one can identify data-related reasons for pursuing MTL and computational-related reasons. Furthermore, we can split the data-related group into two scenarios. In the first scenario, tasks with less data benefit from tasks with more data. In the second, joint training on tasks where each task has only m data points achieves per-task performance equivalent to having $n > m$ training examples if the tasks were trained separately. Similarly, we can split the computational-related benefits of MTL into two subgroups. First, MTL can better exploit GPU/TPU parallelism during training. Second, one can save on memory and computation by deploying a single model that solves many problems. Finally, deploying a single model is more convenient in terms of versioning, debugging, and model serving.

Ideally, we would like to measure synergy by quantifying these benefits. For the data-related group, a methodological issue is that achieving synergy depends on the relationships among the tasks as well as the representational bottleneck in the network architecture. If the tasks have no similarities (which are also hard to define precisely), then there is no data synergy benefit. However, if we use an overly-wide bottleneck, then the network may not discover the synergy.

It seems that in deep learning, which is highly empirical, the only way to study this in practice is to train with various bottleneck widths and measure performance. Standard architecture search and hyperparameter tuning to optimize validation performance, if successful, should maximize the effects of data synergy, especially synergy of the second type if it is present. Therefore, in this dissertation, we focus

on the overall performance of MTL algorithms. The results should apply regardless of whether data synergy is present and exploitable.

Modern machine learning is characterised by ever-growing algorithmic complexity, often unjustified, leading to impractical solutions. In contrast, in this work, we show that minimalist approaches are possible, and we can reduce the complexity without sacrificing performance. In this dissertation, *minimalist* is a synonym of *simple* and serves as an inspiration for our work adhering to Occam's razor. It also implies treating the MTL problem as a single-task one, i.e. not introducing MTL-specific optimisation algorithms. From this perspective, our work is a practical demonstration of applying single-task methods to a wide range of multi-task problems, from supervised multi-task classification to applying Reinforcement Learning (RL) for Boolean satisfiability.

Multi-task problems arise in different areas of machine learning. The majority of this thesis deals with the sequential decision-making setting, RL in particular. Furthermore, we mostly consider *incompatible* RL environments, i.e., environments whose state or action set elements have different (*incompatible*) dimensions. For example, two robots with a different number of sensors and actuators are incompatible. Incompatibility here comes from the function approximation perspective, which usually assumes that state and action sets are spaces in the mathematical sense. This assumption makes incompatible environments inconvenient for deep RL and requires more sophisticated function approximation schemes, e.g., Graph Neural Networks (GNNs). For these inconvenience reasons, researchers in Multi-task Reinforcement Learning (MTRL) have primarily focused on compatible cases, which hinders many practical applications.

Furthermore, when the elements of state or action sets are different within one environment (there exist two states whose dimensions differ), we call such an environment *self-incompatible*. This specific case is essential when dealing with combinatorial optimisation problems, Boolean satisfiability or environments with object-oriented state representation. The latter case opens a vast range

of practical applications: using RL as a heuristic optimisation tool in generic object-oriented software [Car+18].

In the rest of this chapter, we provide a concise summary of the thesis. After that, in Chapter 2, we formally describe the problem setting and provide the necessary background for the rest of the work. We continue with the literature review in Chapter 3. The following four chapters (§§4-7) constitute the actual contribution of this thesis. We end with Chapter 8 drawing the conclusions and describing promising directions for future work. Appendix A provides the implementation details needed for replicating the results. Appendix B contains proofs for the theoretical results. Appendix C gives a list of practical advice for training GNNs in the RL setting.

1.2 Contribution

We will now give a high-level overview of the contributions of our work. This thesis comprises four contribution chapters, with the first one addressing multi-task optimisation and the last three exploring graph-based representations in incompatible RL settings.

1.2.1 Optimisation for Multi-task Learning

Recent multi-task learning research argues against simple minimisation of the sum of the task losses. Several ad-hoc multi-task optimisation algorithms have instead been proposed, inspired by various hypotheses about what makes multi-task settings difficult. The majority of these optimisers require per-task gradients and introduce significant memory, runtime, and implementation overhead.

Surprisingly, in Chapter 4, we show that when coupled with standard regularisation and stabilisation techniques from single-task learning, unitary scalarisation, a simple gradient summation baseline, matches or improves the performance of complex multi-task optimisers in both supervised and reinforcement learning settings. In addition, we present a theoretical analysis suggesting that many specialised multi-task optimisers can be interpreted as forms of regularisation.

We believe our results call for a critical reevaluation of recent research in the area and demonstrate the attractiveness of minimalist approaches. These findings motivate us to set the optimisation problem aside and focus on the representation, exploring graph-based state-action space representation for incompatible MTRL in Chapters 5, 6, and 7.

1.2.2 Scaling Graph Networks for Continuous Control

GNNs are an attractive choice for locomotion policies in continuous control. Wang et al. [Wan+18] and Huang et al. [HMP20] demonstrate generalisation and multi-task training benefits of GNNs across a set of incompatible environments. In Chapter 5, we show that the efficiency of GNNs decreases with the dimensionality of the problem, i.e. the dimensions of state and action set elements. We believe overfitting is one of the factors impacting performance deterioration and investigate this hypothesis by proposing freezing parts of a GNN to scale these models to high-dimensional problems. Surprisingly, such a minimalist technique considerably improves the performance of GNNs policies matching the performance of Multilayer Perceptrons (MLPs) while retaining incompatible generalisation and multi-task training abilities.

1.2.3 Transformers for Continuous Control

Chapter 5 shows that GNNs can learn a generalisable policy across incompatible environments when observations are represented as graphs. Like in prior work, the agent has access to the topologies of these graphs, assuming that they provide a practical inductive bias for learning and generalisation.

Chapter 6 relaxes this assumption asking the following questions. What if the provided topology is not optimal for the downstream task? What if the provided inductive bias is harmful to learning? Current research in incompatible continuous control, including our Chapter 5, uses the physical morphology to construct a graph inducing the message-passing schema. We demonstrate that this is not necessarily

the optimal case, showing that less constrained transformers [Vas+17] are capable of learning a generalisable policy across a set of incompatible environments.

Transformers are simpler than vanilla GNNs. They do not require the graph topology to bias the computation because they operate on fully-connected graphs and use attention for edge-to-vertex aggregation. Attention weights represent the implicit graph structure learnt from data and can vary depending on the inputs to the network. Moreover, this implicit structure can change throughout training, which might be helpful from the curriculum point of view.

1.2.4 Reinforcement Learning for Boolean Satisfiability

Graphs arise naturally in many critical scientific and practical industrial applications. In Chapter 7 we disembark from common RL benchmarks, and consider the Boolean Satisfiability (SAT) problem. The SAT problem is an NP-complete problem [Kar72], and SAT solvers heavily rely on heuristics laboriously crafted by humans. In our work, we investigate whether an RL algorithm equipped with a GNN can learn a generalisable branching heuristic for a SAT solver.

A SAT solver is an iterative algorithm which makes assumptions about Boolean variables to be verified in the future. The branching heuristic decides which variable to make an assumption about, i.e. set the chosen variable to *true* or *false*. Such problems provide an excellent testbed for MTRL, allowing for varying state and action set sizes and task distribution families.

We name our method Graph-Q-SAT and show that it reduces the number of iterations required to solve unseen SAT instances by 2-3X. Our method generalises to unsatisfiable instances without seeing them in training data, which has been an issue with previous approaches [Sel+19]. Graph-Q-SAT generalises to problems with 5X more variables than in its training data, which would be impossible with the conventional function approximation techniques. We also show positive transfer behaviour when testing the model on a different task distribution. We show that an RL agent does not require elaborate dataset crafting or feature engineering to work.

Moreover, we demonstrate its data efficiency showing that the model is capable of generalising even from a single instance of a SAT problem.

The more I think about language, the more it amazes me
that people ever understand each other at all.

— Kurt Gödel

2

Background

Contents

Before we cover the necessary background, we will briefly describe the mathematical notation used throughout this work. We use plain letters to denote scalars and bold letters to denote vectors: $\boldsymbol{x} \in \mathbb{R}^n$. Most of the time, we use $\boldsymbol{\theta}$ to denote neural network parameters. We use other bold greek letters when we want to separate multiple models trained together, e.g. $f_{\boldsymbol{\theta}}$ and $g_{\boldsymbol{\phi}}$. Capital calligraphic letters denote sets: $\mathcal{X} = \{x_1, x_2, \cdots, x_n\}$, where $|\mathcal{X}|$ is the number of the elements in the set: $|\mathcal{X}| = n$. Moreover, when all elements of a set are of the same dimension, we use $dim(\mathcal{X})$ to denote this dimension: $dim(\mathcal{X}) = d \iff \boldsymbol{x}_i \in \mathbb{R}^d \; \forall x_i \in \mathcal{X}$. Finally, we also use calligraphic letters to denote losses, e.g. $\mathcal{L}(\boldsymbol{\theta})$, and graphs, e.g. $\mathcal{G}$. We use $\mathbb{E}_{x \sim p(x)}\, x$ to denote the expected value of a random variable x. We

use capital plain letters for matrices, e.g., $X \in \mathbb{R}^{n \times m}$ with $X_{i,:}$ denoting the i-th row of matrix X, and $X_{:,j}$ denoting the j-th column of X.

The goal of MTL is to learn a single (typically task-aware) parametrised model $f_{\boldsymbol{\theta}}$ that performs well across a set of m tasks $\mathcal{T} := \{1, \dots, m\}$. The parameter space is often split into a set of shared parameters across tasks (generally the majority of the architecture), denoted $\boldsymbol{\theta}_{\|}$, and (possibly empty) task-specific parameters, denoted $\boldsymbol{\theta}_{\perp}$, so that $\boldsymbol{\theta} := [\boldsymbol{\theta}_{\|}, \boldsymbol{\theta}_{\perp}]^T$. In this context, the model f often takes on an encoder-decoder architecture, where the encoder g learns a shared representation across tasks, and the decoders h_i are task-specific predictive heads: $f(\boldsymbol{\theta}, X, i) = h_i(g(\boldsymbol{\theta}_{\|}, X), \boldsymbol{\theta}_{\perp})$. In this case, we denote by $\mathbf{z} = g(\boldsymbol{\theta}_{\|}, X) \in \mathbb{R}^{r \times n}$ the r-dimensional shared representation of X. In some of the works, however, the first layers of the encoder are not shared across the tasks providing an alternative way to condition the network on the task identifier.

For some of the related settings (e.g. transfer learning), the training task set $\mathcal{T}_{\text{train}}$ might differ from tasks the model is tested on $\mathcal{T}_{\text{test}}$. The exact meaning of *maximising the performance* and other essential details vary across problem settings which we will describe in the following sections §2.1 and §2.2.

2.1 Supervised Learning

In supervised learning, a task $\mathcal{T}_i$ is a tuple $(\mathcal{X}_i, \mathcal{Y}_i, \mathcal{L}_i)$, where $X \in \mathbb{R}^{d \times n}$ is a set of d-dimensional input points, $Y \in \mathbb{R}^{o \times n}$ is a set of o-dimensional labels, and $\mathcal{L}_i : \mathbb{R}^{o \times n} \times \mathbb{R}^{o \times n} \to \mathbb{R}$ is the task loss for the i-th task. There might be more general definitions, e.g. when X is a set of graphs, but we will not describe those here. Typically, the MTL optimisation problem is defined as a minimisation of the sum of per-task losses [SK18; Yu+20; Che+20]:

$$\min_{\boldsymbol{\theta}} \left[\mathcal{L}^{\text{MT}}(\boldsymbol{\theta}) := \sum_{i \in \mathcal{T}} \mathcal{L}_i(f(\boldsymbol{\theta}, X, i), Y) \right]. \tag{2.1}$$

Mean squared error loss:

$$\mathcal{L}(f(\boldsymbol{\theta}, X, i), Y) = \frac{1}{N} \sum_{j=1}^{n} (f(\theta, X_{:,j}, i) - Y_{:,j})^2, \tag{2.2}$$

and cross-entropy loss:

$$\mathcal{L}(f(\boldsymbol{\theta}, X, i), Y) = -\frac{1}{N}\sum_{j=1}^{n}(\log(f(\theta, X_{:,j}, i))^T \cdot Y_{:,j}) \tag{2.3}$$

are examples of typical task losses.

2.2 Reinforcement Learning

In RL, a task is typically formalised as an Markov Decision Process (MDP) [Put14] with some solution criterion defining the loss to describe RL problems. An MDP is a tuple $\langle \mathcal{S}, \mathcal{A}, \mathcal{R}, T, \rho \rangle$. The first two elements define the set of states $\mathcal{S}$ and the set of actions $\mathcal{A}$. We consider fully observable settings, and use *states* and *observations* interchangeably. The next element defines the reward function $\mathcal{R}(s, a, s')$ with $s, s' \in \mathcal{S}$ and $a \in \mathcal{A}$. $T(s'|s, a)$ is the probability distribution over states $s' \in \mathcal{S}$ after taking action a in state s. The last element of the tuple ρ is the distribution over initial states. A *task* and an *environment* are synonyms with an MDP in this work. Unless specified otherwise, we do not assume additional restrictions on the MDPs, e.g. shared state sets or the reward functions.

When environments are not self-incompatible, all elements of their state or action sets have the same dimensionality. Due to convenience reasons, modern MTRL usually considers *compatible* tasks that have the same dimensionality of state and action sets: $dim(\mathcal{S}_i) = dim(\mathcal{S}_j)$ or $dim(\mathcal{A}_i) = dim(\mathcal{A}_j)$, where subscipts denote a task index. In this work, we consider *incompatible cases*, i.e. when dimensionalities of their state or action spaces disagree: $dim(\mathcal{S}_1) \neq dim(\mathcal{S}_2)$ or $dim(\mathcal{A}_1) \neq dim(\mathcal{A}_2)$. We do not have additional assumptions on the semantics behind the state and action set elements. For instance, j-th coordinate of a state vector from $\mathcal{T}_1$ might have different semantics than j-th coordinate of a state vector from $\mathcal{T}_2$ even if their dimensions agree.

Sometimes, we expect tasks to have some additional structure known to us, e.g. joints connecting a robot's limbs. We can represent this as an annotated graph $\mathcal{G} = \langle \mathcal{V}, \mathcal{E} \rangle$, $\mathcal{V} = \{v_1, v_2, ..., v_n\}$, $\mathcal{E} = \{(i, j) \mid v_i, v_j \in \mathcal{V}\}$, with nodes

corresponding to the parts of the state set vector, and edges representing the relationships between these parts. We further discuss the graph representation of state and action sets in §5.2 and §7.3.

The goal of an RL agent is to find a *policy* $\pi_{\boldsymbol{\theta}}$ that maximises a given criterion. A policy $\pi_{\boldsymbol{\theta}}(a|s)$ is a mapping from states to distributions over actions. While there are several valid criteria in RL, in this work, we solely focus on the expected discounted cumulative return:

$$J(\boldsymbol{\theta}) = \mathbb{E}_{a\sim\pi_{\boldsymbol{\theta}},T,\rho}\Big[\sum_{t=0}^{\infty}\gamma^t r_t\Big], \tag{2.4}$$

where $\gamma \in [0, 1)$ is a discounting coefficient, t is the environment step and r_t is the reward for a transition from state s_t to state s_{t+1}. In the rest of the work, we will omit the variables under the expectation sign to make notation less cluttered. In the MTRL setting, one wants to maximise the average performance across several tasks:

$$J_{\text{mtrl}}(\boldsymbol{\theta}) = \frac{1}{M}\sum_{\mathcal{T}} J_{\mathcal{T}}(\boldsymbol{\theta}), \tag{2.5}$$

where M is the number of tasks.

2.2.1 Deep Reinforcement Learning Algorithms

We now have the background to describe RL algorithms used throughout this work. Here, we consider model-free methods, i.e. those that do not learn a model for the transition function of a task. Model-free methods are typically classified into two groups: policy-based and value-based, with actor-critic methods lying in between.

In value-based methods, one learns a value function and uses it to make decisions. Depending on the algorithm, one either learns the state-value function $V^{\pi}_{\boldsymbol{\phi}}$ or the action-value function $Q^{\pi}_{\boldsymbol{\phi}}$. State-value function $V^{\pi}_{\boldsymbol{\phi}}(s)$ is the expected discounted return a policy π receives starting at state s:

$$V^{\pi}_{\boldsymbol{\phi}}(s) = \mathbb{E}[\sum_{t=0}^{\infty}\gamma^t r_t \mid s_0 = s]. \tag{2.6}$$

The action-value function $Q^{\pi}_{\phi}(s,a)$ is the expected discounted return a policy π receives starting at state s, taking action a and following policy π aftwerards:

$$Q^{\pi}_{\phi}(s,a) = \mathbb{E}[\sum_{t=0}^{\infty} \gamma^t r_t \mid s_0 = s, a_0 = a]. \tag{2.7}$$

Policy gradient methods [PS06] aim to learn a policy using a gradient ascent on the objective: $\boldsymbol{\theta}_{t+1} = \boldsymbol{\theta}_t + \alpha \nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta})|_{\boldsymbol{\theta}=\boldsymbol{\theta}_t}$, where $\boldsymbol{\theta}$ parameterises a policy, and α is the learning rate. Often, to fight high variance in the gradients, one learns a critic with policy gradient becoming $\nabla_{\boldsymbol{\theta}} J(\boldsymbol{\theta}) = \mathbb{E}\left[\nabla_{\boldsymbol{\theta}} \log \pi_{\theta}(a_t|s_t) A^{\pi}_t\right]$, where A^{π}_t is an estimate of the advantage function, e.g. temporal difference residual $r_t + \gamma V^{\pi}_{\phi}(s_{t+1}) - V^{\pi}_{\phi}(s_t)$ [Sch+16]. While there are discrepancies between the theory and practice of policy gradient methods [NT20], these algorithms have been widely adopted in the research community, demonstrating success across a variety of problems [Sch+15; Hee+17; Ber+19].

2.2.1.1 Deep Q-Networks

Deep Q-Networks (DQN) [Mni+15] is an extension of Q-learning [WD92] that uses neural networks for state-action value function approximation (Q-function) explaining the name of the algorithm. The resulting policy behaves greedily with respect to the learnt Q-function. The learner samples transition data (s, a, s', r, d) from the replay buffer $\mathcal{B}$, where r is the reward for the transition from state s to s', and d is a boolean indicating the termination of an episode. The replay buffer has a limited capacity overwriting the oldest entries when full. The parameters of the model are updated to minimise the squared temporal difference loss:

$$L(\boldsymbol{\theta}) = \mathbb{E}_{(s,a,s',r,d)\sim\mathcal{B}}\left[r + \gamma(1-d) \cdot \max_{a'} Q_{\boldsymbol{\theta}}(s',a') - Q_{\boldsymbol{\theta}}(s,a)\right]^2. \tag{2.8}$$

To stabilise learning, a separate *target* network is used, resulting in the following update:

$$L(\boldsymbol{\theta}) = \mathbb{E}_{(s,a,s',r,d)\sim\mathcal{B}}\left[r + \gamma(1-d) \cdot \max_{a'} Q_{\bar{\boldsymbol{\theta}}}(s',a') - Q_{\boldsymbol{\theta}}(s,a)\right]^2, \tag{2.9}$$

where $Q_{\bar{\boldsymbol{\theta}}}$ is the target network. The parameters of the target network are hard-copied from $Q_{\boldsymbol{\theta}}$ once every k episodes, where k is a hyperparameter trading off learning speed and stability. We use DQN in Chapter 7.

2.2.1.2 Proximal Policy Optimisation

Instabilities coming from drastic changes in policy behaviour are a major challenge in policy-based RL. Proximal Policy Optimisation (PPO) [Sch+17] is an actor-critic method, that constrains the size of policy updates. In particular, Chapter 5 and 6 use a clipping version of PPO that does k sweeps over the sampled rollouts $\mathcal{B}_e = \{\tau_1, \tau_2, ..., \tau_b\}$ and clips the objective to avoid large changes (e denotes the epoch index):

$$L(\boldsymbol{\theta}) = \mathbb{E}_{\tau\sim\mathcal{B}_e}\, \mathbb{E}_{s,a\sim\tau} \min[r_k(\boldsymbol{\theta})\hat{A}^{\pi_{\boldsymbol{\theta}_k}}(s,a),\ \text{clip}(r_k(\boldsymbol{\theta}), 1-\epsilon, 1+\epsilon)\hat{A}^{\pi_{\boldsymbol{\theta}_k}}(s,a)], \quad (2.10)$$

where $r_k(\boldsymbol{\theta}) = \frac{\pi_{\boldsymbol{\theta}}(a|s)}{\pi_{\boldsymbol{\theta}_k}(a|s)}$ and $\boldsymbol{\theta}_k$ are policy parameters after k iterations over $\mathcal{B}_e$, and $\boldsymbol{\theta}$ are policy parameters before any updates on $\mathcal{B}_e$. Hyperameter ϵ specifies how much an updated policy can deviate within an epoch. PPO is an on-policy algorithm, and the data $\mathcal{B}_e$ is discarded after k sweeps over $\mathcal{B}_e$.

2.2.1.3 Twin Delayed Deep Deterministic Policy Gradients

Fujimoto et al. [FvHM18] introduce Twin Delayed DDPG(TD3), an off-policy policy-gradient method based on Deep Deterministic Policy Gradient (DDPG) [Lil+16]. DDPG aims to learn a deterministic policy to maximise the action-value function:

$$\max_{\boldsymbol{\theta}} \mathbb{E}_{s\in\mathcal{B}}\, Q_{\boldsymbol{\phi}}(s, \pi_{\boldsymbol{\theta}}(s)), \quad (2.11)$$

where $\mathcal{B}$ is the replay buffer used as in DQN (§2.2.1.1). Learning a Q-function is performed similarly to DQN, but there is one important difference. In DQN, maximising the action-value function over action is possible for a discrete action set. DDPG admits continuous action spaces when finding the maximum iterating over all possible actions is impossible. To do this, DDPG learns a target function $Q_{\hat{\phi}}$ to output an approximate maximum:

$$L(\boldsymbol{\phi}) = \mathbb{E}_{(s,a,s',r,d\sim\mathcal{B})}(r + \gamma(1-d)\cdot Q_{\hat{\boldsymbol{\phi}}}(s',a') - Q_{\boldsymbol{\phi}}(s,a))^2. \tag{2.12}$$

Notice, that Equation 2.12 has no maximisation over state-action values in contrast to Equation 2.9. Moreover, instead of the hard weight copying, the target network is updated by averaging with a hyperparameter ρ:

$$\hat{\boldsymbol{\phi}} \leftarrow \rho\hat{\boldsymbol{\phi}} - (1-\rho)\boldsymbol{\phi}. \tag{2.13}$$

Deep RL algorithms are known to be brittle [Hen+17]. Twin Delayed Deep Deterministic policy gradient (TD3) stabilises DDPG by three techniques. First, TD3 learns two Q functions instead of one and uses the minimum value when calculating the targets ("twin"). Second, the algorithm updates the actor less frequently than the critic ("delayed"). Finally, TD3 adds noise to actions when computing the targets to *'smoothen the value estimate'* [FvHM18]. We use TD3 in Chapter 6.

2.2.1.4 Soft Actor-Critic

Soft Actor-Critic (SAC) [Haa+18] is an off-policy actor-critic algorithm that incorporates entropy regularisation to the action-value function targets. SAC also uses the same twin-value trick as TD3, which we omit here for clarity. This makes the Q-function update look as follows:

$$L(\boldsymbol{\phi}) = \mathbb{E}_{(s,a,r,s',d)\sim\mathcal{B}}\left[r + \gamma(1-d)\Big(Q_{\bar{\boldsymbol{\phi}}}(s',\tilde{a}') - \alpha\log\pi_{\boldsymbol{\theta}}(\tilde{a}'|s')\Big) - Q_{\boldsymbol{\phi}}(s,a)\right]^2, \tag{2.14}$$

where action $\tilde{a}'$ is sampled from the policy: $\tilde{a}' \sim \pi_{\boldsymbol{\theta}}(\cdot \mid s')$. To update the policy, SAC makes an ascent on the following objective:

$$L(\boldsymbol{\phi}) = \hat{\mathbb{E}}_{s\in\mathcal{B}}[Q_{\boldsymbol{\phi}}(s,\tilde{a}) - \alpha\log\pi_{\boldsymbol{\theta}}(\tilde{a} \mid s)], \text{ with } \tilde{a} \sim \pi_{\boldsymbol{\theta}}(\cdot \mid s). \tag{2.15}$$

To backpropagate through sampling, SAC uses the reparameterisation trick [KW14]. Hyperparameter α controls the importance of the entropy regularisation and, in some of the implementations, is being learnt by adding an additional component to the loss. We use SAC in Chapter 4.

2.3 Graph Neural Networks

In this work, we make extensive use of GNNs due to their ability to process graphs of arbitrary sizes. We adopt the formalism of Battaglia et al. [Bat+18], which unifies many existing Message Passing Neural Network (MPNN) models. Throughout the work, we are using MPNN and GNN as synonyms.

In this formalism, an annotated graph $\mathcal{G} := \langle \mathcal{V}, \mathcal{E}, u \rangle$ consists of a set of vertices $v^i \in \mathcal{V}$, annotated with vectors $\boldsymbol{v}^i \in \mathbb{R}^{m_v}$, a set of directed edges $e^{ij} \in \mathcal{E}$ from vertex v^i to v^j, annotated with vectors $\boldsymbol{e}^{ij} \in \mathbb{R}^{m_e}$, and the global attribute u, annotated with vector $\boldsymbol{u} \in \mathbb{R}^{m_u}$. We use *nodes* and *vertices* interchangeably throughout this work. Also, we often call the edge set $\mathcal{E}$ *topology of the graph* or *connectivity of the graph*. Figure 2.1 provides an example of a graph without the global attribute. On

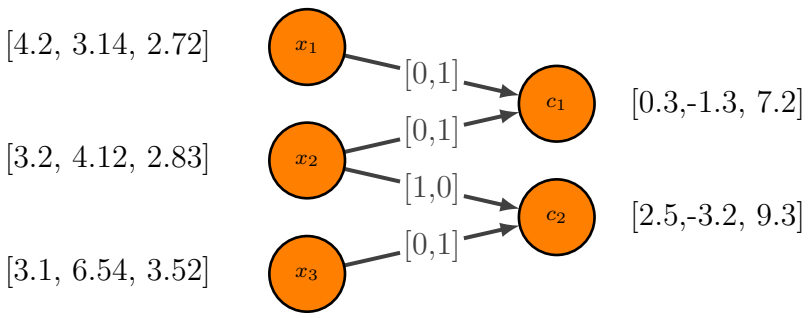


Figure 2.1: An example of a graph with five vertices and four edges. Vertex features are three-dimensional vectors, and edge features are two-dimensional. This graph does not have the global attribute.

a higher level, a GNN is a model that takes an annotated graph $\mathcal{G}$ as input and outputs a graph $\mathcal{G}'$ with different annotations but the same topology. Note that the output graph $\mathcal{G}'$ has the same topology, but the annotations can be of different dimensionality than the input, that is, $\boldsymbol{v}'^i \in \mathbb{R}^{m'_v}$, $\boldsymbol{e}'^{ij} \in \mathbb{R}^{m'_e}$ and $\boldsymbol{u}' \in \mathbb{R}^{m'_u}$.

We now describe a forward step of a GNN. Output annotations for entities of type k are computed by parameterised *update functions* $\varphi^k_{\boldsymbol{\theta}}$, for example, neural networks, where we denote all GNN parameters jointly as $\boldsymbol{\theta}$. These updates can depend on a varying number of edges or vertices, which have to be summarised first using *aggregation functions*, which we denote ρ^h_k to indicate that they aggregate

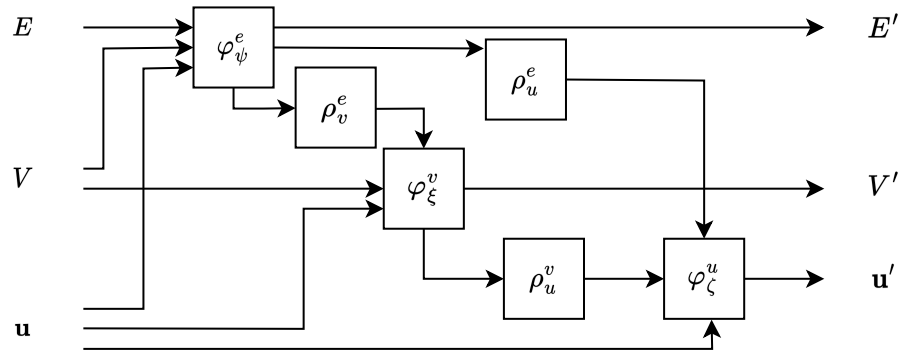


Figure 2.2: Visual representation of a forward step of a GNN (Equation 2.16). Square boxes denote functions, and incoming arrows are function inputs, outgoing arrows are the outputs.

entities of type h for the update of an entity of type k. Apart from their abilities to operate on sets of elements, aggregation functions should be permutation invariant. Examples of such aggregation functions include summation, averaging and *max* or *min* operations. In some models [Vel+17], the aggregation functions are also learnt. Update functions are usually neural networks that can be learned end-to-end via backpropagation [RHW86].

The GNN block performs the following computations (in order):

$$\begin{aligned} \boldsymbol{e}'^{ij} &\leftarrow \varphi^e_{\boldsymbol{\psi}}\Big(\boldsymbol{u},\, \boldsymbol{e}^{ij},\, \boldsymbol{v}^i,\, \boldsymbol{v}^j\Big) && \forall e^{ij} \in \mathcal{E}\,, \\ \boldsymbol{v}'^{i} &\leftarrow \varphi^v_{\boldsymbol{\xi}}\Big(\boldsymbol{u},\, \boldsymbol{v}^i,\, \rho^e_v\{\boldsymbol{e}'^{ki} \,|\, e^{ki} \in \mathcal{E}\}\Big) && \forall v^i \in \mathcal{V}\,, \\ \boldsymbol{u}' &\leftarrow \varphi^u_{\boldsymbol{\zeta}}\Big(\boldsymbol{u},\, \rho^v_u\{\boldsymbol{v}'^{i} \,|\, v^i \in \mathcal{V}\},\, \rho^e_u\{\boldsymbol{e}'^{ij} \,|\, e^{ij} \in \mathcal{E}\}\Big)\,, \end{aligned} \tag{2.16}$$

Figure 2.2 provides a visual schema of the forward step of a GNN. Using only one GNN block restricts the learned function to local computations on the graph, i.e. each vertex will have access to its immediate neighbourhood. On the other hand, chaining multiple blocks after another (with possible parameter sharing) can propagate information further throughout the graph.

2.3.1 Transformers

Transformers [Vas+17] are another network architecture we use in this work. Transformers can be seen as GNNs operating on complete graphs (every node is connected

to every other node) using attention as an edge-to-vertex aggregator [Bat+18].

We are interested in transformers for two reasons. First, transformers do not need to propagate messages across distant paths in the graph (the graph diameter is one). Second, their message-passing schema depends on a particular input to the network, e.g. it is state-dependent in the RL setting.

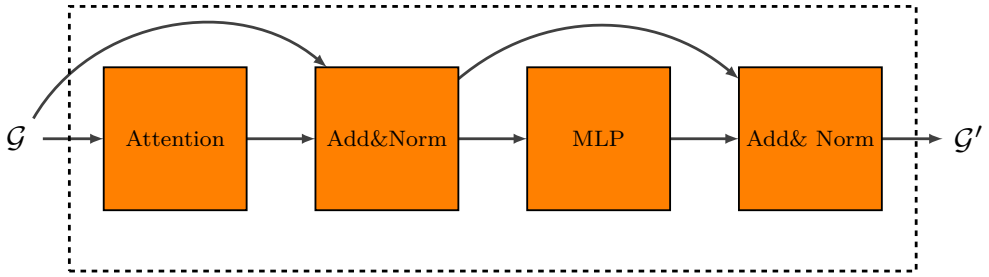


Figure 2.3: Visual representation of operations in a Transformer block [Vas+17].

The attention mechanism is a crucial part of a Transformer. Intuitively, attention is an associative memory-like mechanism when one compares the query vector of a sender node with a key vector of the receiver node and uses similarities between them as a weighting factor for the value vectors of the sender node. The more similar the query-key pairs of two nodes, the larger the weight of the value vector will be in the final weighted sum. The dot product is used as a similarity metric.

More formally, the input to the Transformer block is multiplied by three matrices W_Q, W_K and W_V, weights for the query, key and value vectors, respectively. The node annotations are updated using the following procedure:

$$V \leftarrow \texttt{softmax}\Big(\frac{W_Q V \cdot (W_K \cdot V)^T}{\sqrt{d_k}}\Big) W_V \cdot V, \tag{2.17}$$

where $V \in \mathbb{R}^{n \times m_v}$ matrix of node annotations. The constant d_k is used to counteract small gradients in softmax when the dot product grows large in magnitude.

Often, multiple attention heads, i.e., W_K, W_V, W_Q matrices, are used to learn different interactions between the nodes and mitigate the consequences of unlucky initialisation. The output of multiple heads is concatenated and later projected to respect the dimensions.

A Transformer block is a combination of an attention block and a feedforward layer with a possible normalisation between them. In addition, there are residual connections from the input to the attention output and from the output of the attention to the feedforward layer output (see Figure 2.3 for visual description). Similarly to standard GNN blocks, one can stack Transformer blocks to take higher-order dependencies into account, i.e., reacting not only to the nodes' features but how the nodes' features change after applying a Transformer block.

Reading, after a certain age, diverts the mind too much from its creative pursuits. Any man who reads too much and uses his own brain too little falls into lazy habits of thinking.

— Albert Einstein

3 Literature Review

Contents

In this chapter, we overview existing MTL literature, identifying the main trends and providing context for our work. Since the current thesis focuses mostly on MTRL, with just a brief encounter with supervised MTL in Chapter 4, we give only a concise overview of supervised MTL in §3.1 with a more detailed comparison of multi-task optimisers in §4.4. We refer the reader to Ruder [Rud17] for a more detailed reference on supervised MTL.

In addition to MTL in a narrow sense (training on multiple tasks), the RL section of this chapter (§3.2) also presents a concise overview of other related problems, e.g., study of generalisation, curriculum learning, transfer or meta-learning which constitute MTRL in a broad sense. Finally, §3.3 describes existing research of incompatibility in RL identifying arising challenges and reviewing existing benchmarks in the field.

3.1 Supervised Learning

MTL is represented by an extensive body of literature, including Caruana's pioneering work [Car97a] exploring *hard parameter sharing*, i.e., sharing neural network parameters between all tasks with, possibly, a separate part of the model for each task. Hard parameter sharing is still the major MTL approach adopted in natural language processing [CW08; CZY21], computer vision [Mis+16], and speech recognition [SD13].

Many works strive to improve the performance of deep multi-task models. One line of research hypothesizes that conflicting per-task gradient directions lead to suboptimal models, and thus focuses on explicitly removing conflicts across task gradients [Yu+20; Che+20; Liu+21b; Wan+21; JV22; Liu+21a]. Some authors postulate that loss imbalances across tasks hinder learning, proposing loss reweighting methods [KGC18; Che+18b; LYZ21]. Finally, Sener et al. [SK18] and Navon et al. [Nav+22] propose that tasks compete for model capacity and interpret MTL as multi-objective optimisation in order to cope with inter-task competition. In Chapter 4, we focus on algorithms that explicitly rely on per-task gradients to try to outperform unitary scalarisation (§4.4). Research on multi-task architectures [Mis+16; GLU20] or MTL algorithms exclusively motivated by deterministic loss reweighting [KGC18; Guo+18; LJD19] are orthogonal to our work. Both topics are investigated by a recent survey on pixel-level multi-task computer vision problems [Van+22], which found that the minimisation of tuned weighted sums

of losses is empirically competitive with deterministic loss reweighting and Multiple-Gradient Descent Algorithm (MGDA) (§4.2.2) in the considered settings.

Many of the MTL methods implicitly assume that each parameter update employs information from all tasks. However, not all works satisfy this assumption, either due to a large number of tasks or as an implementation decision. This detail builds a bridge from a narrow understanding of MTL as multi-task training approach to a broader view on MTL including other problems dealing with multiple tasks, i.e., continual [Khe+20a], curriculum [Nar+20a], and meta-learning [Hos+22].

3.2 Reinforcement Learning

Sutton admits the importance of MTL for learning useful representations Sutton [Sut92]:

> Finally, a broad conclusion that I make from this work has to do with the importance of looking at a series of related tasks, such as here in a non-stationary tracking task, as opposed to conventional single learning tasks. Single learning tasks have certainly proved extremely useful, but they are also limited as ways of exploring important issues such as representation change and identification of relevant and irrelevant features. Such meta-learning issues may have only a small, second-order effect in a single learning task but a very large effect in a continuing sequence of related learning tasks. Such cross-task learning may well be key to powerful human-level learning abilities.

In a broad sense, MTRL means a subfield of RL dealing with multiple environments and investigating challenges arising in this setting. The goal of MTRL is to alleviate the deficiencies of single-task RL, obtaining a model that generalises better, is more robust in training, yields better performance and exhibits better sample efficiency. While MTRL can leverage information unavailable to its single-task counterpart, dealing with multiple tasks can be challenging and leads to multiple issues we describe in the following subsections.

3.2.1 Multi-task Learning

This section includes the work where we expect an agent to solve all the training tasks comprising the training task set (multi-task learning in a narrow sense). In contrast to RL with auxiliary tasks, all the tasks are MDPs (or some variations of them, e.g., partially observable MDPs).

Wilson et al. [Wil+07] and Lazaric et al. [LG10] take a hierarchical Bayesian approach to MTRL. The former assumes the tasks have similar dynamics and the reward function, working with one task at a time. This is closer to linear curriculum learning (§3.2.5). The task sequence might be infinite, which brings this work closer to continual learning (§3.2.6). Lazaric et al. [LG10] assume that the tasks have some common structure in their value function and use Gaussian process temporal-difference [EMM05] and hierarchical Bayesian models based on Dirichlet processes. The latter allows an arbitrary number of possible value function classes. Li et al. [LLC09] use a nonparametric approach with Dirichlet processes to share information for the policy between similar partially observable MDPs. It is currently unknown whether these methods will scale to modern deep RL settings, leaving a potentially fruitful research direction wide open.

We would like to split modern MTRL literature into two groups: *parameter space* and *behaviour space* approaches.

3.2.1.1 Parameter Space Approaches

Parameter space methods focus on updating the model weights ignoring the effect of those weights on the agent's behaviour. We can further split this group into *sequential task sampling* and *gradient averaging* methods.

In **sequential task sampling**, agent models are updated using the rollouts from one task at a time, similar to curriculum learning. Parisotto et al. [PBS16] uses sequential task sampling for their MTRL baselines, showing the weaknesses of this approach. We believe this approach is subpar due to *catastrophic forgetting* which is known to be an issue with neural networks [MC89] and even happening within one

environment [Fed+20]. However, there is research showing impressive results with sequential updates Huang et al. [HMP20, Shared Modular Policies (SMP)]. The authors do not discuss the reasons behind this design choice, omitting it as a trivial implementation detail. We further discuss Huang et al. [HMP20] in §6.2. In addition, we observed a slight advantage of per-task stepping over gradient averaging [Iba+22] in the neural algorithmic reasoning setting [VB21]. We believe that task subsampling might be an attractive middle ground in terms of data efficiency and convergence speed, similarly to mini-batch stochastic gradient descent in the single-task setting.

Every gradient update uses data from multiple tasks in the **gradient averaging group**. The optimiser averages (or sums) the gradients to make an update step [Wan+18]. In Chapter 4, we call this approach unitary scalarisation. Sometimes, different tasks are weighted based on some performance metric.

Many researchers report that MTRL often brings new challenges instead of facilitating learning. There is no consensus on why this is the case. *Gradient interference* is one of the widespread explanations of this fact, meaning that the gradients from different tasks may cancel each other leading to difficulties when learning in the MTL regime. Yu et al. [Yu+20] propose a way to alleviate gradient interference via projecting the conflicting gradients to a normal vector of the grads of another task (pairwise). While they demonstrate that using their method dramatically improves results on the Meta-world benchmark, the question of whether gradient interference is the reason behind the subpar performance of the baseline remains an open question. In Chapter 4, we empirically demonstrate that previous literature severely underreported the baseline performance. Moreover, we show that with well-known single-task regularisation and learning stabilisation methods, none of the specialised optimisers consistently outperforms the baseline. In particular, increasing the replay buffer size and proper reward normalisation greatly improved the gradient averaging baseline.

Differences in reward scales are an alternative hypothesis on why MTRL is challenging. Hessel et al. [Hes+19] extend van Hasselt et al. [vHas+16] for MTRL achieving good performance on Atari 2600 and DMLab benchmarks. Adding PopArt

normalisation [vHas+16] to IMPALA [Esp+18] shows that equalising the impact of different tasks on the learning dynamics positively affects the performance. At the same time, even without PopArt, IMPALA is a successful parameter-space approach. It is a distributed architecture with an off-policy correction that demonstrates impressive results on 30 DMLab tasks [Bea+16] and 59 Atari [Bel+13] games. IMPALA exhibits better asymptotic performance and gains in sample efficiency on DMLab. For Atari experiments, the deep version of the network (ResNet) shows comparable median performance to the single-task DQN-like architecture used in most baselines. Interestingly, comparing deep architectures for both shows that these gains are partly due to the architecture (59.7% vs 117% in the median and 176.9% vs 503.6% in the mean), showing the potential of scaling in the MTRL setting. PopArt has been shown effective for V-MPO [Son+20], an on-policy version of Abdolmaleki et al. [Abd+18a] that performs a policy improvement step toward a policy nonparametrically designed from the value function. V-MPO has higher asymptotic performance for both of the benchmarks compared to IMPALA. In both Atari and DMLab, the agent learns from pixels with Convolutional Neural Networks (CNNs) responsible for state set compatibility. Apart from this, the paper is quite scarce on the details of their MTRL setup.

Learning modular policies is another crucial subgroup in MTRL with a discrete task distribution. Options [SPS99] can be viewed as a part of this group as well. Devin et al. [Dev+17] and Andreas et al. [AKL17] train separate modules and learn how to recombine them. In Peng et al. [Pen+19], the primitives are activated simultaneously using multiplicative composition. Yang et al. [Yan+20] use a routing network to reuse the same policy for different tasks. However, they use a soft combination of all possible routes, compared to the previous work that uses a specific route for a task.

Most of the MTRL work assumes full sharing of the policy or value network, with some exceptions, e.g., having separate entropy coefficients for SAC in [SZP21]. However, there is a line of work deviating from this assumption. For example, D'Eramo et al. [DEr+20] and Ibarz et al. [Iba+22] learn separate encoder and

decoder layers, partly due to incompatibility. Wang et al. [Wan+18] has a similar implementation of their MTRL baseline, showing that NerveNet yields higher returns compared to sharing, demonstrating the benefits of the modular GNNs approach. In addition, Wang et al. [Wan+18] do not share critics across the tasks in their primary hyperparameter setting. While this might improve learning stability, the sample efficiency and the memory footprint will suffer when the number of tasks is large.

3.2.1.2 Behaviour Space Approaches

Policy distillation approaches represent most of the behaviour space group. Policy distillation implies having task-specific policy networks and distilling [BCN06; HVD15] them into one model using supervised learning techniques.

Rusu et al. [Rus+16a] train ten single-task *teacher* policies until convergence, one per Atari 2600 game, and later train a single network to match the actions of the teacher policies. This approach leads to 108% of the parameter-space MTRL performance using the geometric mean as a metric [FW86]. Compared to single-task models, their approach achieves 89.3%. The data is sampled sequentially from ten different experience buffers generated by teacher networks. The learner uses a separate output head per task that, according to the authors, *allows different action sets for different environments*. We believe that this implementation detail brings additional stability by providing more network capacity, which is updated using the data of a specific task. When evaluated on three games, the multi-task baseline achieves 83.5% of the single-task policies. However, as soon as the number of tasks becomes ten, the authors do not even provide the baseline results, saying that the baseline failed to outperform even the random agent. The authors explain the failure of MTRL with interference, different reward scaling and instability of learning value functions. While this is a viable hypothesis, we believe this might also be related to how they train the baseline. In particular, sequential sampling of the data (using only one task data for the update) might be damaging due to catastrophic forgetting.

Concurrently, Parisotto et al. [PBS16] distil converged single-task policies into a single policy in a supervised manner. They also study the transfer behaviour

of multi-task policies, using the distilled network weights as initialisation for the target task network. They demonstrate that sometimes there is positive transfer, and sometimes the transfer is negative, probably related to source-target task similarities. Surprisingly, the distilled policy outperforms teacher policies in 3 out of 8 games (judging by the mean returns for the last ten training epochs) and (1 out of 8) judging by the max returns.

Teh et al. [Teh+17] propose Distral, a policy distillation method which uses the distilled policy only as a regulariser for task-specific policies. Notably, both teacher and student policies learn concurrently, an essential factor for the sample efficiency of the approach. Rusu et al. [Rus+16a] also try to learn teacher and student concurrently, but they do it for the single-task distillation only. Distral outperforms the multi-task version of A3C [Mni+16], explaining this with gradient interference when learning from multiple tasks. Distral assumes that the state and action spaces are shared, allowing the environment dynamics and the reward functions to differ. Czarnecki et al. [Cza+19] extensively compares policy distillation techniques, including practical advice based on theoretical and empirical analysis.

Looking at the impressive results of policy distillation research, one might contemplate the reasons for such success. We believe a part of the explanation is controlling the behaviour space rather than controlling the parameter space. Averaging in the parameter space is meaningful in MAML-like meta-learning, where one wants to obtain a policy most likely to adapt to any new task from some distribution. In MTRL, when the goal is to find a policy excelling in all of the tasks, averaging the parameters might be less meaningful. In addition, single-task RL has several successful algorithms where controlling the behaviour space bears fruit. The success of [Sch+15, TRPO] and [Sch+17, PPO] continuing the work of [Kak01a] comes from restricting large steps in the behaviour space. Kirkpatrick et al. [Kir+16] apply similar ideas to retain policy in the continual learning setting (§3.2.6).

Recently, several works showed that a minimalist approach to multi-task learning is enough to achieve impressive results, supporting our findings in Chapter 4. Humphreys et al. [Hum+22] show that multi-task training leads to more than

a twofold increase in performance and sample efficiency on Liu et al. [Liu+18, Miniwob++], a set of computer control tasks such as copy-pasting, navigating a file tree to find a folder, or filling in the login form. Reed et al. [Ree+22, Gato] train a single Transformer (§2.3.1) to perform 604(!) distinct incompatible tasks including continuous control, playing Atari 2600 games, captioning images and generating text given prompts. Notably, they use unitary scalarisation, only controlling the relative number of samples per task in a mini-batch. While the total number of parameters of Gato is modest (1.18B) compared to modern language models (Brown et al. [Bro+20] uses 175B), the model is large by the RL standards. We believe not giving enough capacity for the model is another reason behind the subpar performance of unitary scalarisation in some MTRL literature. Finally, in our own work [Iba+22], we learn a single GNN model (with separate encoders and decoders) to execute 30 algorithms, including sorting, graph, searching, and geometric algorithms.

While the MTRL field is rapidly expanding and new works appear every week, we still find it helpful to refer to the surveys for more context on the research field. Kalashnikov et al. [Kal+21] gives a recent review on deep MTRL. Chapter 3 of Zhang et al. [ZY17] provides pointers to the pre-DQN RL research.

3.2.2 Generalisation

Generalisation in RL has multiple meanings. A standard way of evaluating RL agents is to collect multiple rollouts using the resulting policy and average the returns. The agent is evaluated on the *same* task, but due to the stochasticity of the policy and the environment, the policy is evaluated on a different data distribution. This section deals with a different type of generalisation: training on a distribution of tasks and evaluating either a single target task or a test distribution of target tasks. In what follows, we will use *generalisation* to denote the latter kind.

Whiteson et al. [Whi+09] and Whiteson et al. [Whi+11] propose *generalised environments* to evaluate RL algorithms preventing overfitting to a particular benchmark. A generalised environment comprises a distribution over tasks induced

by the parameters defining the reward function and transition dynamics. Doshi-Velez et al. [DK16] explore similar ideas formalising them as *Hidden Parameter MDPs*.

Recently, a vast body of work has identified problems with overfitting and generalisation in deep RL [ZBP18; Zha+18]. As a result, the community proposed multiple benchmarks to study these phenomena. These benchmarks are, effectively, generalised environments, where tasks differ in their visual appearance [Cob+19], map layout [Cob+19; CWP18] or dynamics [Yu+19].

Many works in modern RL investigate generalisation for environments with rich visual inputs when CNNs [LeC+89] are employed. For example, Lee et al. [Lee+20] tackles the generalisation problem via introducing a random CNN to perturb the inputs making the agents more adaptable by learning features invariant across randomised environments. We refer the reader to the 'Related Work' section of Igl et al. [Igl+19] for a comprehensive overview of such literature.

In **domain adaptation** [Tze+15], one wants to bring the training data distribution closer to the target. Rajeswaran et al. [Raj+17] train a policy on a distribution of tasks adjusting the distribution given the policy performance on the target task encouraging training more on the environments where the agent performs poorly. Similarly, Chebotar et al. [Che+19] interleave policy training on a distribution of rollouts from different environments optimising the average performance. Eysenbach et al. [Eys+21] study domain adaptation modifying the reward function penalising the agent that visits the states which do not exist in the target domain.

Similarly, Ciosek et al. [CW17] concurrently optimise for the policy and the distribution over parameters inducing the transition function, making the policy robust to significant rare events, e.g. unusual wind conditions. Implicitly, their method trains a policy on a distribution of tasks induced by a parameter vector. However, the final goal of optimisation is to find a policy which excels in one target task.

In contrast to domain adaptation, **domain randomisation** does not require the target distribution, sampling environment parameters from some distribution so that the variability explains the discrepancies between the real world and the

simulator. As one can see, domain randomisation effectively adopts *generalised environments* for training and evaluation.

Many works in robotics focus on the vision part of the pipeline [SL17; Jam+19]. Tobin et al. [Tob+17] also focus on the vision part but vary the number of objects and their positions in the scene, leaving the transition dynamics untouched. Antonova et al. [Ant+17] train a feedforward policy randomising the dynamics of the environment.

Sim2real is a critical problem that deals with challenges when deploying a model, trained within a simulation, in the real world. No simulation is perfect, sensors and actuators of a robot are noisy, leading to compounding errors and model failures. Mordatch et al. [MLT15] propose to optimise the trajectories on a set of different environment dynamics for robustness in the offline planning setting. Injecting noise to the action's delay and estimates of the friction values makes the policy more robust when tested on a real robot, i.e. helping with SIM2REAL transfer. Peng et al. [Pen+18] train a recurrent policy sampling mass, friction or damping parameters from either logarithmic or uniform distribution. A recurrent policy enables the agent to infer the hidden properties of the environment. This approach allows deploying the policy on a robotic arm without additional training on the real data.

Works above take the most straightforward approach, sampling the parameters defining the environments from some generic distribution, e.g., uniform or logarithmic. Such uninformative sampling is computationally expensive and involves unnecessary training of the agents in unrealistic environments. Mehta et al. [Meh+19] adjust the environment distribution training an agent on the most informative environments, measuring the discrepancies in the policy performance on the reference environment and the environments sampled from the random distribution. While the authors focus on generalisation, they perform SIM2REAL transfer showing a slightly better performance on average and reducing the variance across test rollouts. OpenAI et al. [Ope+19] demonstrates the potential of automatic domain randomisation by training a robotic hand to manipulate a Rubik's cube. This paper's 'Related Work' section can serve as a further reference on domain randomisation research for SIM2REAL.

The majority of the generalisation studies focuses on continuous distribution over tasks. Studies of generalisation in RL on discrete distributions of tasks are scarce and mainly consider large samples from discrete distributions. This line of work includes combinatorial optimisation with RL [Kha+17; Kur+20; Cap+21] as well as ProcGen environments gaining popularity within the RL community [Cob+20; Küt+20] including our own work [Sam+21]. In Chapter 7, based on Kurin et al. [Kur+20], we study generalisation properties of a graph-based DQN agent using the SAT problem as a benchmark.

Finally, while a high-fidelity simulation is impossible for every task, a good simulator reduces the need for algorithmic complexity. Recently, Degrave et al. [Deg+22] demonstrated an impressive zero-shot SIM2REAL generalisation controlling plasma in a real tokamak using Maximum a Posteriori Policy Optimisation [Abd+18b], a standard RL algorithm.

Generalisation research is too broad to cover in one short subsection. We refer the reader to Kirk et al. [Kir+21] for an extensive survey, including an overview of the benchmarks and the algorithms.

3.2.3 Transfer Learning

Transfer is a staple of all multi-task learning approaches. We expect multi-task learning to work because the valuable knowledge in one task can help solve another, i.e. can be *transferred* from the source task to the target task. Transfer learning is very similar to generalisation and is often considered part of generalisation studies. A standard way to separate the two is to use *generalisation* for in-distribution generalisation, i.e. sampling tasks from the same distribution, and *transfer* for out-of-distribution generalisation. However, it is hard to distinguish transfer from generalisation precisely. A standard way to do transfer learning is to train a model on the source task, stop training, and then train the resulting model on a target task, potentially after some transformations to the model.

Taylor et al. [TS09] give a substantial overview of transfer learning in pre-DQN RL. Importantly, they discuss how to measure the success of transfer learning and identify five different metrics: jumpstart, asymptotic performance, total reward, time to some return threshold, and the ratio of the total reward of a transferred model vs the model trained from scratch. They also provide a classification of the literature based on the allowed task differences, which might be helpful for us in terms of incompatibility. The paper reviews research on transferring knowledge between the environments with different state variables and actions. The authors split this work into two main groups: when no explicit mappings are built to transfer the knowledge and when there is one, which can either be provided or learnt. Konidaris et al. [KB06] is the most representative paper from the former group. They learn two representations when solving a task: *agent space* and *problem space*. Agent space is inherent to the agent, e.g. its physical configuration. Problem space is a part of the environment that changes from task to task, e.g. different map layouts. Agent space is shared across tasks and is used to shape the reward when learning on a new task, speeding up the learning in problem space.

One can also modify the model before being applied to the target domain. Taylor et al. [TWS07a] is an illustrative example of this approach. They introduce *inter-task mapping* [TS11], a technique in which there is a given mapping between source and target state and action sets. The mapping is given, and this is a limiting assumption in general. Taylor et al. [TWS07b] and Taylor et al. [TJS08] relax this assumption and learn the mapping between the tasks. In Taylor et al. [TKS08], the authors fit the transition model and simultaneously use it to construct the inter-task mapping. Banerjee et al. [BS07] learn abstract features useful in different games and do the value-function transfer showing positive results on games representable with a game tree.

Notably, modern RL research primarily considers transfer done without explicit knowledge about the target task. Current research covers a wide range of setups that differ in what parts of an MDP can change. Keeping dynamics the same is a popular choice partly due to its less challenging nature. For example, Barreto

et al. [Bar+17] study the properties of successor features for transfer across tasks with different reward functions but shared dynamics. Parisotto et al. [PBS16] go to another extreme and study transfer behaviour on totally different Atari 2600 games. They demonstrate positive and negative transfer behaviour of their distilled policies linking that to visual similarities and common game mechanics. We believe that, as we find in our transfer experiments in §6.5, transfer outcomes in the latter setting are more due to experimental variability rather than an algorithm's credit.

Most of the deep RL research investigates only compatible environments due to their convenience. Yu et al. [YLT19] train a policy that takes observations as inputs and parameters inducing the environmental dynamics. Later it uses the covariance matrix adaptation [Han06] to search for the best vector of parameters to use in the target task. Julian et al. [Jul+20] fine-tune a policy offline using the data from both the source and the target task. They initialise the target policy from the policy trained offline and then online on a real robot. For the target task, only initial data collection requires a robot. All the fine-tuning is done offline. The authors vary the background, lighting conditions, gripper shape, and robot morphology and add unseen objects to the scene. The number of sensors or actuators does not change. The authors also experiment with chaining several transfer and fine-tuning epochs, calling it *continual learning*. However, in contrast to the continual learning we discuss in §3.2.6, their agent is not expected to retain its past policies.

GNNs enabled studying transfer learning properties across incompatible environments by treating observations as graphs. Wang et al. [Wan+18] and Huang et al. [HMP20] study transfer properties of locomotion agents in incompatible environments (§5.2). They both report positive transfer behaviour of GNN-based policies and motivate our further research in Chapters 5 and 6. Recently, the follow-up works improved upon the method proposed in Chapter 6 adding physical agent information (e.g. limb sizes) [Gup+22] and vertex positional embeddings to the observation space [HYK22]. In Chapter 7, we demonstrate that a Q-function learnt as a branching heuristic of a SAT solver shows positive transfer behaviour when tested on a new target task distribution, including transfer to larger problems

and from satisfiable to unsatisfiable instances. Finally, we further discuss transfer learning in the incompatible setting in §3.3. For a broad recent survey on transfer in deep RL, we refer the reader to Zhu et al. [ZLZ20].

3.2.4 Meta-Learning

Meta-Learning is another approach leveraging training on multiple tasks to learn biases for the learning algorithm itself: the objective, hyperparameters of the learning algorithm, or the policy used as initialisation for a new task.

The tasks are expected to share similarities and are sampled from a continuous distribution over the parameters inducing environment dynamics (e.g. friction coefficient) or the reward function (e.g. target velocity or goal's coordinates). In contrast to multi-task training approaches, the model usually does not have access to the task identifier, making task identification a part of the problem.

The goal of MAML-like approaches [FAL17; Zin+19] is to learn such a model which will adapt to a new unseen task within a couple of gradient steps. The learning algorithm usually consists of inner and outer loops, with the first corresponding to updating the policy and the second corresponding to the meta-parameters, inducing the learning algorithm. The gradients are averaged in the parameter space, which is reasonable given that the goal is to obtain a model that will adapt as fast as possible to unseen tasks.

Hospedales et al. [Hos+22] gives an extensive overview of the existing research in deep meta-learning and can be used as a further reference. Curiously, the authors put MTL and transfer learning under the umbrella of meta-learning, which contrasts to our approach (meta-learning is a part of broader MTL term). 'Related Work' section of [Oh+20] also provides a good overview of approaches that dicover RL algorithms by meta-learning, i.e., learning the update rule itself.

Adding the support of incompatible environments to meta-learning algorithms is a challenging endeavour and deserves a separate DPhil project. Even with compatible environments, not many works try meta-learning on the discrete distribution of

tasks. Therefore, we consider meta-learning outside the scope of the current work providing this section for completeness and as an example of averaging the gradients in the parameter space.

3.2.5 Curriculum Learning

This section is a brief overview of curriculum learning in RL. On the one hand, curriculum learning is a generalisation of transfer learning, when an agent faces a sequence of tasks to master one target task. However, the sequence of tasks is supposed to have increasing difficulty, i.e. making the transfer easier. Here, we will describe the work most relevant to us, referring the reader to Narvekar et al. [Nar+20b] and Portelas et al. [Por+20] for a more comprehensive overview. We split curriculum learning into two broad categories based on whether the task distribution on hand is discrete or continuous.

3.2.5.1 Continuous task distribution

Automatic curriculum generation is a crucial problem in curriculum learning. When the task distribution is continuous, this often implies inferring such parameters inducing the environment dynamics, making learning faster. Svetlik et al. [Sve+17] build a curriculum as an acyclic directed graph rather than a linear schedule which is a default option in most research. Florensa et al. [Flo+17] generate a curriculum gradually moving the initial position of the agent further from the goal. Molchanov et al. [Mol+18] is another similar work considering a distribution over initial states and a distribution over goals as curriculum components. Florensa et al. [Flo+18] use a goal generator network to propose the next goal for an agent in the environment, focusing on those goals which are more challenging for an agent to reach but not hard enough to make the reward impossible to get. Portelas et al. [Por+19] consider multi-armed bandits and Gaussian mixture models to decide on the regions of the parameter space inducing the environment to design the curriculum setting automatically.

Klink et al. [Kli+20] approach curriculum learning as an inference problem, bringing the task distribution closer to the target task distribution (not a single target task). By *curriculum learning*, they imply 'evolving the task along with the learner'. This setup is closer to domain adaptation (§3.2.2) requiring data from the target task distribution. They use the framework of Contextual MDPs [HCM15], which is a distribution over MDPs with the shared state-action space. Zhang et al. [ZAP20] apply curriculum learning for goal sampling in the goal-directed RL with shared state and action sets. The authors base their approach on the epistemic uncertainty of the value function, sampling the goal at the frontier of a collection of goals the agent can reach.

Bassich et al. [Bas+20] approach curriculum learning using *progression functions* and *mapping functions*. Progression functions define the complexity of the next task to consider and are bounded between zero and one, with one being the complexity of the most complex task, i.e., the target task. Mapping functions generate the environment given the complexity, i.e., generating an agent's initial position in a maze at a certain distance (the further, the harder). Similarly, Liu et al. [LPK22] solve a transfer learning problem with the curriculum learning approach, interpolating the environmental parameters between the source and target tasks, gradually moving closer to the target task in the parameters inducing the environmental dynamics.

Finally, Wang et al. [Wan+19] use coevolution to generate a set of more and more challenging environments while training the policy. This paper is closer to the *continual learning* setting having no target task on hand. It is an open-ended task discovery, where the goal might be to discover a task in addition to finding an optimal policy for it.

3.2.5.2 Discrete task distribution

Discrete curricula settings generally have no assumption on task similarities and are thus much harder than the continuous counterpart. On the other hand, the

discrete nature of the problem allows using multi-armed bandits to select a task for an update making it easier to optimise.

Sinapov et al. [Sin+15] select a source task for transfer from a pool, estimating intertask transferability from a feature vector describing each task, e.g., number of enemies in the level or nodes in the maze graph. They also perform chain transfer designing a curriculum for the target task. Jain et al. [JT17] consider multiple options for curriculum design, including choosing the task where an agent is making the fastest progress or estimating the transfer between task pairs and greedily selecting the next task. In contrast to Sinapov et al. [Sin+15], the curriculum is designed concurrently with training.

Lopes et al. [LO12] consider a *strategic student problem* in which a learner has to optimise the average performance across multiple tasks given limited data using multi-armed bandits to sequentially select a task to learn from. This paper has an extensive literature review on the subject and might be extremely useful for considering the sample efficiency of MTRL.

Narvekar et al. [NSS17] pose a curriculum problem as an MDP minimising the amount of time needed to achieve a given return on a target task. Sharma et al. [Sha+18] consider multi-armed bandits for the Atari 2600 benchmark [Bel+13] selecting the most *underperforming* task. Their main limitation is that one needs to specify a reference score to define what an *underperforming* task is. For that, one either needs to train single-task models or to get the numbers from another paper. Matiisen et al. [Mat+20] use ideas from the non-stationary multi-armed bandits, favouring tasks making the most progress. Their setup consists of one target task and several auxiliary tasks. The authors assume that success on auxiliary tasks implies good performance on the target task. The authors achieve comparable results with their automatic curriculum to the human-designed curriculum regime. Mysore et al. [MPS] use bandits and EXP3 [Aue+02] to sample a task for an update using the clipped and normalised change in the returns as bandit rewards. This paper is similar to Matiisen et al. [Mat+20], but Mysore et al. [MPS] directly update the

bandits, whereas Matiisen et al. [Mat+20] use some intermediate steps, e.g. fitting a linear regression to estimate the change in the returns before and after an update.

Curriculum learning has been applied to automatic theorem proving with RL. Bansal et al. [Ban+19] and Zombori et al. [Zom+19] control the difficulty of the problem by starting from intermediate steps of the proof, similarly to Florensa et al. [Flo+17]. Polu et al. [Pol+22] vary the complexity of the statements and their composition depth to generate a curriculum.

3.2.6 Continual Learning

This section also considers sequential learning of tasks. However, in contrast to curriculum learning, an agent must retain its ability to solve past tasks. Continual learning [Rin95; Thr95] is challenging in deep learning since neural networks are known to forget past experiences, a phenomenon known as catastrophic forgetting [MC89].

Rusu et al. [Rus+16b] attack the problem of forgetting policies for past tasks via retaining them and learning extra layers. Kirkpatrick et al. [Kir+16] attacks the problem of catastrophic forgetting in a sequential learning regime keeping the policy in the regions of the parameter space which do not let policies move far away in the behaviour space, low loss regions in this case. The results of the paper go in line with the success of RL algorithms working in the behaviour space (NPG, TRPO, PPO) and policy distillation results (§3.2.1.2): controlling the behaviour space is easier than managing the parameter space.

Surprisingly, even tasks considered to be a single RL environment, in the presence of function approximation, might exhibit catastrophic forgetting and have the problems inherent to multi-task setting [Fed+20]. We find this work vital to realising that the boundary between single-task learning and multi-task learning is not sharply defined, a view that we emphasise in Chapter 4.

This thesis does not directly contain work on continual learning, and we refer the reader to an exhaustive survey on the topic [Khe+20b]. However, we believe

that this line of research is essential to understanding the weaknesses of the multi-task training settings when the tasks are updated sequentially, as in Huang et al. [HMP20] we consider in Chapter 6.

3.2.7 Auxiliary Tasks

Adding auxiliary tasks to facilitate learning for the target task is the last research direction we will cover in this section.

The first group introduces non-RL auxiliary tasks to solve in a supervised or unsupervised way. Lin et al. [Lin+19] uses differences in the losses for the target task before and after an update as an objective and tries to maximise it w.r.t. weights of the auxiliary task losses. The idea is to channel the update toward the primary task loss decrease. The auxiliary tasks include predicting forward dynamics and optical flow. Other works consider depth and loop closure prediction[1] [Mir+17], reward, dynamics prediction and state reconstruction [She+17].

We believe, that such auxiliary tasks might be important to graph-based reinforcement learning we investigate in Chapters 5, 6 and 7. GNNs are poorly understood, and unsupervised pretraining might facilitate learning of message passing [Hu+20]. The success of denoising objectives in GNN-based supervised learning Godwin et al. [God+22] can be transferrable to the RL setting or even be more helpful in the presence of data nonstationarity inherent to deep reinforcement learning [Igl+21].

The second group adds additional MDPs as auxiliary tasks. This group overlaps greatly with curriculum learning (§3.2.5). However, in this approach, the tasks are usually jointly optimised. Sutton et al. [Sut+11] propose learning a set of value functions, one per a pseudo-reward called *cumulants*. Jaderberg et al. [Jad+17] further develop the idea, introducing weight sharing to the method and testing it in the deep RL setting. They achieve high performance controlling an agent in a 3D environment when using reward prediction and maximising change in the pixel

[1] Loop closure aims to predict whether the agent has previously visited the location within a current episode.

density of the observational inputs as auxiliary tasks. Riedmiller et al. [Rie+18] use auxiliary tasks inducing a reward vector and learn a separate policy for each of the vector elements leading to better exploration and data efficiency. Several recent works investigate the mechanisms behind auxiliary tasks [Bel+19; Lyl+21; Dab+21] relating their effectiveness to shaping better representations and regularisation.

3.3 Incompatible Reinforcement Learning

A significant part of this thesis deals with incompatible RL described in §2.2. This section gives an overview of the related work, including the review of incompatible environments in §3.3.1.

Due to convenience, most MTRL research considers the compatible case [Rus+16a; PBS16; Teh+17; VM20]. MTRL for continuous control is often done from pixels, with CNNs partly solving the compatibility issue. DMLab [Bea+16] is a popular choice when learning from pixels with a compatible action space shared across the environments [Hes+19; Son+20].

There are few ways of dealing with environment incompatibility when using function approximation. In our opinion, GNNs are the most principled and general approach capable of combining the flexibility of deep learning with an ability to inject desired inductive biases into a model [Bat+18].

While graph-based learning is a relatively young area of research, there is already a vast body of work in this field, including many surveys [Zho+20; Bat+18; ZCZ18; Wu+21; Bro+17b; Don+20]. In this thesis, we adopt the message-passing approach to GNNs, using the formalism of Battaglia et al. [Bat+18], which unifies many existing spatial-based approaches and provides a convenient framework from both algorithmic and implementation perspective [HM20]. We give a detailed description of the formalism in §2.3. Spectral-based approaches [DBV16; KW17] are another popular direction in GNN research relying on strong foundation of

signal processing [Don+20]. Due to time constraints, we do not cover spectral-based GNN approaches in this thesis, and we refer the reader to an extensive survey by [Wu+21] (Section V.A).

There has recently been a growth in using GNNs in the RL community due to their ability to accommodate arbitrary input and output sizes. While GNNs are still poorly understood and, using them as function approximators in RL is a challenging endeavour, they have already started to stretch the possibilities of RL allowing MTRL in incompatible environments. Khalil et al. [Kha+17] learn combinatorial optimisation algorithms over graphs. Kurin et al. [Kur+20] learn a branching heuristic of a SAT solver. In these settings, applying conventional neural networks is impossible because they expect input and output to be of fixed size. Another form of (potentially incompatible) RL using message passing are coordination graphs [e.g. DCG, BKW20], that use the max-plus algorithm [Pea89] to coordinate action selection between multiple agents. One can apply DCG in single-agent RL using ideas of Tavakoli et al. [TFK21].

Several other methods for incompatible continuous control have also been proposed. Chen et al. [CMG18] pad the state vector with zeros to have the same dimensionality for robots with different numbers of joints and condition the policy on the hardware information of the agent. D'Eramo et al. [DEr+20] demonstrate a positive effect of learning a shared network for multiple tasks, learning a specific encoder and a decoder per task. While a simple and practical approach, its sample inefficiency will suffer when the number of tasks is large or generalisation to other environments is desirable. Wang et al. [Wan+18] have a similar implementation of their baseline, showing that GNNs have benefits over MLPs for incompatible control. SMP [HMP20], whose work is the main baseline in Chapter 6, apply a GNN-like approach and study its MTRL and generalisation properties. One of the significant limitations of SMP is that it operates only on tree-like graphs. Wang et al. [Wan+18] and Huang et al. [HMP20] attribute the effectiveness of their methods to the ability

of the GNNs to exploit information about agent morphology[2,3]. However, Chapter 6 presents evidence against this hypothesis, showing that existing approaches do not exploit morphological information as was previously believed.

Attention mechanisms have also been used in the RL setting. Zambaldi et al. [Zam+18] consider self-attention to deal with an object-oriented state space. They further generalise this to variable action spaces and test generalisation on Starcraft-II mini-games that have a varying number of units and other environmental entities. Duan et al. [Dua+17] apply attention to both temporal dependency and a factorised state space (different objects in the scene), keeping the action space compatible.

Parisotto et al. [Par+20] use transformers to replace a recurrent policy. Loynd et al. [Loy+20] use transformers to add history dependence in partially observable MDPs and for factored observations, having a node per game object. The authors do not consider a factored action space, with the policy receiving the aggregated information of the graph after the message passing ends. Baker et al. [Bak+20] use self-attention to account for a factored state-space to attend over objects or other agents in the scene. Amorpheus, a method we propose in Chapter 6, does not use a Transformer for recurrency but the factored state and action spaces, with each non-torso node having an action output. Iqbal et al. [IS19] apply attention to generalise MTRL multi-agent policies over varying environmental objects and Iqbal et al. [Iqb+20] extend this to a factored action space by summarising the values of all agents with a mixing network [Ras+20]. Li et al. [Li+20] learn embeddings for a multi-agent actor-critic architecture by generating the weights of a graph convolutional network [GCN, KW17] with attention allowing a different topology in every state.

Another line of work aims to infer graph topology instead of hardcoding one. Differentiable Graph Module [Kaz+20] predicts edge probabilities doing a continuous

[2]Huang et al. [HMP20]: '*Exceptions are works of Pathak et al. (2019); Wang et al.(2018), which similarly to our work exploit graph structure present in the agent's morphology*'

[3]Wang et al. [Wan+18]: '*NerveNet can leverage the structure information encoded by the agent's body which is advantageous in learning the correct inductive bias, and thus is less prone to overfitting.*'

relaxation of k-nearest neighbours to differentiate the output with respect to the edges in the graph. Johnson et al. [JLT20] learn to augment a given graph with additional edges to improve the performance of a downstream task. Kipf et al. [Kip+18] use variational autoencoders [KW14] using a GNN for reconstruction. Notably, the authors notice that message passing on a fully connected graph might work better than when restricted by skeleton when evaluated on human motion capture data.

3.3.1 Incompatible Environments

This chapter reviews the existing environments suitable for incompatibility research. Due to their inconvenience for deep reinforcement learning, incompatible environments have been largely ignored by the RL community. Notably, Yu et al. [Yu+19] explicitly state:

> `In our benchmarks, some tasks have different numbers of objects, but the state dimensionality is always the same, meaning that some state coordinates are unused for some tasks.`

3.3.1.1 Mainstream Single-Task environments

MuJoCo [TET12] constitute a set of incompatible environments when trained on multiple tasks simultaneously. Sanchez-Gonzalez et al. [San+18] use GNNs to learn transition models of MuJoCo environments and use them for model-predictive control. Wang et al. [Wan+18] design a set of Centipede MuJoCo environments where a user can vary the number of legs and impair the centipede, e.g., restricting the movements of the last two legs. They also provide five walkers environments for multi-task training experiments. Huang et al. [HMP20] design a separate set of incompatible environments based on MuJoCo walkers. We use benchmarks from Wang et al. [Wan+18] and Huang et al. [HMP20] for our work on incompatible control in Chapters 5 and 6.

3.3.1.2 Object-Oriented Environments

As we mentioned, any environment providing a state in an object-oriented way can be a good candidate for incompatible MTRL. Bapst et al. [Bap+19] propose a Unity [Jul+18] environment testing abilities of the agents to do construction tasks. Küttler et al. [Küt+20, NetHack Learning Environment] allows accessing the environment state as a Python dictionary rather than a tensor of a fixed size. NetHack is inherently multi-modal (textual description of the environment, image-like dungeon map, a real-valued vector of agent stats), which provides another form of incompatibility in a reinforcement learning environment. In Samvelyan et al. [Sam+21], we build upon NetHack making a sandbox environment suitable for testing specific skills of RL agents. Interestingly, the Nethack Challenge [Cha21] we organised at NeurIPS 2021 showed that current RL methods are lagging behind symbolic methods [Ham+21] in this setting. We believe that the performance gap is at least partially a result of the community's lack of attention to incompatible environments.

3.3.1.3 Scientific Problems as RL environments

Many conventional computer science problems are, in fact, incompatible environments. Bello et al. [Bel+17] consider combinatorial optimisation problems with RL, showing results on the Travelling Salesman Problem and the Knapsack Problem. Khalil et al. [Kha+17] approach combinatorial optimisation using GNNs and DQN, learning a heuristic that is later used greedily. Paliwal et al. [Pal+20] use GNNs with imitation learning for theorem proving. Carbune et al. [Car+18] propose a general framework for injecting an RL agent into existing algorithms. Yolcu et al. [YP19] use REINFORCE [Wil92] to learn the variable selection heuristic of a local search SAT solver [SKC93]. Wang et al. [WR18] turn the MiniSat [SE05] solver into an RL environment. In Kurin et al. [Kur+20], we further develop their environment, making it possible to use with any SAT problem (vs random 3 SAT in the original version) and providing graph representation of states. Simm et al.

[SPH20] provide an environment for molecular design in which one has to pick an atom and put it on a three-dimensional canvas.

3.3.1.4 Multi-agent Reinforcement Learning Environments

Multi-agent reinforcement learning works with environments that might be incompatible with different combinations of agents.

Many existing StarCraft papers concatenate all agents' features into a single multidimensional array, zeroing entities outside of the receptive field of the agents. This has recently changed with GNNs entering the field [JDL18; Mal+18; Aga+20]. Iqbal et al. [Iqb+20] use transformers in the setting testing a learned model on different agent formations.

While many intriguing problems arise in multi-agent reinforcement learning, we restrict ourselves to the single-task case due to a DPhil project's time and resource constraints.

Simplicity is a great virtue, but it requires hard work to achieve it and education to appreciate it. And to make matters worse: complexity sells better.

— Edsger Dijkstra

4

In Defense of the Unitary Scalarisation for Deep Multi-Task Learning

Contents

4.1 Introduction

4.1.1 Motivation

MTL [Car97b] exploits similarities between tasks to yield models that are more accurate, generalise better and require less training data. Owing to the success of MTL on traditional machine learning models [Hes00; BH03; EP04] and of deep single-task learning across a variety of domains, a growing body of research has focused on deep MTL. The most straightforward way to train a neural network for multiple tasks at once is to minimise the sum of per-task losses. Adopting terminology from multi-objective optimisation, we call this approach *unitary scalarisation*.

While some work shows that multi-task networks trained via unitary scalarisation exhibit performance superior to independent per-task models [Kok17; Kal+21], others suggest the opposite [Teh+17; KGC18; SK18]. As a result, many explanations for the difficulty of MTL have been proposed, each motivating a new Specialised Multi-Task Optimiser (SMTO) [SK18; Liu+21b; Yu+20; Che+20; Wan+21]. These works typically claim that the proposed SMTO outperforms unitary scalarisation. However, SMTOs usually require access to per-task gradients either with respect to the shared parameters or the shared representation. Therefore, their reported performance gain comes at significant computation and memory cost, the overhead scaling linearly with the number of tasks. By contrast, unitary scalarisation requires only the average of the gradients across tasks, which can be computed via a single backpropagation sweep.

4.1.2 Contribution

Existing SMTOs were introduced to solve challenges related to the optimisation of the deep MTL problem. We instead postulate that the reported weakness of unitary scalarisation is linked to experimental variability or a lack of regularisation, leading

to the following contributions. We perform a comprehensive experimental evaluation (§4.3) of recent SMTOs on popular multi-task benchmarks, showing that no SMTO consistently outperforms unitary scalarisation despite the added complexity and overhead. In particular, either the differences between unitary scalarisation and SMTOs are not statistically significant, or they can be bridged by standard regularisation and stabilisation techniques from the single-task literature. Our RL experiments include optimisers previously applied only to supervised learning. We publicly release a unified PyTorch [Pas+19] codebase for all SMTOs in the experimental evaluation, including one whose implementation was previously unpublished [Liu+21b], available at https://github.com/yobibyte/unitary-scalarization-dmtl.

We believe that our results suggest that the considered SMTOs can often be replaced by less expensive techniques. We hope that these surprising results stimulate the search for a deeper understanding of MTL.

This chapter is based on Kurin et al. [Kur+22], to which Alessandro de Palma made an equal contribution. The author of this thesis contributed to the empirical part of the work, primarily the RL section, and Alessandro worked on the theoretical side of the project and supervised learning experiments. Both first authors took part in defining the problem, software engineering, project management, and writing the paper. All the other authors played significant advisory roles and contributed to writing individual sections of the paper.

4.2 Multi-Task Learning Optimisers

We will now describe common algorithms employed for solving the MTL problem stated in Equation (2.1).

4.2.1 Unitary Scalarisation

An obvious way to minimise the multi-task training objective in Equation (2.1) is to rely on a standard gradient-based algorithm. While, for simplicity, we focus on

standard gradient descent rather than mini-batch stochastic gradient descent, the notation can be adapted by replacing the dataset size n with the mini-batch size b.

Equation (2.1) corresponds to a linear scalarisation with unitary weights under a multi-objective interpretation of MTL [SK18]; hence, we call the direct application of gradient descent on Equation (2.1) *unitary scalarisation*. For vanilla gradient descent, this corresponds to taking a step in the opposite direction as the one given by the sum of per-task gradients: $\nabla_{\boldsymbol{\theta}} \mathcal{L}^{\mathrm{MT}} = \sum_{i \in \mathcal{T}} \nabla_{\boldsymbol{\theta}} \mathcal{L}_i$.

Per-task gradients are not required, as it suffices to directly compute the gradient of the sum $\mathcal{L}^{\mathrm{MT}}$. Hence, when relying on deep learning frameworks based on reverse-mode differentiation, such as PyTorch [Pas+19], the backward pass is performed once per iteration (rather than m times). Furthermore, the memory cost is a factor m less than most SMTOs, which require access to each $\nabla_{\boldsymbol{\theta}} \mathcal{L}_i$. Consequently, unitary scalarisation is simple, fast, and memory efficient. Our experiments in §4.3 demonstrate that, when possibly coupled with single-task regularisation such as early stopping, ℓ_2 penalty or Dropout layers [Sri+14], this simple optimiser is strongly competitive with SMTOs.

4.2.2 MGDA

Sener et al. [SK18] point out that Equation (2.1) can be cast as a multi-objective optimisation problem with the following objective: $\boldsymbol{\mathcal{L}}^{\mathrm{MT}}(\boldsymbol{\theta}) := [\mathcal{L}_1(\boldsymbol{\theta}), \dots, \mathcal{L}_m(\boldsymbol{\theta})]^T$. A commonly employed solution concept in multi-objective optimisation is Pareto optimality. A point $\boldsymbol{\theta}^*$ is called Pareto-optimal if, for any another point $\boldsymbol{\theta}^\dagger$ such that $\exists i \in \mathcal{T} : \mathcal{L}_i(\boldsymbol{\theta}^\dagger) < \mathcal{L}_i(\boldsymbol{\theta}^*)$, then $\exists j \in \mathcal{T} : \mathcal{L}_j(\boldsymbol{\theta}^\dagger) > \mathcal{L}_j(\boldsymbol{\theta}^*)$. Sener et al. [SK18] rely on MGDA [Dés12] to reach a Pareto-stationary point for shared parameters $\boldsymbol{\theta}_{\|}$, i.e. a point for which there exists a zero-valued convex combination of the gradients at this point. MGDA takes steps in a direction that decreases the loss of all tasks at once [FS00; Dés12], which can be found by solving the following optimisation problem:

$$\min_{\mathbf{g}, \epsilon} \left[\epsilon + \frac{1}{2} \|\mathbf{g}\|_2^2 \right] \quad \text{s.t.} \quad \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i^T \mathbf{g} \leq \epsilon \quad \forall\, i \in \mathcal{T}, \tag{4.1}$$

whose dual takes the following form (corresponding to the formulation from Désidéri [Dés12]):

$$\min_{\boldsymbol{\alpha}} \frac{1}{2} \|\mathbf{g}\|_2^2 \quad \text{s.t.} \quad \mathbf{g} = -\sum_i \alpha_i \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i, \quad \sum_{i \in \mathcal{T}} \alpha_i = 1, \quad \alpha_i \geq 0, \quad i \in \mathcal{T}. \tag{4.2}$$

In other words, MGDA takes a step in a direction $\mathbf{g}$ given by the negative convex combination of per-task gradients, whose coefficients are given by solving Equation (4.2). The original authors' implementation [SK18] rescales per-task gradients before applying MGDA[1]:

$$\nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i \leftarrow \frac{\nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i}{\left\| \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i \right\| \mathcal{L}_i(\boldsymbol{\theta})}. \tag{4.3}$$

The authors do not discuss the motivation behind rescaling in the paper, and we believe that the reason is mostly empirical. The convergence of MGDA to a Pareto-optimal point is still guaranteed after normalisation [Dés12].

4.2.3 IMTL

Impartial multi-task learning (IMTL) [Liu+21b] is presented as an SMTO that is not biased against any single-task. It comprises two complementary algorithmic blocks: IMTL-L, acting on task losses, and IMTL-G, acting on per-task gradients. IMTL-G follows the intuition that a multi-task optimiser should proceed along a direction $\mathbf{g} = -\sum_i \alpha_i \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i$ that equally represents per-task gradients. This is formulated analytically by requiring the cosine similarity between $\mathbf{g}$ and each $\nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i$ be the same. To prevent the resulting problem from being underdetermined, Liu et al. [Liu+21b] add the constraint $\sum_{i \in \mathcal{T}} \alpha_i = 1$, resulting in a problem that admits a closed-form solution for $\mathbf{g}$:

$$\mathbf{g}^T \frac{\nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_1}{\left\| \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_1 \right\|} = \mathbf{g}^T \frac{\nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i}{\left\| \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i \right\|} \;\; \forall \; i \in \mathcal{T} \setminus \{1\}, \quad \mathbf{g} = -\textstyle\sum_i \alpha_i \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i, \quad \textstyle\sum_{i \in \mathcal{T}} \alpha_i = 1. \tag{4.4}$$

IMTL-L, instead, aims to reweight task losses so that they are all constant over time and equal to 1. Furthermore, to limit the scaling factors' oscillations, the authors

[1] https://github.com/isl-org/MultiObjectiveOptimization/blob/d45eb262ec61c0dafecebfb69027ff6de280dbb3/multi_task/train_multi_task.py#L158

propose to learn them jointly with the network by minimising a common objective via gradient descent. In particular, given $s_i \in \mathbb{R}\ \forall i \in \mathcal{T}$, Liu et al. [Liu+21b] derive the following form for the joint minimisation problem: $\min_{\mathbf{s},\boldsymbol{\theta}} \left[\sum_i \left(e^{s_i}\mathcal{L}_i(\boldsymbol{\theta}) - s_i\right)\right]$. As proved by Liu et al. [Liu+21b], IMTL-L only has a rescaling effect on the update direction of IMTL-G. Unlike IMTL-G and the other SMTOs presented in this section, IMTL-L rescaling is designed to affect the updates for task-specific parameters $\boldsymbol{\theta}_\perp$ as well.

4.2.4 PCGrad

Let us write $\cos(\mathbf{x}, \mathbf{z})$ for the cosine similarity between vectors $\mathbf{x}$ and $\mathbf{z}$. Yu et al. [Yu+20] postulate that multi-task convergence is severely slowed down if the following three conditions (named the *tragic triad*) hold at once: (i) conflicting gradient directions: $\cos(\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_i, \nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_j) < 0$ for some $i, j \in \mathcal{T}$; (ii) differing gradient magnitudes: $\left\|\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_i\right\| \gg \left\|\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_j\right\|$ for some $i, j \in \mathcal{T}$; and (iii) the unitary scalarisation $\mathcal{L}^{\text{MT}}$ has high curvature along $\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}^{\text{MT}}$. The PCGrad [Yu+20] SMTO is presented as a solution to the tragic triad, targeted at the first condition. Consistent with the previous sections, let us denote the update direction by $\mathbf{g}$. Furthermore, let $[\mathbf{x}]_+ := \max(\mathbf{x}, \mathbf{0})$. Given per-task gradients $\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_i$, PCGrad iteratively projects each task gradient onto the normal plane of all the gradients with which it conflicts:

$$\begin{gathered}\left[\mathbf{g}_i \leftarrow \nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_i,\ \ \mathbf{g}_i \leftarrow \mathbf{g}_i + \left[\frac{-\mathbf{g}_i^T \nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_j(\mathbf{x})}{\left\|\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_j\right\|^2}\right]_+ \nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_j\ \ \forall j \in \mathcal{T} \setminus \{i\}\right] \forall i \in \mathcal{T}, \\ \mathbf{g} = -\sum_{i \in \mathcal{T}} \mathbf{g}_i,\end{gathered} \tag{4.5}$$

where the iterative updates of $\mathbf{g}_i$ with respect to $\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_j$ are performed in random order.

4.2.5 GradDrop

Chen et al. [Che+20] focus on conflicting signs across task gradient entries, arguing that such conflicts lead to gradient "tug-of-wars". The GradDrop SMTO [Che+20], presented as a solution to this problem, proposes to randomly mask per-task

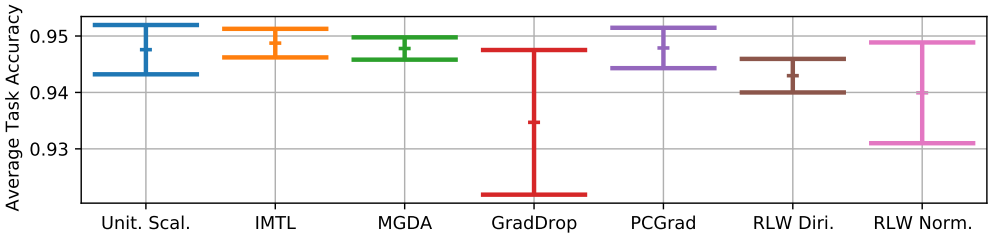
gradients $\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i$ so as to minimize such conflicts. Specifically, GradDrop computes the "positive sign purity" p_j for the task gradient's j-th entry and then masks the j-th entry of each per-task gradient with the probability increasing with p_j if the entry is negative or decreasing with p_j if the entry is positive. Let us write $\mathbf{p} := [p_1, \dots, p_S]$, where S is the dimensionality of the parameter space (see §4.2), $\odot$ for the Hadamard product and $\mathbb{1}_{\mathbf{a}}$ for the indicator vector on condition $\mathbf{a}$. Given a vector $\mathbf{u}_i$, uniformly sampled in $[\mathbf{0}, \mathbf{1}]$ at each iteration, GradDrop takes a step in the direction given by:

$$\begin{aligned} \mathbf{g} &= -\sum_{i\in\mathcal{T}} \nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i \odot \left(\mathbb{1}_{\left(\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i > 0\right)} \odot \mathbb{1}_{(\mathbf{u}_i > \mathbf{p})} \;+\; \mathbb{1}_{\left(\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i < 0\right)} \odot \mathbb{1}_{(\mathbf{u}_i < \mathbf{p})} \right), \text{ with} \\ \mathbf{p} &= \frac{1}{2}\left(1 + \frac{\sum_{i\in\mathcal{T}} \nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i}{\sum_{i\in\mathcal{T}} \left|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i\right|}\right). \end{aligned} \tag{4.6}$$

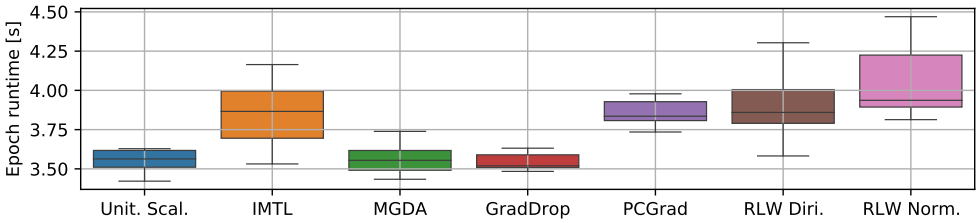
4.3 Experimental Evaluation

Relying on a unified experimental pipeline, we present an empirical evaluation on common MTL benchmarks of unitary scalarisation (§4.2), of the popular SMTOs presented in §4.2, and of the recent Random Loss Weighting (RLW) algorithms [LYZ21] due to their similarities with PCGrad and GradDrop (see §4.4.2). In particular, we benchmark against the two RLW instances that showed the best average performance in the original paper: RLW with weights sampled from a Dirichlet distribution ("RLW Diri."), and RLW with weights sampled from a Normal distribution ("RLW Norm."). Whenever appropriate, we employ "Unit. Scal." as shorthand for unitary scalarisation. We first present supervised learning experiments (§4.3.1), and then evaluate on a popular reinforcement learning benchmark (§4.3.2).

Our experiments indicate that the performance of unitary scalarisation has been consistently underestimated in the literature. By showing the variability between runs and relying on standard regularisation and stabilisation techniques from the single-task literature, we demonstrate that *no SMTO consistently outperforms unitary scalarisation across the considered settings*. This result holds despite the added complexity and computational overhead associated with most SMTOs. We provide a potential explanation of our results in §4.4.



(a) Avg. task test accuracy: mean and 95% CI (10 runs).



(b) Box plots for the training time of an epoch (10 runs).

Figure 4.1: Despite larger overheads, no algorithm outperforms unitary scalarisation on the Multi-MNIST dataset.

4.3.1 Supervised Learning

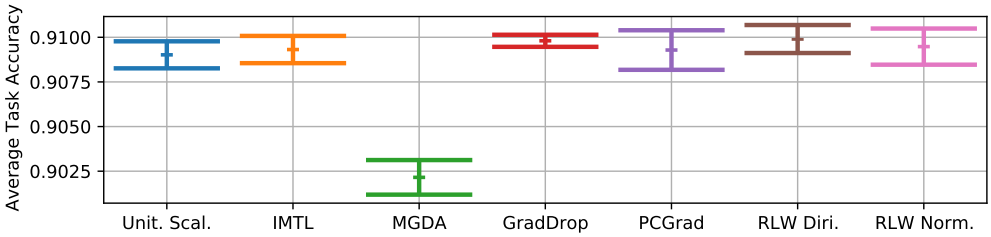
All the architectures employed in the supervised learning experiments conform to the encoder-decoder structure detailed at the beginning of Chapter 2. Whenever suggested by the original authors for this context, the SMTO implementations rely on per-task gradients with respect to the last shared activation, $\nabla_{\mathbf{z}}$, rather than on the usually more expensive $\nabla_{\boldsymbol{\theta}}\mathcal{L}_i$. In particular, this is the case for MGDA, IMTL and GradDrop. See Appendix B for details concerning each individual algorithm. Surprisingly, several MTL works [Yu+20; Che+20; Liu+21b; LYZ21] report validation results, making it easier to overfit. Instead, following standard machine learning practice, we select a model on the validation set and later report test metrics for all benchmarks. Validation results are also available in Appendix A.1.3. Appendix A.1.1.1 describes the tasks, and reports the computational setup, hyperparameter and tuning details.

4.3.1.1 Multi-MNIST

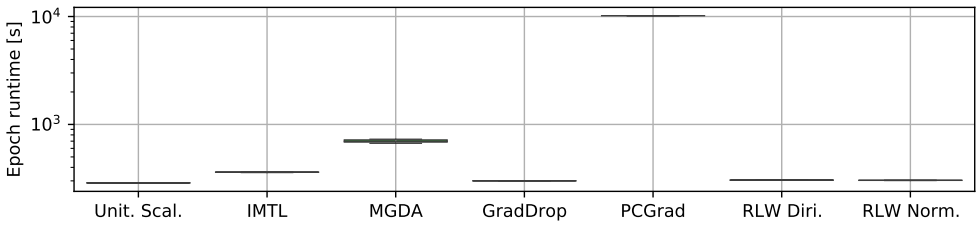
We present results on the Multi-MNIST [SK18] dataset, a simple two-task supervised learning benchmark. We employ a popular architecture from previous work [SK18; Yu+20] (see Appendix A.1.1.1), where a single Dropout layer [Sri+14] (with 0.5 Dropout probability) is employed in both the encoder and the decoder. ℓ_2 regularisation did not improve validation performance and was therefore omitted. Figure 4.1 reports the average task test accuracy and the training time per epoch. For each run, the test model was selected as the model with the largest average task validation accuracy across the training epochs. Appendix A.1.3 presents the results of Figure 4.1 in tabular form, as well as the average task validation accuracy per epoch. As seen from the overlapping confidence intervals, none of the considered algorithms clearly outperforms the others. However, GradDrop displays higher experimental variability. Finally, Figure 4.1b shows that unitary scalarisation also has among the lowest training times.

4.3.1.2 CelebA

We now show results for the CelebA [Liu+15] dataset, a challenging 40-task multi-label classification problem. We employ the same network architecture as many previous studies [SK18; Yu+20; LYZ21; Liu+21b] (see Appendix A.1.1.1). We tuned ℓ_2 regularisation terms λ for all SMTOs in the following grid: $\lambda \in \{0, 10^{-4}, 10^{-3}\}$. The best validation performance was attained with $\lambda = 10^{-3}$ for unitary scalarisation, IMTL and PCGrad, and with $\lambda = 10^{-4}$ for MGDA, GradDrop, and RLW. The addition of several Dropout layers further stabilised validation performance (see Figure 4.7), with Dropout probabilities from 0.25 to 0.5. We present an ablation study on the effect of regularisation on this experiment in §4.4.1. Figure A.4 (Appendix A.1.3.2) shows that regularisation improves the peak average validation performance for all the considered methods. Analogously to our Multi-MNIST results, Figure 4.2 plots the distribution of the training time per epoch and the average test task accuracy. Again, as with Multi-MNIST, the test model for each



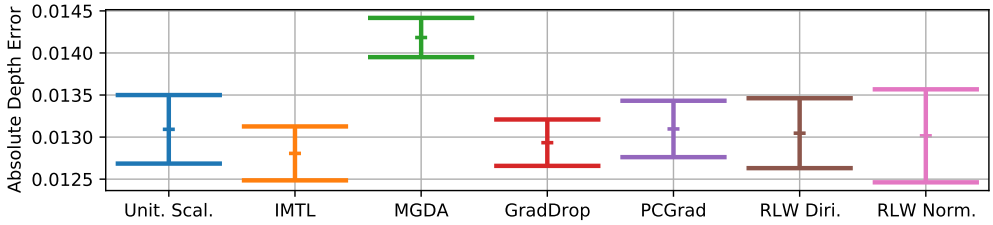
(a) Avg. task test accuracy: mean and 95% CI (3 runs).



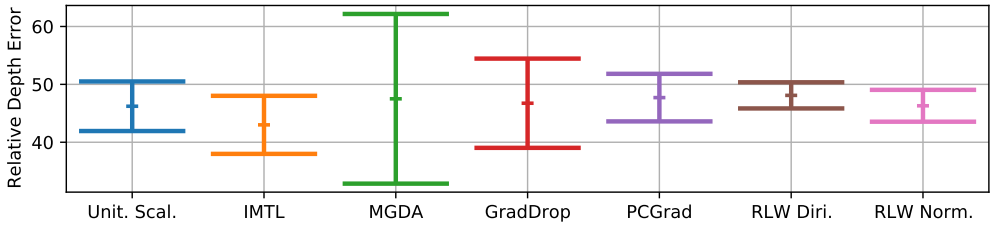
(b) Box plots for the training time of an epoch (10 runs).

Figure 4.2: While SMTOs display larger runtimes, none of them outperforms the unitary scalarisation on the CelebA dataset. Due to computational constraints, Figure 4.2a aggregates 3 repetitions only.

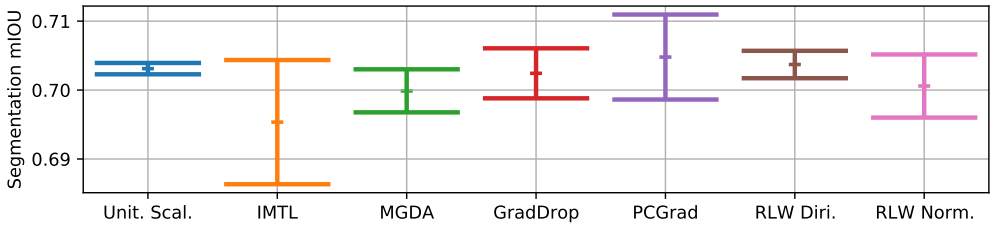
run was the one with maximal average validation task accuracy across epochs. In other words, if the peak is attained before the last epoch, we perform early stopping: as shown in Figure A.2a in Appendix A.1.3, this is the case for most methods. Due to a large number of tasks, Figure 4.2b shows relatively large runtime differences across methods. PCGrad is the slowest (roughly 35 times slower than unitary scalarisation). In fact, amongst the considered algorithms, it is the only one that computes per-task gradients over the parameters ($\nabla_{\boldsymbol{\theta}} \mathcal{L}_i \ \forall i \in \mathcal{T}$) at each iteration. GradDrop, MGDA and IMTL have overhead factors (compared to unitary scalarisation) ranging from roughly 1.05 to 2.4 due to the relatively small size of $\mathbf{z}$ for the employed architecture. The overhead of RLW is negligible: roughly 5%. Nevertheless, due to largely overlapping confidence intervals in Figure 4.2a, none of the methods consistently outperforms unitary scalarisation. In fact, owing to our adoption of explicit regularisation techniques (see §4.4.1), its average performance is superior to that reported in the literature [SK18; Liu+21b].



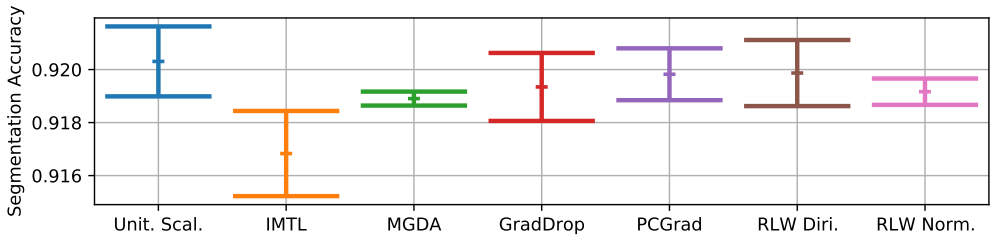
(a) Absolute depth test error: lower is better.



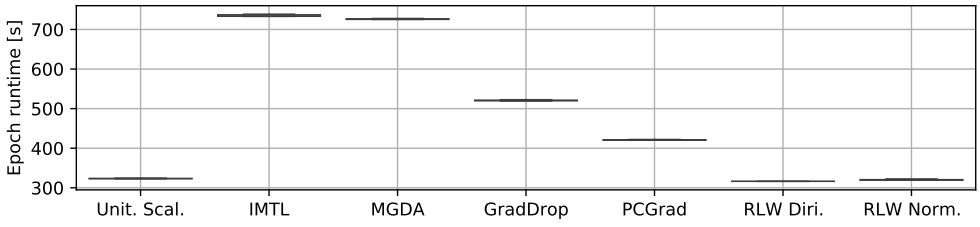
(b) Relative depth test error: lower is better.



(c) Test segmentation mIOU: higher is better.



(d) Test segmentation accuracy: higher is better.



(e) Box plots for the training time of a single epoch (10 repetitions).

Figure 4.3: On Cityscapes, none of the SMTOs outperforms unitary scalarisation, which proves to be the most cost-effective algorithm. Subfigures a-d report means for three runs, and their 95% CIs.

4.3.1.3 Cityscapes

In order to complement the multi-task classification experiments for Multi-MNIST and CelebA, we present results for Cityscapes [Cor+16], a dataset for semantic understanding of urban street scenes. We rely on a common encoder architecture from the literature [Liu+21b; LYZ21] (see Appendix A.1.1.1), a single Dropout layer is contained in the task-specific heads [LYZ21]. As for CelebA, unitary scalarisation, IMTL, and PCGrad benefit from more regularisation than the other optimisers. We employ $\lambda = 10^{-5}$ for these three algorithms, as it resulted in better validation performance on the majority of metrics, and $\lambda = 0$ for the remaining methods.

As common for Cityscapes, we report absolute and relative depth errors. Absolute depth error is the sum of the per-pixel absolute differences for depth values. To get a better understanding of performance, the relative error is used, in which the absolute value of the difference between the ground truth and prediction is divided by the ground truth values. Figure 4.3 shows test results for two metrics per task and the distribution of the training time per epoch. The test model for each run and each metric was selected as the one associated with the best (maximal or minimal, depending on the metric) validation result across epochs. We, therefore, perform per-run early stopping for most methods (see Figure A.9). As with Multi-MNIST and CelebA, no training algorithm clearly outperforms unitary scalarisation (significant overlaps across confidence intervals exist), which is again the least expensive method. In contrast with a popular hypothesis [KGC18; Che+18b; Liu+21b], this holds despite relatively large loss imbalances. In fact, the loss for the depth task is roughly 10 times smaller than that of the segmentation task: see Figures A.9f-A.9g. Unlike CelebA (see Figure 4.2b), IMTL, MGDA and GradDrop are significantly slower than unitary scalarisation (factors from 1.6 to 2.3) due to the relatively (compared to the parameter space) large size of $\mathbf{z}$ in the employed architecture. PCGrad, instead, appears to be less expensive (30% more than the baseline), demonstrating the benefits of working on $\nabla_{\boldsymbol{\theta}} \mathcal{L}_i$ on this model.

4.3.2 Reinforcement Learning

For RL experiments, we use Meta-World [Yu+19] shortly described in Appendix A.1.1.2. We employ the Soft Actor-Critic [Haa+18] implementation from [SZP21]. Unlike in §4.3.1, the employed network architecture (see Appendix A.1.1.1) is fully shared across tasks. Therefore, all SMTO implementations for these experiments rely on per-task gradients with respect to network parameters $\nabla_{\boldsymbol{\theta}}\mathcal{L}_i$ (see §4.4). Among the SMTOs we consider, PCGrad is the only one developed with the RL setting in mind. For fairness and completeness, we add all the other SMTOs from the supervised learning experiments and are the first to test these optimisers in the RL setting.

To stabilise learning, we increase the replay buffer size, a well known technique in single-task RL, add actor l_2 regularisation (also shown to be beneficial in some settings [Cob+19; Liu+21c]), and modify the reward normalisation employed by Sodhani et al. [SZP21]. The unitary scalarisation performance reported by Yu et al. [Yu+20] is considerably lower than that of Sodhani et al. [SZP21], which we believe is due to the lack of reward normalisation in the former. Sodhani et al. [SZP21] keep a moving average of rewards in the environment, with a hyperparameter controlling the speed of the moving average. Figure 4.4 shows that multi-task agent performance is highly sensitive to the reward normalisation moving average hyperparameter[2] motivating our buffer normalisation in §4.3.2. Moreover, such normalisation might make similar transitions have drastically different rewards stored in the replay buffer depending on when the transition is written to the buffer. To alleviate these issues, we store the raw rewards in the buffer and normalise only when a mini-batch is sampled.

Figure 4.5 reports the best average success rate across the updates and the runtime for 10,000 updates. In addition to these summary statistics, reported for consistency with §4.3.1, as typical for the RL literature, Figure 4.6 shows the learning curves. Our MT10 (10 tasks) results in Figure 4.5a show that by stabilising the baseline using standard RL techniques, unitary scalarisation performs on par

[2] https://github.com/facebookresearch/mtenv/blob/4a6d9d6fdfb321f1b51f890ef36b5161359e972d/mtenv/envs/metaworld/wrappers/normalised_env.py#L69

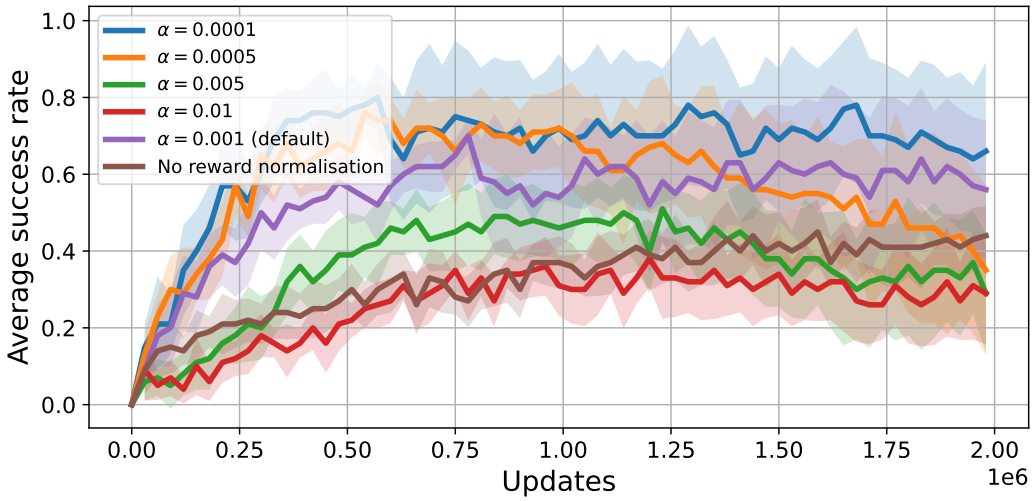
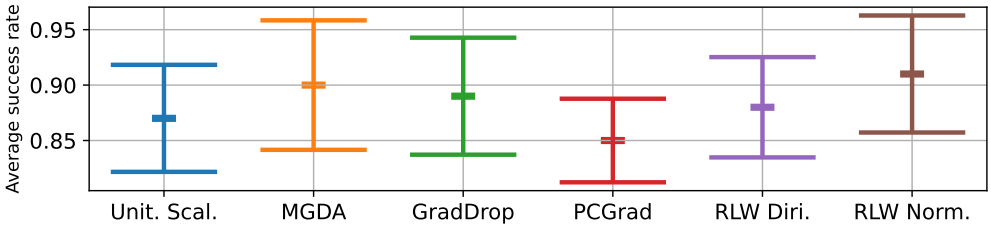
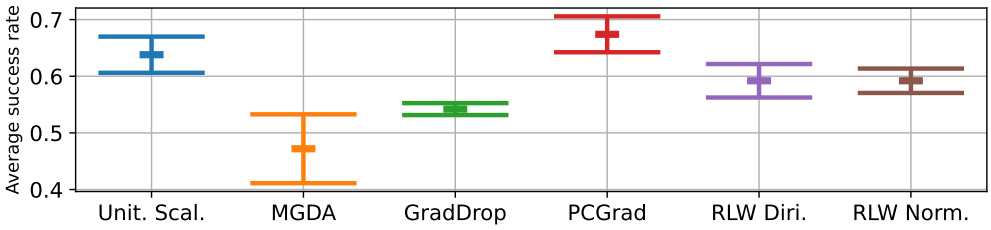


Figure 4.4: The learning outcomes of a Multi-task SAC agent vary considerably depending on the reward normalisation hyperparameter. Each curve represents an average of 10 runs with a shaded 95% confidence interval.

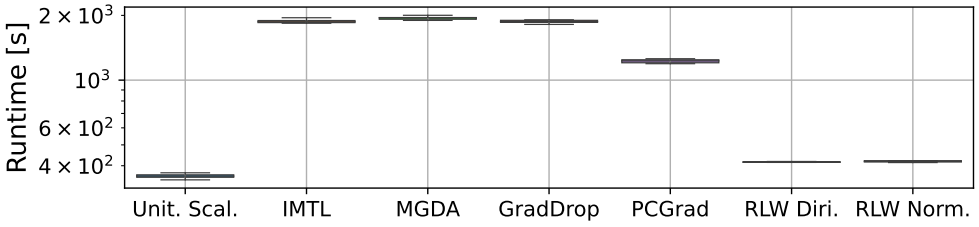
with other SMTOs, mirroring our findings in §4.3.1. This contrast with the previous literature, which reported that PCGrad outperforms unitary scalarisation [Yu+20; SZP21]. Figure 4.5b presents results on MT50 (50 tasks): similarly to MT10, none of the SMTOs significantly outperforms unitary scalarisation, with PCGrad's average being slightly above unitary scalarisation. We speculate that the stochastic loss rescaling performed by PCGrad (see Proposition 3) reduces the differences in task return scales and expect that methods like PopArt [vHas+16] would have a similar effect without requiring access to per-task gradients. While we did not tune hyperparameters for MT50 (we employed those found for MT10), it would be much easier to do that for unitary scalarisation due to its lower runtime (see Figure 4.5d). In fact, a single unitary scalarisation run takes roughly 15 hours, whereas PCGrad, MGDA and GradDrop require more than a week. As we observed in MT10, unitary scalarisation benefited from actor regularisation on MT50 (see Appendix A.1.5.2). Overall, as in the supervised learning setting, unitary scalarisation performs comparably to SMTOs despite being simpler and less demanding in both memory and compute. IMTL was unstable on this RL benchmark, and all runs crashed due to numerical overflow. We, hence, omit IMTL



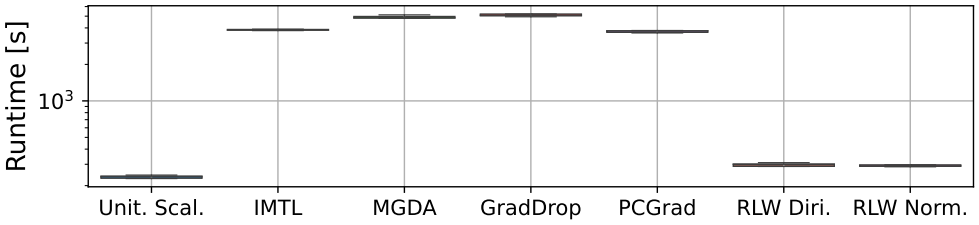
(a) MT10 performance



(b) MT50 performance



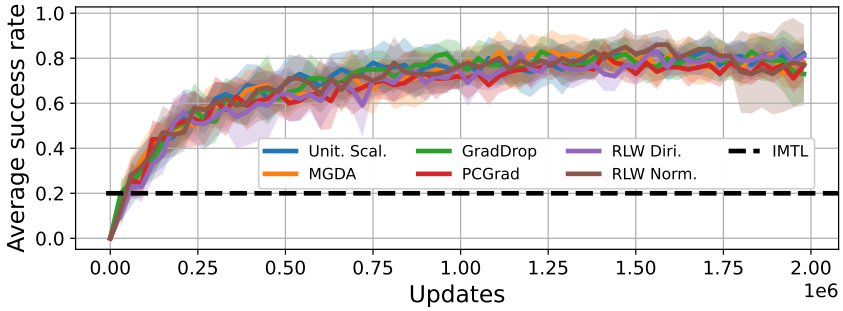
(c) MT10 runtime



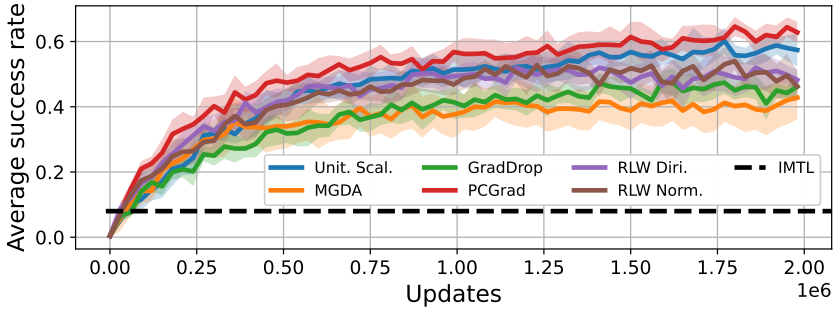
(d) MT50 runtime

Figure 4.5: On Meta-world, none of the SMTOs significantly outperforms Unit. Scal., which is the least expensive method. Subfigures a-b report mean and 95% CI (10 repetitions) for the best (over the updates) average success rate. Subfigures c-d show box plots for the training time of 10,000 updates.

results from the main text and show its results in Figure A.8 in Appendix A.1.5, which also describes a possible explanation. We hypothesise that the instability of IMTL is due to a lack of bounds on scaling coefficients. See Appendix A.1.1.2 for hyperparameter settings and ablation studies.



(a) MT10



(b) MT50

Figure 4.6: Mean and 95% CI (10 runs per method) for the avg. success rate on Meta-world. None of the SMTOs significantly outperforms unitary scalarisation. While PCgrad performs marginally better on average on MT50, we believe the gap will narrow with hyperparameter search, which will be much faster for unitary scalarisation: 15 hours versus more than a week for PCGrad.

4.4 Regularisation in Specialised Multi-Task Optimisers

The empirical results presented in §4.3 motivate the need to analyze existing SMTOs carefully. We make an initial attempt in this direction by viewing their effects through the lens of regularisation. Let us define a regulariser as a technique to reduce overfitting [Die95]. We first show that the SMTOs considered in §4.3 empirically act as regularisers via an ablation study (§4.4.1). We then take a closer look at their behaviour, presenting technical results that support their alternative interpretation as regularisers (§4.4.2). Unless otherwise stated, we assume that MTL methods apply only to $\boldsymbol{\theta}_{\|}$ and that standard gradient-based updates are employed for tasks-specific parameters $\boldsymbol{\theta}_{\perp}$. We furthermore adopt the following shorthands: $\mathcal{L}_i(\boldsymbol{\theta})$

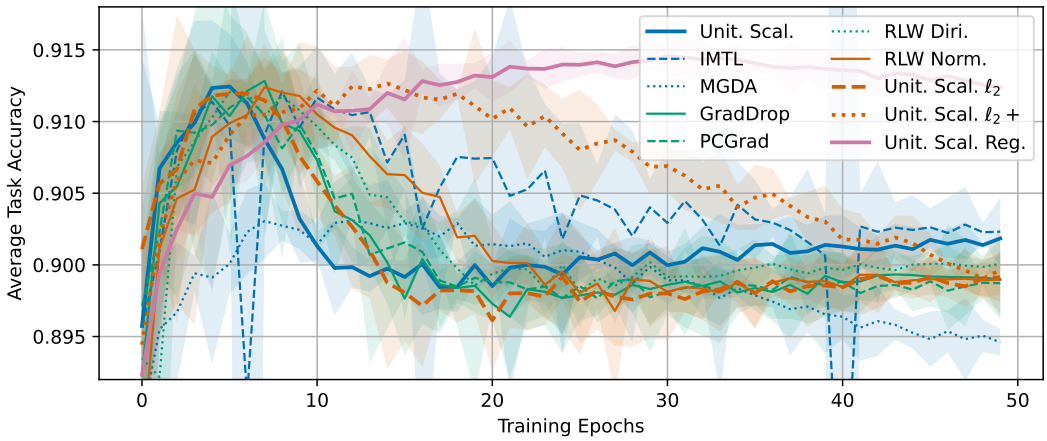


Figure 4.7: Mean and 95% CI (3 runs) avg. task validation accuracy over epochs on CelebA. SMTOs postpone the onset of overfitting, mirroring the effect of ℓ_2 regularisation on unitary scalarisation.

for $\mathcal{L}_i(f(\boldsymbol{\theta}, X, i), Y)$, and $\nabla_{\boldsymbol{\theta}}\mathcal{L}_i$ for $\nabla_{\boldsymbol{\theta}}\mathcal{L}_i(f(\boldsymbol{\theta}, X, i), Y)$.

4.4.1 Ablation Study

We repeat the experiment from §4.3.1.2 and remove explicit regularisation: no Dropout layers are added to the encoder-decoder architecture and $\lambda = 0$ for all optimisers. In addition, we examine the behaviour of two different ℓ_2-regularised instances of unitary scalarisation: $\lambda = 10^{-4}$ for "Unit. Scal. ℓ_2", $\lambda = 2 \times 10^{-3}$ for "Unit. Scal. ℓ_2+". Figure 4.7 shows that SMTOs behave similarly to an ℓ_2-penalised unitary scalarisation. Importantly, SMTOs delay overfitting, requiring less early stopping than unitary scalarisation to obtain comparable performance. In other words, early stopping is sufficient for unitary scalarisation to perform on par with SMTOs. Moreover, overfitting is further reduced by "Unit. Scal. Reg.", which plots the regularized unitary scalarisation from §4.3.1.2, with dropout layers and a weight decay of $\lambda = 10^{-3}$. Finally, Figure A.3a shows that unregularized unitary scalarisation and most SMTOs rapidly drive the training loss of each task towards its global optimum. This suggests that the main difficulty of MTL is not associated with the optimisation of its training objective, but rather to incorporating adequate regularisation. Interestingly, the accuracy curve of "Unit. Scal. Reg." does

not change abruptly, and so would probably be the least vulnerable to a bad early stopping decision. Additional results are presented in appendices A.1.3.2 and A.1.4.

4.4.2 Technical Results

All the methods considered in §4.4.1 regularise more than unitary scalarisation. While RLW was shown to reduce overfitting by the original authors [LYZ21, Theorem 2], we now provide a collection of novel and existing technical results that potentially explain the regularising behaviour of each of the other algorithms, complementing the presentation from §4.2. In particular, we show that MGDA, IMTL and PCGrad have a larger convergence set than unitary scalarisation, reducing the chances of landing on sharp local minimum [Die95]. Furthermore, GradDrop and PCGrad introduce significant stochasticity, which is often linked to the same effect [Kes+17; KLY18]. We hope these observations will steer further research.

MGDA Let us denote the convex hull of a set $\mathcal{A}$ by $\text{Conv}(\mathcal{A})$. We now recall a well-known property of MGDA [Dés12] and relate it to the behaviour of unitary scalarisation.

Proposition 1. *The MGDA SMTO [SK18] converges to a superset of the convergence points of unitary scalarisation. More specifically, it converges to any point* $\boldsymbol{\theta}^*_{\|}$ *such that:* $\mathbf{0} \in \textit{Conv}(\{\nabla_{\boldsymbol{\theta}^*_{\|}} \mathcal{L}_i \mid i \in \mathcal{T}\})$.

See Appendix B.1 for a simple proof. As a consequence of Proposition 1, MGDA does not necessarily reach a stationary point for $\mathcal{L}^{\text{MT}}$ (that is, a point for which $\sum_{i \in \mathcal{T}} \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i = \mathbf{0}$) or $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i = \mathbf{0}$ for any of the losses $\mathcal{L}_i$. For example, any point $\boldsymbol{\theta}_{\|}$ for which two per-task gradients point in opposite directions is Pareto stationary. On account of the well-known [Die95] relationship between under-optimising (e.g., early stopping [CLG00; LSO20]) and overfitting, Proposition 1 supports the interpretation of MGDA as a regulariser for Equation (2.1). Empirical evidence that MGDA under-optimises is provided in Figure A.3a, Appendix A.1.4, and Figure 4.7, which shows over-regularisation. Proposition 1 can be extended to the recent Nash-MTL, which shares the same convergence set [Nav+22, Theorem 5.4]

IMTL We now show that aggregating per-task gradients so that their cosine similarity is the same (Equation (4.4)) yields a constrained steepest-descent algorithm (Proposition 2). This view on the update step of IMTL leads to a novel analysis of its convergence points (Corollary 1). Proofs can be found in Appendix B.2. We will denote by $\mathrm{Aff}(\mathcal{A})$ the affine hull of a set $\mathcal{A}$.

Proposition 2. *IMTL by Liu et al. [Liu+21b] updates $\boldsymbol{\theta}_{\|}$ by taking a step in the* steepest descent direction whose *cosine similarity with per-task gradients is the same across tasks.*

Corollary 1. *IMTL by Liu et al. [Liu+21b] converges to a superset of the Pareto-optimal points for $\boldsymbol{\theta}_{\|}$ (and hence of the convergence points of the unitary scalarisation). More specifically, it converges to any point $\boldsymbol{\theta}_{\|}^*$ such that:*

$$\mathbf{0} \in \mathit{Aff}\left(\left\{ \frac{\nabla_{\boldsymbol{\theta}_{\|}^*} \mathcal{L}_i}{\left\| \nabla_{\boldsymbol{\theta}_{\|}^*} \mathcal{L}_i \right\|} \mid i \in \mathcal{T} \right\}\right).$$

As seen for MGDA, Corollary 1 implies that, even if the employed model f has the capacity to reach the minimal loss on $\mathcal{L}^{\mathrm{MT}}$, IMTL may stop before reaching a stationary point. Recalling the relationship between under-optimising and overfitting [Die95], this supports the interpretation of IMTL as a regulariser for Equation (2.1). This is empirically shown in Figures 4.7, A.3a, and A.6. In particular, unitary scalarisation reaches the same average performance of IMTL but requires earlier stopping.

PCGrad We provide an alternative characterisation of the PCGrad update rule, highlighting its stochasticity in the context of its interpretation as loss rescaling [Liu+21b; LYZ21]. See Appendix B.3 for a proof.

Proposition 3. *PCGrad is equivalent to a dynamic, possibly stochastic, loss rescaling for $\boldsymbol{\theta}_{\|}$. At each iteration, per-task gradients are rescaled as follows:*

$$\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i \leftarrow \left(1 + \textstyle\sum_{j \in \mathcal{T} \setminus \{i\}} d_{ji}\right) \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i, \;\; d_{ji} \in \left[0, \frac{\left\| \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_j \right\|}{\left\| \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i \right\|}\right].$$

Furthermore, if $|\mathcal{T}| > 2$, d_{ji} is a random variable, and the above range contains its support.

The results from Proposition 3 can be extended to GradVac [Wan+21], which generalises PCGrad's projection onto the normal vector to arbitrary target cosine similarities between per-task gradients. When $|\mathcal{T}| > 2$, PCGrad corresponds to a stochastic loss re-weighting. As such, PCGrad bears many similarities with RLW [LYZ21]. RLW proposes to sample scalarisation weights from standard probability distributions at each iteration and proves that this leads the better generalisation [LYZ21, Theorem 2]. Indeed, it is well-known that adding noise to stochastic gradient estimations leads the optimisation towards flatter minima and that such minima may reduce overfitting [Kes+17; KLY18]. In line with the main technical results by Yu et al. [Yu+20], we now restrict our focus to two-task problems, which allows for an easy description of PCGrad's convergence points. The result is primarily based on [Yu+20, Theorem 1]: we relax some of the assumptions and provide a proof in Appendix B.3.

Corollary 2. *If $|\mathcal{T}| = 2$, PCGrad will stop at any point where $\cos(\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_1, \nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_2) = -1$. Furthermore, if $\mathcal{L}_1$ and $\mathcal{L}_2$ are differentiable, and $\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}^{MT}$ is L-Lipschitz with $L > 0$, PCGrad with step size $t < \frac{1}{L}$ converges to a superset of the convergence points of the unitary scalarisation.*

Corollary 2 implies that, when $|\mathcal{T}| = 2$, PCGrad may under-optimise equation (2.1) as MGDA and IMTL. In particular, if $\cos(\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_1, \nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_2) = -1$, then $\mathbf{0} \in \text{Conv}(\{\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_1, \nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_2\})$ (see proposition 1) We believe that PCGrad's stochasticity and enlarged convergence set potentially explain its regularising effect.

GradDrop While the motivation behind GradDrop is to avoid entry-wise gradient conflicts across tasks, the main property of the method is to drive the optimisation towards "joint minima": points that are stationary for all the individual tasks at once [Che+20, Proposition 1]. In other words: $\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_i = \mathbf{0}\ \forall\ i \in \mathcal{T}$. While this property is desirable, we show that it holds beyond GradDrop, and independently of the gradient directions. Under strong assumptions on the model capacity, the above property would trivially hold for unitary scalarisation (Proposition 5,

Appendix B.4). Proposition 4 shows that it holds for a simple randomised version of unitary scalarisation, which we name Random Grad Drop (RGD).

Proposition 4. *Let* $\mathcal{L}^{RGD}(\boldsymbol{\theta}_{\|}) := \sum_{i\in\mathcal{T}} u_i \mathcal{L}_i(\boldsymbol{\theta}_{\|})$*, where* $u_i \sim Bernoulli(p)\ \forall i \in \mathcal{T}$ *and* $p \in (0, 1]$*. The gradient* $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}^{RGD}$ *is always zero if and only if* $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i = \mathbf{0}\ \forall i \in \mathcal{T}$*. In other words, the result from [Che+20, Proposition 1] can be obtained without any information on the sign of per-task gradients.*

Proposition 4 (see Appendix B.4 for a simple proof) shows that an inexpensive sign-independent stochastic scalarisation shares GradDrop's main reported property. $\mathcal{L}^{\text{RGD}}$ can be directly cast an instance of RLW, and hence as a regularisation method [Kes+17; KLY18]. Furthermore, Figure A.7 in Appendix A.1.4.1 shows that the empirical results of GradDrop on CelebA [Liu+15] are closely matched by a sign-agnostic gradient masking, partly undermining the conflicting gradients assumption. We believe that the above results, along with the authors' original experiments showing that GradDrop delays overfitting on CelebA [Che+20, Figure 3], suggest that GradDrop behaves as a regulariser.

4.5 Conclusions

This chapter made two main contributions. First, we evaluated popular SMTOs using a single experimental pipeline, including previously unpublished results of MGDA, IMTL, RLW, and GradDrop in the RL setting. Surprisingly, our evaluation showed that none of the SMTOs consistently outperform unitary scalarisation, the simplest and least expensive method. Second, to explain our surprising results, we postulated that SMTOs act as regularisers and presented an analysis that supports our hypothesis. We believe our work calls for a further reevaluation of progress in developing principled and efficient MTL algorithms.

We conclude by addressing the limitations of our work. While we covered a wide range of popular benchmarks, we do not exclude the existence of settings where unitary scalarisation underperforms: discovering them is an exciting direction

for future work. Furthermore, our experimental results were obtained via grid searches under limited compute resources: some methods might benefit from further fine-tuning. Nevertheless, we remark that fine-tuning will be easier for unitary scalarisation due to its shorter runtime. Finally, we presented the regularisation hypothesis only as a partial explanation of our results: we hope it will steer further analysis and consequently improve the understanding of MTL. We are curious about qualitative differences in regularisation induced by different SMTOs, as well as the potential interplay between the task nature (e.g., difficulty) and the effectiveness of the regularisation.

In this chapter, we implicitly assume that each parameter update employs information from all tasks. However, not all works satisfy this assumption, either due to a large number of tasks [Cap+21; Kur+20] or simply as an implementation decision [HMP20; Kur+21]. In this setting, MTL resembles other problems dealing with multiple tasks, i.e., continual [Khe+20a], curriculum [Nar+20a], and meta-learning [Hos+22], which are not the focus of this work.

It is not daily increase but daily decrease,
hack away the unessential.

— Bruce Lee

5

Scaling GNNs to High-Dimensional Continuous Control

Contents

5.1 Introduction

5.1.1 Motivation

GNNs have been successfully applied to MTRL, with promising results on incompatible locomotion control tasks. Not only are GNN policies as effective as MLPs on certain training tasks, but when a trained policy is transferred to another similar task, GNNs significantly outperform MLPs [Wan+18; HMP20] in terms of sample efficiency.

This is most likely due to the capacity of a single GNN to operate over arbitrary graph topologies and sizes without retraining. In contrast, MLPs require ad hoc masking and padding to accommodate changes in the input or output dimensions or require training of new input and output layers to deal with incompatibility.

However, so far GNNs in RL have only shown competitive performance with MLPs on lower-dimensional locomotion control tasks. For higher-dimensional tasks, one must therefore choose between superior training task performance (MLPs) and superior transfer performance (GNNs). This chapter studies the reasons behind the subpar scaling of GNNs in the incompatible control setting and tries to alleviate these problems. In this chapter, we use *scaling* in a slightly non-conventional way. Typically, in GNN research [Jos22] (mostly in supervised settings), scaling implies having giant graphs with hundreds of thousands (or even millions) of nodes and edges that are not even possible to fit in the memory of accelerators.

5.1.2 Contribution

Chapter §5.2 formalises the *incompatible control* problem and introduces a GNN-based architecture [Wan+18, NerveNet], chosen for its strong zero-shot transfer performance. In §5.3, we show that optimisation updates for the GNN policy tend to cause excessive changes in policy space, leading to performance degrading. To

combat this, current state-of-the-art algorithms [Sch+15; Sch+17; Abd+18c] employ trust region-like constraints, inspired by natural gradients [Ama96; Kak01b], that limit the change in policy for each update. We outline how this policy instability can be framed as a form of overfitting—a problem GNNs are known to suffer from in supervised learning and show that parameter regularisation (a standard remedy for overfitting) leads to a slight improvement in GNN performance.

We then investigate which structures in the GNN contribute most to this overfitting by applying different learning rates to different parts of the network. Surprisingly, the best performance is attained when training with a learning rate of zero in the parts of the GNN architecture that encode, decode, and propagate messages in the graph; in effect, training only the part that updates node representations.

We use this approach as the basis of our method, Snowflake (SF), which freezes the parameters of particular operations within the GNN to their initialised values, keeping them fixed throughout training while updating the non-frozen parameters as before. This simple technique enables GNN policies to be trained much more effectively in high-dimensional environments. Experimentally, we show that applying SF to GNN dramatically improves asymptotic performance and sample complexity on such tasks. Finally, we demonstrate that a policy trained with SF exhibits improved zero-shot transfer compared to regular GNN or MLPs on high-dimensional tasks.

This chapter is based on Blake et al. [Bla+21] in which the first author contributed the majority of the ideas, figures, text, implementation and experiments. The author of this thesis played a significant advisory role, defined the original problem, contributed to the codebase upon which the first author built and took part in the writing of the individual chapters.

5.2 Incompatible Continuous Control

We now formalise the *incompatible control* problem and describe Nervenet [Wan+18], a prominent method for solving this problem.

Incompatible control is a continuous MTRL problem that implies learning a common locomotion policy for a set of agents with a varying number of limbs and connectivity of those limbs, i.e. *morphology*. To be more precise, a set of incompatible continuous control environments is a set of MDPs described in §2.2. When a state is represented as a graph, each node label contains features of its corresponding limb, e.g., limb type, coordinates, and angular velocity. Similarly, each dimension of an action set element corresponds to a node with the label meaning the torque for a joint to emit. MuJoCo `gym` [TET12; Bro+16] is a popular set of environments used for benchmarking continuous control agents. The reward function of these agents includes a reward for staying alive (not falling), distance covered, and a penalty for action magnitudes.

GNNs can process graphs of arbitrary sizes and topologies, hence they are well-suited for tackling the incompatible control problem. To use them, we assume that the environment provides the graph representation of the state, i.e., we are aware of the connectivity of the graph. Moreover, we assume access to mapping from each element of an observational vector to annotations of graph nodes, e.g. the first coordinate of the observation is the first coordinate of the feature vector of node v_0. As common in RL, we do not assume any knowledge of the semantics behind the features. NerveNet [Wan+18] is a pioneering incompatible control method using GNNs to approximate the policy and the critic. We will now describe it in detail.

5.2.1 NerveNet

NerveNet [Wan+18] is the first work that successfully performs transfer and multi-task RL in incompatible environments. NerveNet uses PPO (§2.2.1.2) to find an optimal policy $\pi(a|s)$ with a Gated GNN [Li+16] as a policy and an MLP as a

critic, training one critic per task. The authors show transfer results on `Centipede` environments, varying the number of legs of a centipede-like creature. For multi-task experiments, the authors train a single model on `Walkers`, a diverse set of five locomotion environments resembling five different animals: an ostrich, a wolf, a hopper, a humanoid, and a horse.

Instead of providing the flat observation vector to the policy, the observation is split into a graph whose topology is defined by the physical morphology of a robot. Node labels represent each actuator features: coordinates and angular velocities. In NerveNet, the input observations are first encoded via a MLP processing each node labels as a batch element: $\boldsymbol{v}^i \leftarrow \varphi_{\boldsymbol{\chi}}\Big(\boldsymbol{v}^i\Big),\ \forall v^i \in \mathcal{V}$. Using the GNN formalism from §2.3, NerveNet's forward step is a simplified version of Equation 2.16, without the global component and with the edge updater using only the source node label $\boldsymbol{v}^i$ when computing the message:

$$\begin{aligned} \boldsymbol{e}'^{ij} &\leftarrow \quad \varphi^e_{\boldsymbol{\psi}}\Big(\boldsymbol{v}^i\Big) \quad &, \forall e^{ij} \in \mathcal{E}\,, \\ \boldsymbol{v}'^{i} &\leftarrow \quad \varphi^v_{\boldsymbol{\xi}}\Big(\boldsymbol{v}^i,\, \rho\{\boldsymbol{e}'^{ki} \,|\, e^{ki} \in \mathcal{E}\}\Big) \quad &, \forall v^i \in \mathcal{V}\,. \end{aligned}$$

The edge updater MLP $\varphi^e_{\boldsymbol{\psi}}$ in NerveNet does not take the receiver's state into account. Using only one message pass restricts the learned function to local computations on the graph. The node updater $\varphi^v_{\boldsymbol{\xi}}$ is a Gated Recurrent Unit [Cho+14] which maintains the internal state when doing multiple message-passing iterations and takes the aggregated outputs of the edge updater for all incoming edges as inputs. After the message-passing stage (four steps in the original implementation), the MLP decoder takes the states of the nodes and, like the encoder, independently processes them, emitting scalars used as the mean for the normal distribution from which actions are sampled: $\boldsymbol{v}^i_{dec} \leftarrow \varphi_{\boldsymbol{\eta}}\Big(\boldsymbol{v}^i\Big), \forall v^i \in \mathcal{V}$. We followed the original NerveNet implementation in this design decision, however, it is not unusual to use the mean of the Gaussian as an action during evaluation. On the one hand, sampling during evaluation can prevent the agent from being stuck. On the other, it might deteriorate the performance when an extreme action is sampled. As in

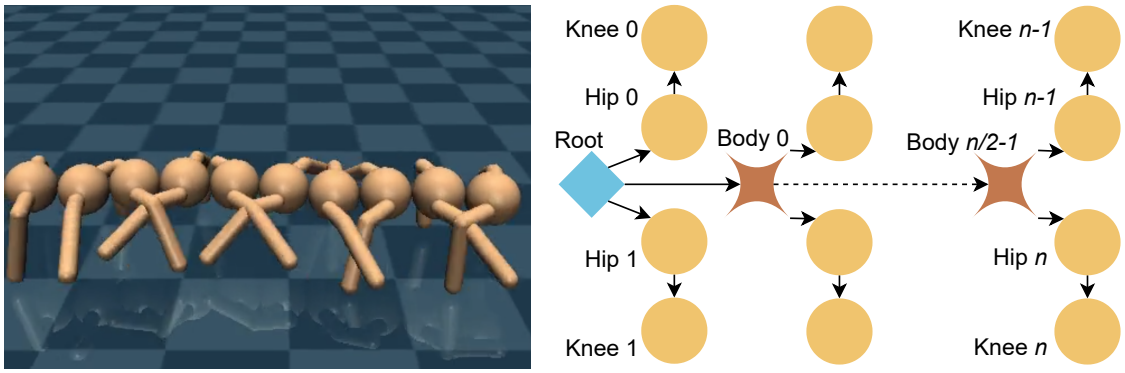


Figure 5.1: A MuJoCo rendering of `Centipede-20` and its corresponding morphological graph.

a typical PPO implementation[1], the standard deviation of this distribution is a separate state-independent vector with one scalar per action.

5.3 Analysing GNN Scaling Challenges

In this section, we analyse the challenges that limit GNNs' ability to scale. We primarily use the same experimental setup as Wang et al. [Wan+18], with Appendix A.2 outlining the details of any differences and our choice of hyperparameters.

We focus on environments derived from the Gym [Bro+16] suite, using the MuJoCo [TET12] physics engine. The main set of tasks we use to assess scaling is the selection of `Centipede-n` agents [Wan+18], chosen because of their relatively complex structure and ability to be scaled up to high-dimensional input-action spaces.

The morphology of a `Centipede-n` agent consists of a line of `n/2` body segments, each with a left and right leg attached (see Figure 5.1).

The graph used as the basis for the GNN corresponds to the physical structure of the agent's body. At each timestep in the environment, the MuJoCo engine sends a feature vector containing positional information regarding body parts and joints, expecting a vector to be returned specifying forces to apply at each joint. The agent is rewarded for forward movement along the y-axis as well as a small 'survival' bonus

[1] https://github.com/ikostrikov/pytorch-a2c-ppo-acktr-gail/blob/41332b78dfb50321c29bade65f9d244387f68a60/a2c_ppo_acktr/distributions.py#L84

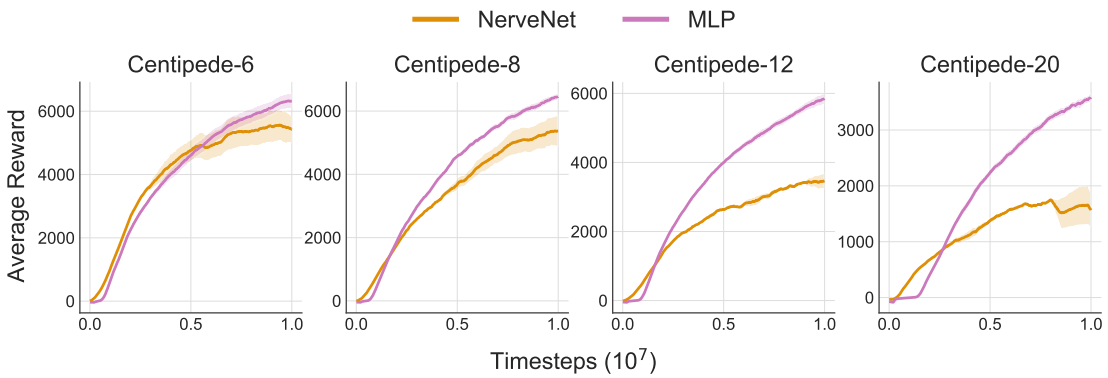


Figure 5.2: Comparison of the scaling of GNN relative to an MLP-based policy. Performance is similar for the smaller agent sizes, but GNN scales poorly to the larger agents.

for keeping its body within certain bounds and given negative rewards proportional to the size of its actions and the magnitude of force it exerts on the ground.

Existing work applying GNNs to locomotion control tasks avoids training directly on larger agents, i.e., those with many nodes in the underlying graph representation. For example, Wang et al. [Wan+18] state that for GNN, "training a `CentipedeEight` from scratch is already very difficult". Huang et al. [HMP20] also limit training their SMP architecture to agents of small size.

5.3.1 Scaling Performance

To demonstrate the scaling properties of GNNs to larger agents, we compare their performance on a selection of `Centipede-n` tasks to MLP policies. Figure 5.2 shows that for the smaller `Centipede-n` agents both policies are similarly effective, but as the size of the agent increases, the performance of GNN drops relative to the MLP. A visual inspection of the behaviour of these agents shows that for `Centipede-20`, GNN barely makes forward progress at all, whereas the MLP moves effectively.

As in previous literature [e.g., Wan+18; HMP20], we are ultimately not concerned with outperforming MLPs on the specific training task but instead matching their training task performance so that the *additional* benefits of GNNs can be realised,

Table 5.1: KL-divergence from the policy before each update to the policy after, calculated over each batch. We train on 10^7 timesteps, recording the mean taken over the last 10% of steps.

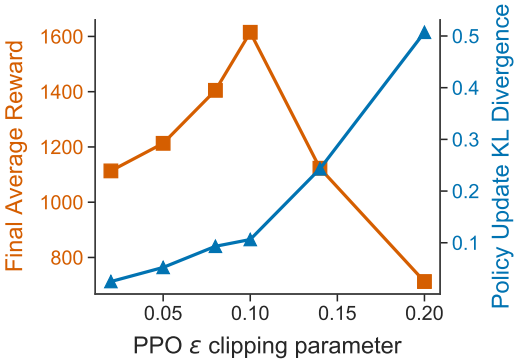
	Policy KL-divergence			
Policy type	`Centipede-6`	`Centipede-8`	`Centipede-12`	`Centipede-20`
MLP	0.021	0.024	0.031	0.044
GNN	0.115	0.137	0.118	0.123

particularly their strong transfer properties. In other words, we focus on closing the gap in Figure 5.2, as doing so makes GNNs a better choice overall given the trained policy transfers better than the MLP equivalent (see §5.5 for experimental results).

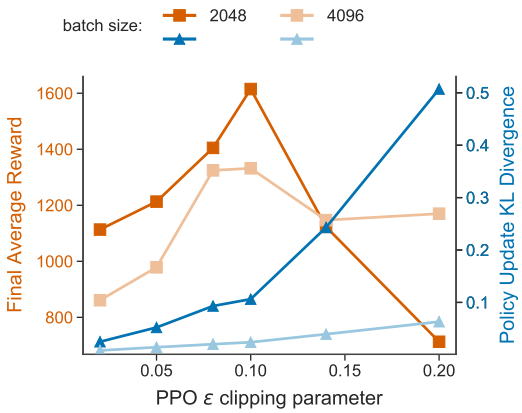
5.3.2 Unstable policy updates

As outlined in §2.2.1.2, one of the key challenges for on-policy RL is preventing individual updates from causing excessive changes in policy space (i.e., keeping it within the trust region). Table 5.1 shows how this problem contributes to GNN's poor scaling, calculating the average KL-divergence from the pre-update policy to the post-update policy for both policy types. GNN has a consistently higher KL-divergence than the MLP policy, indicating that PPO finds it harder to ensure stable policy updates for the GNN. We emphasise that this discrepancy persists even with carefully-tuned hyperparameter values for limiting policy divergence. Figure 5.3a shows the performance of GNN across a range of PPO ϵ-clipping values (see §2.2.1.2), and in all cases GNN is still substantially inferior to an MLP[2]. As we demonstrate later (in Figure 5.11), controlling policy divergence effectively is a key component in making GNNs scale However, we see here that the clipping coefficient alone does not control the divergence sufficiently to achieve this: the best performance is capped by 1600 compared to more than 3000 of the MLP policy.

[2]Note that our experiments on GNN in the rest of the chapter always use the best value of $\epsilon = 0.1$ found here.



(a) As ϵ increases (i.e., clipping is reduced), the KL divergence from the old to new policy (blue) increases. This improves performance (orange) up to a point, when it begins to deteriorate.



(b) Larger batch size reduces the policy divergence making the algorithm less sensitive to high values of ϵ (i.e. low clipping), but reduces the maximum reward attained within the training time-frame.

Figure 5.3: The effect of the the clipping parameter ϵ on the final performance of GNN on `Centipede-20` after ten million timesteps

5.3.3 Overfitting in GNN

Excessive policy divergence resulting from updates can be understood as a form of overfitting. Whereas the supervised interpretation of overfitting implies poor generalisation from training to test set, in this case, we are concerned with poor generalisation across state-action distributions induced by different iterations of the policy during training. Specifically, each update involves an optimisation step aiming to increase the expected reward over a batch of trajectories generated using the *pre-update* policy. The challenge for RL algorithms is that the agent is then evaluated and trained on trajectories generated using the *post-update* policy, i.e., a different distribution to the one optimised.

GNNs are known to overfit in the supervised setting, when edge updaters are implemented as MLPs [Ham20, p.55]. Here, we demonstrate that they also overfit (using the above interpretation) in our on-policy RL setting. Figure 5.4 shows the effect of applying l_2 regularisation (a standard approach to reducing overfitting) to the GNN architecture. We regularise the parameters $\boldsymbol{\psi}$ of GNN's edge updater

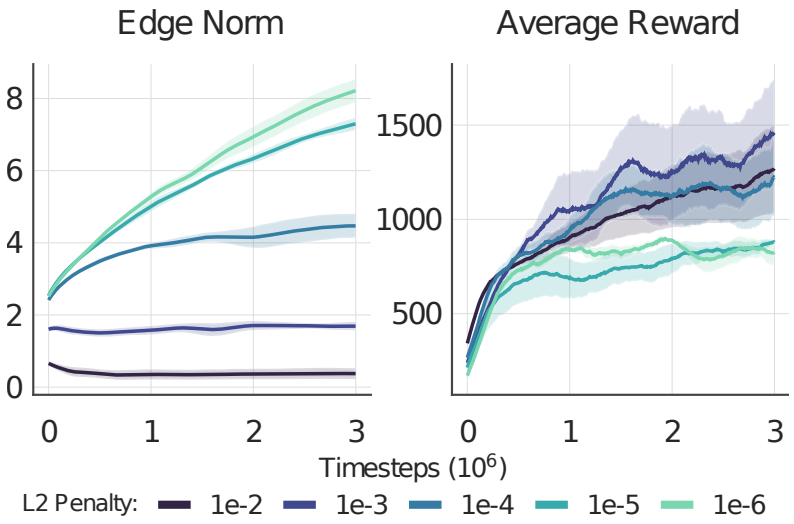


Figure 5.4: l_2 regularisation for GNN's edge updater across a range of values for the L2 penalty λ, trained on `Centipede-20`. Increasing this penalty reduces the L2 norm of the weights learned (left). Improved performance for higher values of λ (right) indicates the presence of overfitting for the edge updater.

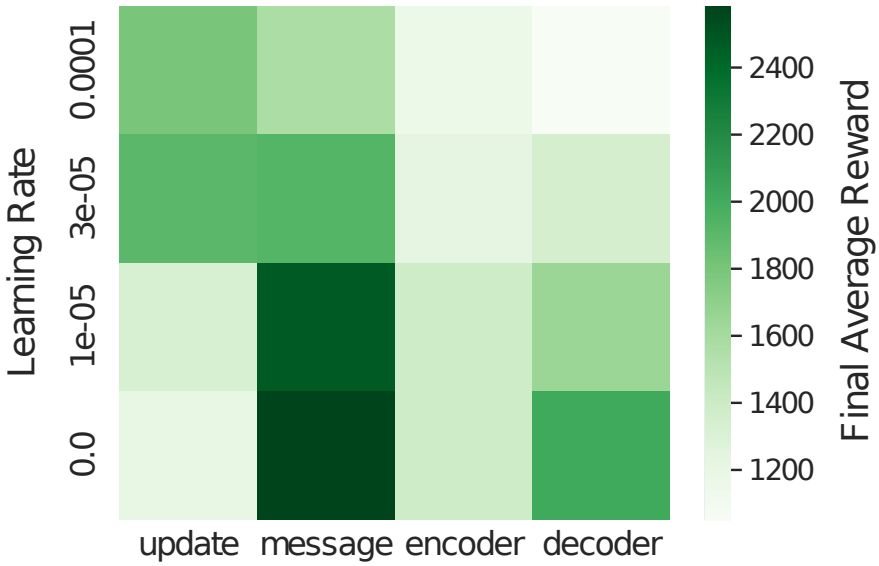


Figure 5.5: Colour-coded final GNN performance after 5M training steps on `Centipede-20` when changing learning rates for *individual* GNN components, compared to the base learning rate of 3×10^{-4}.

MLP φ_ψ, adding a $\lambda||\boldsymbol{\psi}||_2^2$ term to our objective function. At the optimal value of λ, we see a performance improvement (although still substantially inferior to using an MLP), indicating that the unregularised message-passing MLPs overfit.

We also investigate lowering the learning rate in different parts of the GNN to

identify where overfitting is localised. If parts of the network are particularly prone to damaging overfitting, training them more slowly may reduce their contribution to policy instability across updates. Results of this experiment can be seen in Figure 5.5.

Not only does lowering the learning rate in parts of the model improve performance, but surprisingly the best performance is obtained when the encoder $\varphi_{\boldsymbol{\chi}}$, edge updater $\varphi^e_{\boldsymbol{\psi}}$ and decoder $\varphi_{\boldsymbol{\eta}}$ each have their learning rate set to zero. The encoder and decoder play a similar role to the edge updater, all of which are implemented as MLPs. In contrast, the node update function $\varphi^v_{\boldsymbol{\xi}}$ is a Gated Recurrent Unit (we experimented with using an MLP update function, but found that this significantly reduced performance most likely due to partial observability.).

5.4 Snowflake

Training with a learning rate of zero is equivalent to parameter freezing (e.g., Brock et al. [Bro+17a]), where parameters are fixed to their initialised values throughout training. GNN can learn a policy with some of its functions frozen, as learning still takes place in the un-frozen functions. For instance, if we consider freezing the encoder, this results in an arbitrary mapping of input features to the initial hidden states. As we still train the update function that processes this representation, so long as crucial information from the input features is not lost via arbitrary encoding, the update function can still learn valuable representations. The same logic applies to using a frozen decoder or edge updater.

Based on the effectiveness of parameter freezing within parts of the network, we propose a simple technique for improving the training of GNNs via gradient-based optimisation, which we name Snowflake (SF) (a naturally-occurring frozen graph structure). SF assumes a GNN architecture made up internally of functions $F^1_{\boldsymbol{\theta}}, \ldots, F^n_{\boldsymbol{\theta}}$, where $\boldsymbol{\theta}$ denotes the parameters of a given function. Prior to training we select a fixed subset $\mathcal{Z} \subseteq \{F^1_{\boldsymbol{\theta}}, \ldots, F^n_{\boldsymbol{\theta}}\}$ of these functions. Their parameters are then placed in SF's *frozen set* $\zeta = \{\boldsymbol{\theta} \mid F_{\boldsymbol{\theta}} \in Z\}$. During training, SF excludes parameters in ζ from being updated by the optimiser, instead fixing them to

whatever values the GNN architecture uses as an initialisation. Gradients still flow through these operations during backpropagation, but their parameters are not updated. In practice, we found optimal performance for $\{\varphi_{\boldsymbol{\chi}}, \varphi_{\boldsymbol{\eta}}, \varphi^{e}_{\psi}\}$, i.e. when freezing the encoder, decoder and edge updater of the GNN. If not stated otherwise, this is the architecture we refer to as SF in subsequent sections. Figure 5.6 shows the training curves for other configurations of frozen parameters.

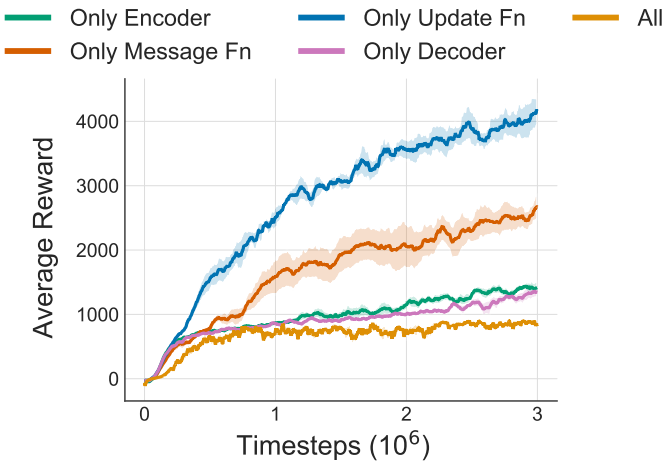


Figure 5.6: Ablation demonstrating the effect of only training single parts of the network (freezing the rest). The configuration of SF we use for our experiments is equivalent to only training the update function, which is the most effective approach here. All approaches are superior to training the entire GNN. For this experiment, we train on `Centipede-6` using the small batch size of 256 in all cases.

For our experiments, we initialise the values in the GNN using the orthogonal initialisation [SMG14]. We found this slightly more effective for frozen and unfrozen training than uniform and Xavier initialisations [GB10]. For our edge updater, which has input and output dimensions of the same size, we find that performance with the frozen orthogonal initialisation is similar to that of simply using the identity function instead of an MLP. However, in the general case where the input and output dimensions of functions in the network differ (such as in the encoder and decoder, or in GNN architectures where layers use representations of different dimensionality), this simplification is not possible.

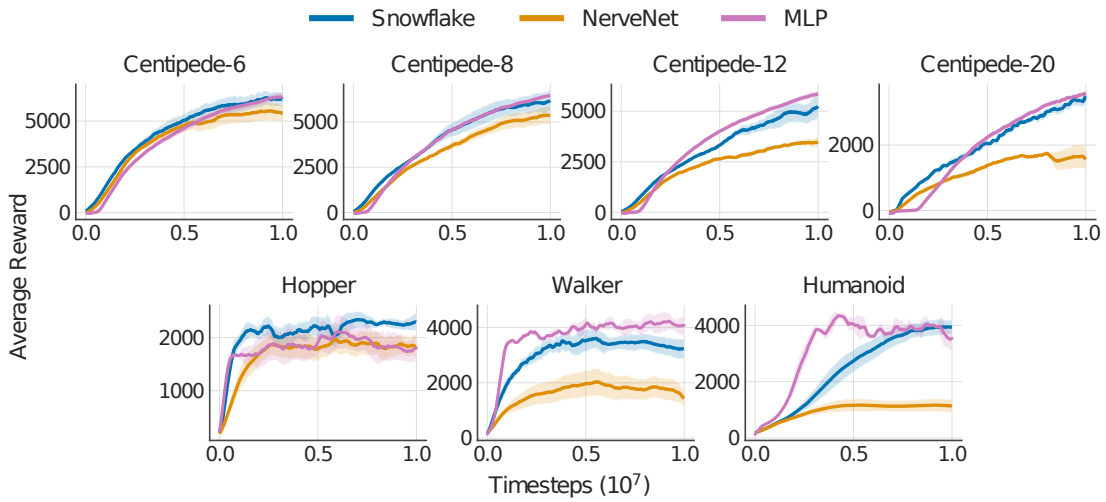


Figure 5.7: Comparison of the performance of SF training, regular GNN and the MLP-based policy. SF enables effective scaling to the larger agents, significantly outperforming regular GNN and comparable to using an MLP-based policy.

5.5 Empirical Results

We present experiments evaluating the performance of SF when applied to GNN, and compare against regular GNN and MLP policies. We evaluate each model on a selection of MuJoCo tasks, including three standard tasks from the Gym suite [Bro+16] and the `Centipede-n` agents from Wang et al. [Wan+18]. Note that we do not train on even larger `Centipede-n` agents due to wall-clock simulation time becoming prohibitively large.

All training statistics are calculated as the mean across six independent runs (unless specified otherwise), with the standard error across runs indicated by the shaded areas on each graph. The average returns typically have high variance, so we plot the mean taken over a sliding window of 30 data points to smoothen our results. Appendix A.2 provides further experimental details.

5.5.1 Scaling to High-Dimensional Tasks

Figure 5.7 compares the scaling properties of the regular GNN model with SF. As the size of the agent increases, SF significantly outperforms GNN with comparable

asymptotic performance to the MLP. This indicates that SF successfully addresses the deficiencies of regular GNN training and that freezing overfitting parameters is an effective training strategy in this setting. This holds across locomotive agents with substantially different morphologies.

5.5.2 Zero-shot transfer

An essential motivation for improving GNN scaling is to harness their transfer capabilities on large tasks. Regular GNNs are limited because they can only effectively train on and transfer between small agent sizes.[3] We show in Figure 5.8 that SF attains exceptional zero-shot transfer performance across centipede sizes, surpassing alternative methods. SF is the only method that can train a single policy that is effective on `Centipede-20` through to `12`.

SF therefore achieves our initial objective: combining the strong training task performance of an MLP and the strong transfer performance of regular GNN. Consequently, SF-trained GNNs offer the most promising policy representation for locomotion control tasks where the transfer is a desirable property.

5.5.3 Policy Stability and Sample Efficiency

By reducing overfitting in parts of the GNN, SF mitigates the effect of harmful policy updates seen with regular GNN. As a consequence, the policy can train effectively on smaller batch sizes. This is demonstrated in Figure 5.9, which shows the performance of GNN trained regularly versus using SF as the batch size decreases.

A potential benefit of training with smaller batch sizes is improved sample efficiency, as fewer timesteps are taken in the environment per update. However, smaller batch sizes also lead to increased policy divergence due to increased noise in the gradient estimate. When the policy divergence is too significant, performance decreases, limiting how small the batch can be. However, due to a reduction in policy

[3] We found that a regular GNN policy trained to achieve high training task performance on a smaller agent (e.g., `Centipede-6`) does not transfer effectively to larger ones, as reflected in [Wan+18, Figure 4].

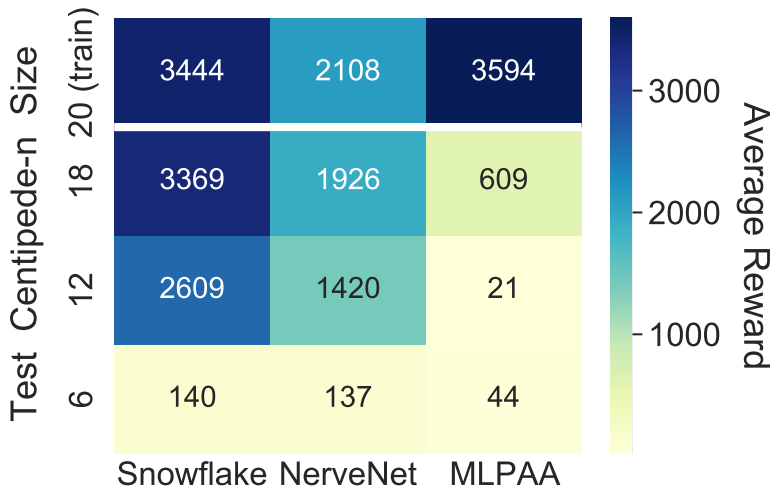
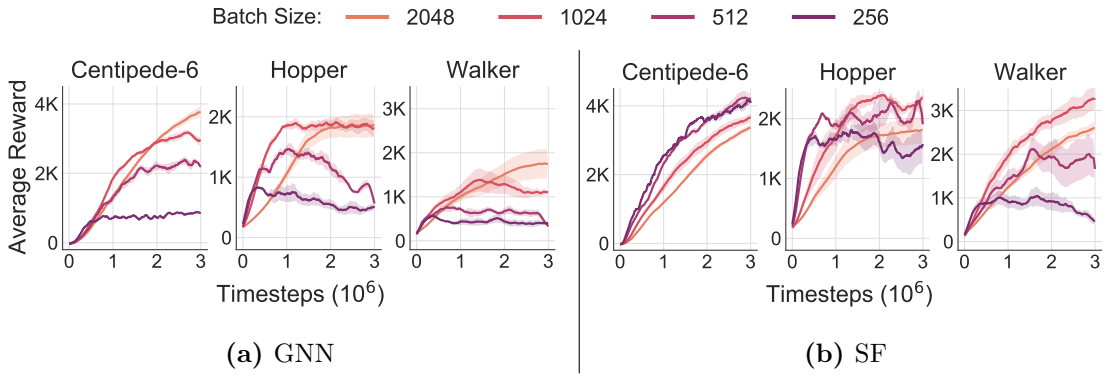


Figure 5.8: Zero-shot transfer performance for SF, GNN, and MLP models trained on `Centipede-20`, evaluated across a range of sizes.



(a) GNN **(b)** SF

Figure 5.9: Effectiveness of SF across smaller batch sizes relative to standard GNN training. SF can smaller batch sizes, improving sample efficiency. This is due to SF reducing policy divergence across updates.

divergence as a result of SF, we can afford to use smaller batch sizes while still keeping the policy under control. This provides a broader motivation for using SF than just scaling to larger agents: it also improves sample efficiency across agents regardless of size. Figure 5.10 demonstrates the policy divergence dependence on the batch size.

The success of SF in scaling to larger agents can also be understood in this context. Without SF, for GNN to attain strong performance on large agents, an infeasibly large batch size would be required, leading to poor sample efficiency. The more stable policy updates enabled by SF make solving these large tasks tractable.

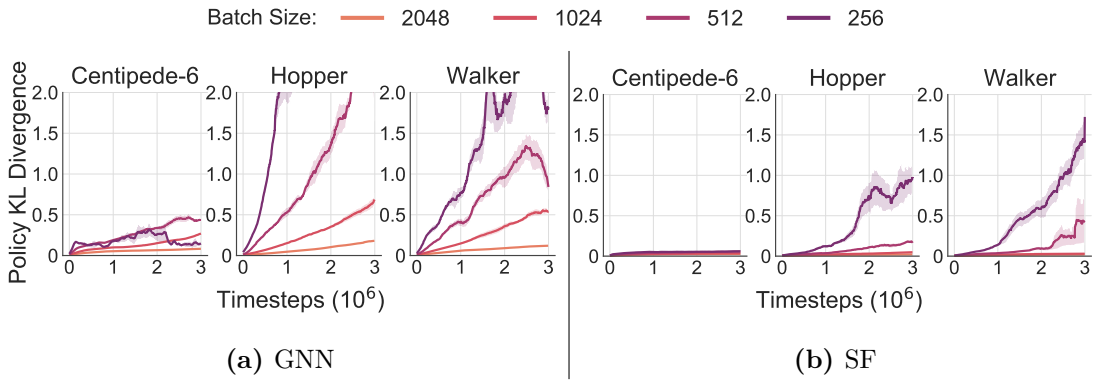


Figure 5.10: Accompanying KL divergence plots for Figure 5.9. As SF reduces the policy divergence between updates, smaller batch sizes can be used before the KL divergence becomes prohibitively large. This effect underlies the improved sample efficiency demonstrated.

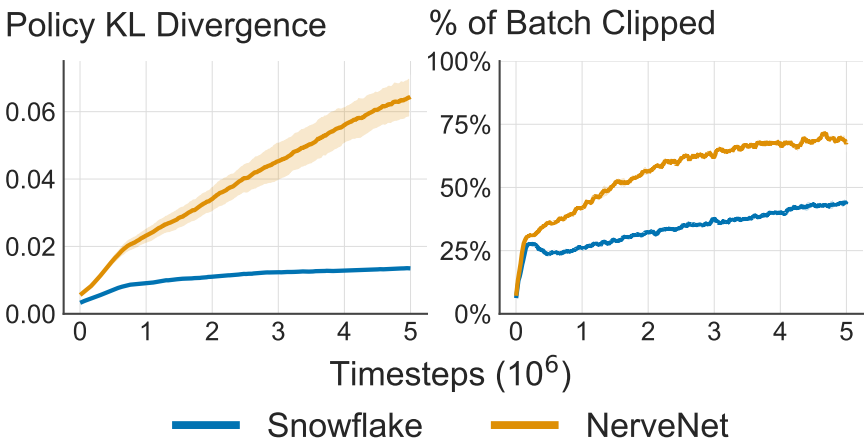


Figure 5.11: The effect of SF on policy divergence and PPO clipping on `Centipede-20`. By freezing parts of the network that overfit, SF reduces the policy KL divergence leading to less clipping during training.

5.5.4 PPO Clipping

SF's improved policy stability also reduces the amount of clipping performed by PPO across each training batch. Figure 5.11 shows the percentage of state-action pairs that are clipped for regular GNN versus SF, on the `Centipede-20` agent.

When GNN is trained without using SF, a larger percentage of state-action pairs are clipped during PPO updates—a consequence of the more significant policy divergence caused by overfitting. For PPO, if too many data points reach the

clipping limit during optimisation, the algorithm can only learn on a small fraction of the experience collected, reducing training effectiveness. One of SF's strengths is that because it reduces policy divergence, it requires less severe restrictions to keep the policy within the trust region. We believe that the combination of this effect and the ability to train well on smaller batch sizes enables SF's strong performance on the largest agents.

5.6 Conclusions and Future work

In this chapter, we proposed SF, a method that enables GNN-based policies to be trained effectively on much larger locomotive agents than was previously possible. We no longer observe a substantial difference in performance between using GNNs to represent a locomotion policy and the standard approach of using MLPs, even on the most challenging morphologies. Consequently, GNNs now offer an alternative to MLPs for more than just low-dimensional continuous control tasks. If the additional features of GNNs such as strong transfer are required, then they are likely to be a better choice. We have also provided insight into why poor scaling occurs for certain GNN architectures and why parameter freezing effectively addresses the overfitting problem we identify.

Our work is interestingly related to the literature on random embeddings. Embeddings represent feature vectors projected into a new, typically lower-dimensional space that is easier for models to process. Bingham et al. [BM01] show that multiplying features by a randomly generated matrix (e.g., with entries sampled from a Gaussian distribution) preserves similarity well and empirically attains comparable performance to principle component analysis. Wang et al. [Wan+16] use this approach to apply Bayesian Optimisation to high-dimensional datasets by randomly projecting them into a smaller subspace. For natural language applications, commonly used pre-trained embeddings (e.g., word2vec [PSM14], GloVe [Mik+13]) have been shown to offer only a small benefit over random embeddings on benchmark datasets [KB17; Dhi+17] and may offer no benefit on industry-scale data [Aro+20].

More generally, random embeddings can be induced by freezing typically-learned parameters within a model to fixed values throughout training. This approach has been explored for Transformer architectures, where fixed attention weights (either Gaussian-distributed [YSI20] or hand-crafted [RST20]) show no significant drop in performance, and even freezing intermediate feedforward layers still enables surprisingly effective learning [Lu+22]. A similar technique can also be found in common fine-tuning methods, where parameters are pre-trained on another, possibly unsupervised objective, but frozen during training except for the final layer [e.g., Yos+14; Hou+19].

One potential limitation of our work is that SF is algorithm-specific. The learning instabilities we pinpointed might be inherent to PPO only. While there are some indications of overfitting by GNNs [Ham20, p.55], more work is required to understand if our findings hold in more general settings, e.g., RL, or even supervised and unsupervised learning. It would also be interesting to investigate the potential relation between the frozen set and the task on hand. For instance, we expect that freezing the encoder will be detrimental to the performance in the multitask setting, when the tasks are similar in the behaviour space, but very different in the input space.

Another limitation of our method is that we do not generally specify how to select a frozen set, i.e., the frozen set is a hyperparameter. As with any hyperparameter, a suboptimal choice might limit the model performance. For instance, Figure 5.5 shows that freezing the node updater is detrimental to the performance. For RL, one needs to interact with the environment to make a decision, which might be prohibitive for some settings, especially real-world applications. More work is required to select the frozen set automatically to enable more sample-efficient learning. Moreover, as we mentioned in §5.3.3, freezing is a special case of tuning the learning rate, when it is set to zero throughout training. For a new task, it might be beneficial to extend the valid range of learning rates to choose from with the learning rate of zero.

However, keeping the discussed limitations in mind, we still believe that our analysis and method can facilitate future work in training GNN-based policies on even larger graphs for a broader range of learning problems.

You can recognise truth by its beauty and simplicity. When you get it right, it is obvious that it is right – at least if you have any experience – because usually what happens is that more comes out than goes in.

— Richard Feynman

6

The Role of Morphology in Graph-Based Incompatible Control

Contents

6.1 Introduction

In the previous chapter, we assume that the structure of the state and action sets is known to us and is used to induce the message-passing schema happening in GNNs.

This chapter assumes we do not have access to such information.

6.1.1 Motivation

There are two main reasons to relax the assumption of knowing the graph connectivity. First, we might have no access to the structure, e.g. in multi-agent coordination, when we do not know who has to coordinate with whom in advance. Second, the chosen structure might not be optimal.

We still assume that the state consists of objects, and we know which features belong to each object. This is important because the nodes exchange information about their annotations, and the annotation should be consistent across the nodes. In this chapter, we use continuous control benchmarks from Wang et al. [Wan+18] and Huang et al. [HMP20] as the testbed.

A key feature of GNNs is that they can process graphs of arbitrary size and thus, in principle, allow MTRL in incompatible environments. However, GNNs also have a second key feature: they allow models to condition on structural information about how state features are related, e.g., how a robot's limbs are connected. In effect, this enables practitioners to incorporate additional domain knowledge where states are described as annotated graphs. Here, a graph is a collection of annotated nodes, indicating the features of corresponding objects, and edges, indicating the relations between them. In many cases, e.g., with the robot mentioned above, such domain knowledge is readily available. This results in a structural inductive bias that restricts the model's computation graph, determining how errors backpropagate through the network.

GNNs have been applied to MTRL in continuous control environments, a staple benchmark of modern RL, by leveraging both of the key features mentioned above [Wan+18; HMP20], and also our Chapter 5. In these works, the graphs are based on agents' physical morphology, with nodes annotated with the observable features of their corresponding limbs, e.g., coordinates, angular velocities and limb type. If two limbs are physically connected, there is an edge between their

corresponding nodes. However, the assumption that it is beneficial to restrict the model's computation graph in this way has to our knowledge, not been validated.

6.1.2 Contribution

To validate whether GNNs profit from restricting the computation graph by providing an input graph structure, we conduct a series of ablations on existing GNN-based continuous control methods. The results show that removing morphological information does not harm the performance of these models. In addition, we propose Amorpheus, a new continuous control MTRL method based on transformers [Vas+17] instead of GNNs that use morphological information to define the message-passing schema. Amorpheus is motivated by the hypothesis that any benefit GNNs can extract from the morphological domain knowledge encoded in the graph is outweighed by the difficulty that the graph creates for message passing. In a sparsely connected graph, crucial state information must be communicated across multiple hops, which we hypothesise is difficult in practice to learn. Amorpheus uses transformers instead, which can be thought of as fully connected GNNs with attentional aggregation [Bat+18]. Hence, Amorpheus ignores the morphological domain knowledge but, in exchange, obviates the need to learn multi-hop communication. Similarly, in Natural Language Processing, transformers were shown to perform better without an explicit structural bias and even learn such structures from data [VB19; Gol19; Ten+19; Pet+18].

Our results on incompatible MTRL continious control benchmarks [HMP20; Wan+18] strongly support our hypothesis: Amorpheus substantially outperforms GNN-based alternatives with fixed message-passing schemas in terms of sample efficiency and final performance. In addition, Amorpheus exhibits non-trivial behaviour such as cyclic attention patterns coordinated with gaits.

We believe that the impact of our contribution is two-fold. First, disentangling the effect of explicit morphological conditioning from the modularity and the critical analysis of the current work allows us to understand the current approaches

better and lead to new insights in the future. Second, we believe that our work further raises the limitations of modern MTRL, increasing its attractiveness from the application perspective.

This chapter is based on Kurin et al. [Kur+21] in which the first author contributed the majority of the ideas, figures, text, implementation and experiments. All the other authors played significant advisory roles and contributed to writing individual sections of the paper.

6.2 Shared Modular Policies

SMP is a recent approach released while we were working on this chapter. It goes one step further from NerveNet and studies the generalisation properties of a multi-task model. SMP shows impressive results; however, it is a less principled approach than NerveNet and has several key disadvantages.

SMP is a variant of a GNN that operates only on trees. Computation is performed in two stages: top-down and bottom-up. In the first stage, information propagates level by level from the leaves to the root, with parents aggregating information from their children. In the second stage, information propagates from parents to the leaves, with parents emitting multiple messages, one per child. The policy emits actions at the second stage of the computation together with the downstream messages.

Instead of a permutation invariant aggregation, the messages are concatenated. This, as well as separate messages for the children, also injects structural bias into the model, e.g., separating the messages for the left and right parts of robots with bilateral symmetry. In addition, its message-passing schema depends on the morphology and the choice of the root node. In fact, Huang et al. [HMP20] show that the root node choice can affect performance by 15%.

To accommodate trees into our formalism, we introduce two types of edges: top-down $\mathcal{E}_{td}$ and bottom-up $\mathcal{E}_{bu}$. The labels do not have subscripts since each edge is either top-down or bottom-up. In contrast to all previous GNNs we describe in §2.3,

the computation happens level by level of a tree. First, the network computes all the messages from bottom nodes to their parents in the tree $\boldsymbol{e}'^{ji}$:

$$\begin{aligned} \boldsymbol{v}'^{i} &\leftarrow \phi^{v}_{\xi}\Big(\boldsymbol{v}^{i}, \rho^{e}_{v}\{\boldsymbol{e}'^{ki} \,|\, e^{ki} \in \mathcal{E}_{bu}\}\Big) && \forall v^{i} \in \mathcal{V}\,, \\ \boldsymbol{e}'^{ij} &\leftarrow \boldsymbol{v}'^{i} && \forall e^{ij} \in \mathcal{E}_{bu}\,. \end{aligned} \tag{6.1}$$

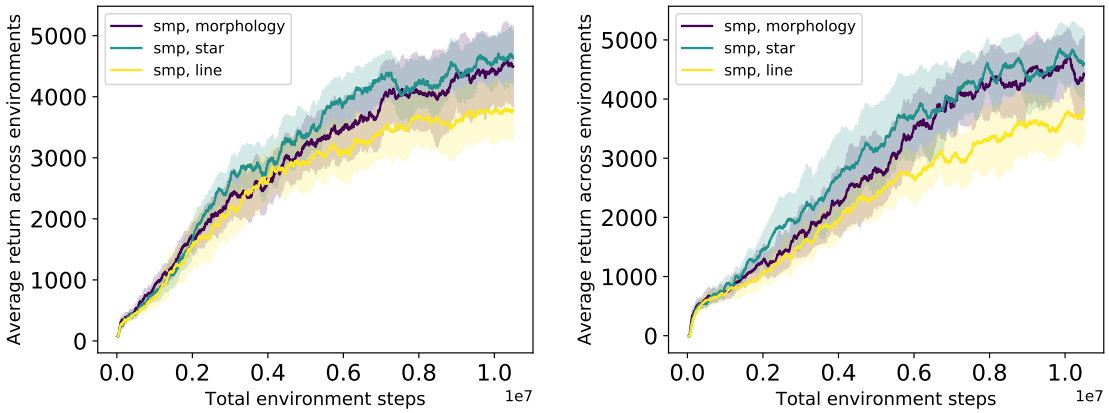
After that, level by level, the network outputs both the value for the node output $\boldsymbol{v}'^{i}$ and the message to its children $\boldsymbol{e}'^{ij}$:

$$\begin{aligned} \boldsymbol{v}'^{i} &\leftarrow \phi^{v}_{\psi}\Big(\boldsymbol{v}'^{i}, \boldsymbol{e}'^{ji}\Big) \forall e^{ij} \in \mathcal{E}_{td}\,, \\ \boldsymbol{e}'^{ij} &\leftarrow \phi^{e}_{\psi}\Big(\boldsymbol{v}'^{i}, \boldsymbol{e}'^{ji}\Big) \forall e^{ij} \in \mathcal{E}_{td}\,. \end{aligned} \tag{6.2}$$

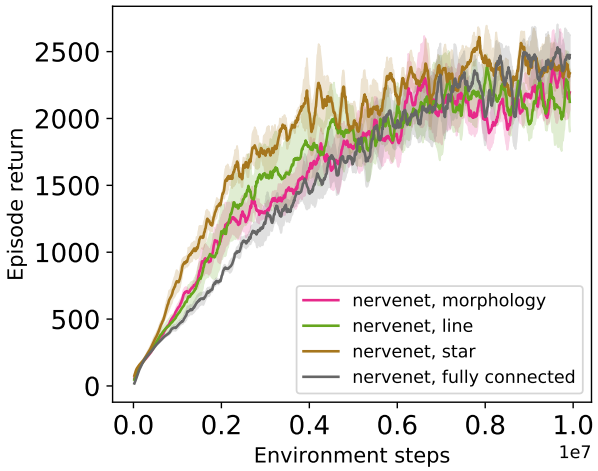
Node labels $\boldsymbol{v}'^{i}$ are used as a policy output for the environment. Top-down and bottom-up updaters are two different networks; however, there is weight sharing between ϕ^{e}_{ψ} and ϕ^{v}_{ψ}. The input is padded with zeros for the root node, which does not have incoming top-down edges, and for the leaves, which do not have the incoming bottom-up edges.

The message aggregation phase is the weakest point of this approach. Instead of doing a permutation invariant operation, SMP concatenates all incoming messages and feeds the resulting vector to the node updater when going bottom-up. For the second phase, the network outputs several messages, and children pick one of them based on their identifier among all the children of the parent node. Apart from not being permutation invariant, such message aggregation schema cannot generalise to graphs for which the maximum number of children of a node is larger than the one in the training data. For instance, this makes it impossible to test a model trained on *Walker* environment on an *Humanoid-2D* agent, since the latter has a node with four children compared to a maximum of two children in *Walker*.

SMP trains a separate model for the actor and critic. An actor outputs one action per non-root node. The critic outputs a scalar per node as well. When updating a critic, a value loss is computed independently per each node with targets using the same scalar reward from the environment.



(a) SMP, `Walker++` **(b)** SMP, `Humanoid++`



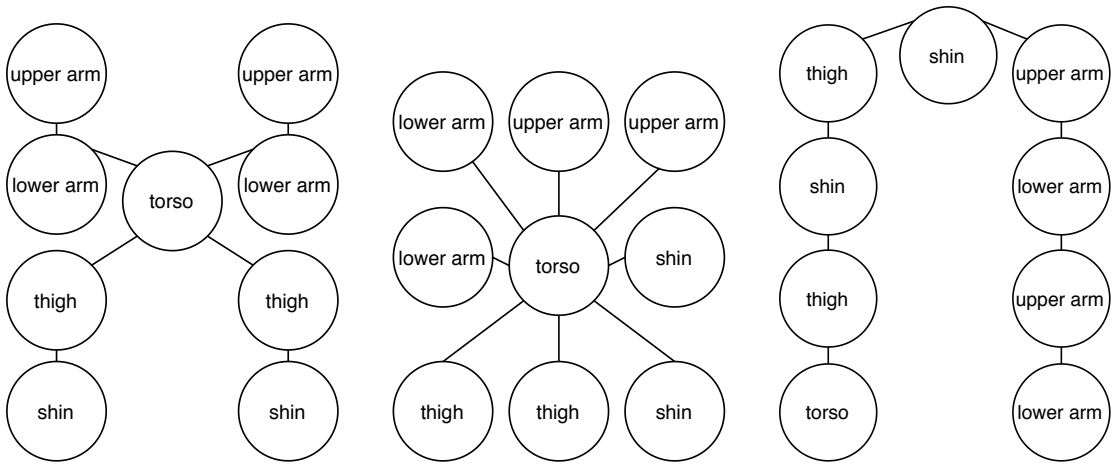
(c) NerveNet, `Walkers`

Figure 6.1: Neither SMP nor NerveNet leverage the agent's morphological information, or the positive effects are outweighed by their negative effect on message passing.

6.3 Role of Morphology in the Existing Work

In this section, we provide evidence against the belief that GNNs improve performance by exploiting information about physical morphology [HMP20; Wan+18]. Here and in the following sections, we run experiments for three random seeds and report the average undiscounted MTRL return and the standard error across the seeds.

To determine if information about the agent's morphology encoded in the relational graph structure is essential to the success of SMP, we compare its performance given complete information about the structure (morphology), given



(a) Morphology **(b)** Star **(c)** Line

Figure 6.2: Examples of graph topologies used in the structure ablation experiments.

no information about the structure (star), and given a structural bias unrelated to the agent's morphology (line). Ideally, we would also test a fully connected architecture, but SMP only works with trees. Figure 6.2 illustrates the tested topologies.

The results in Figure 6.1a and 6.1b demonstrate that, surprisingly, performance is not contingent on having information about the physical morphology. A `star` agent performs on par with the `morphology` agent, thus refuting the assumption that the method learns because it exploits information about the agent's physical morphology. The `line` agent performs worse, perhaps because the network must propagate messages even further away, and information is lost with each hop due to the finite size of the MLPs causing information bottlenecks [AY20].

We also present similar results for NerveNet. Figure 6.1c shows that all of the variants we tried perform similarly well on `Walkers` from [Wan+18], with `star` being marginally better. Since NerveNet can process non-tree graphs, we also tested a fully connected variant. This version learns more slowly at the beginning, probably because of difficulties with differentiating nodes at the aggregation step. Interestingly, in contrast to SMP, in NerveNet `line` performs on par with `morphology`. This might be symptomatic of problems with the message-passing mechanism of SMP, e.g., bottlenecks leading to information loss.

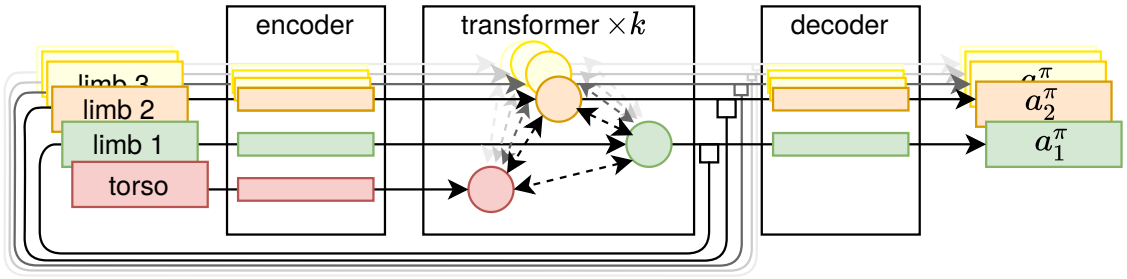


Figure 6.3: Amorpheus architecture. Lines with squares at the end denote concatenation. Arrows going separately through the encoder and decoder denote that rows of the input matrix are processed independently as batch elements. Dashed arrows denote message-passing in a Transformer block. The diagram depicts the policy network, the critic has an identical architecture, with the decoder outputs interpreted as value function values.

6.4 Amorpheus

Inspired by the results above, we propose Amorpheus, a Transformer-based method for incompatible MTRL in continuous control. Amorpheus is motivated by the hypothesis that any benefit GNNs can extract from the morphological domain knowledge encoded in the graph is outweighed by the difficulty that the graph creates for message passing. In a sparse graph, crucial state information must be communicated across multiple hops, which we hypothesise is challenging to learn in practice.

Amorpheus belongs to the encode-process-decode family of architectures [Bat+18] with a Transformer at its core. Since transformers can be seen as GNNs operating on fully connected graphs, this approach allows us to learn a message passing schema for each state and each pass separately and limits the number of message passes needed to propagate sufficient information through the graph. Multi-hop message propagation in the presence of aggregation, which could cause problems with gradient propagation and information loss, is no longer required. We implement both actor and critic in the SMP codebase [HMP20] and made our implementation available online at https://github.com/yobibyte/amorpheus. As in SMP, there is no weight sharing between the actor and the critic. Both of them consist of three parts: a linear encoder, a Transformer in the middle, and the output decoder MLP.

Figure 6.3 illustrates the Amorpheus architecture for the policy. The encoder and decoder process each node independently as if they are different elements of a mini-batch. Like SMP, the policy network has one output per graph node. The critic has the same architecture as in Figure 6.3, and, as in Huang et al. [HMP20], each critic node outputs a scalar with the value loss independently computed per node.

Similarly to NerveNet and SMP, Amorpheus is modular and can be used in incompatible environments, including those not seen in training. In contrast to SMP, which is constrained by the maximum number of children per node seen at the model initialisation in training, Amorpheus can be applied to any other morphology with no constraints on the physical connectivity.

Instead of one-hot encoding used in natural language processing, we apply a linear layer on node observations. Each node observation uses the same state representation as SMP and includes a limb type (e.g. hip or shoulder), position with a relative x coordinate of the limb with respect to the torso, positional and rotational velocities, rotations, angle, and the possible value range for the angle normalised to $[0, 1]$. We add residual connections from the input features to the decoder output to avoid the nodes forgetting their own features by the time the decoder independently computes the actions. Both actor and critic use two attention heads for each of the three Transformer layers. Layer Normalisation [BKH16] is a crucial component of transformers which we also use in Amorpheus. See Appendix A.3 for more details on the implementation.

6.5 Experiments

6.5.1 Multi-Task Learning Performance

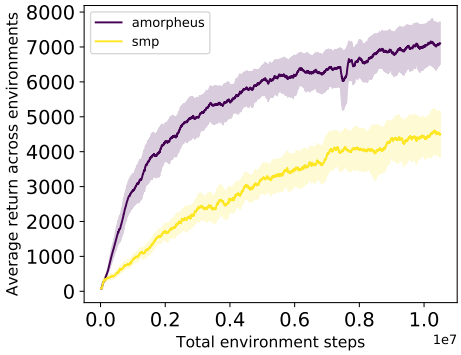
We first test Amorpheus on the set of MTRL environments proposed by Huang et al. [HMP20]. For `Walker++`, we omit flipped environments, since Huang et al. [HMP20] implement flipping on the model level. For Amorpheus, the flipped environments look identical to the original ones. Our experiments in this section are built on top of the TD3 implementation used in Huang et al. [HMP20].

Figure 6.4 supports our hypothesis that explicit morphological information encoded in graph topology is not needed to yield a single policy achieving high average returns across a set of incompatible continuous control environments. Free from the need to learn multi-hop communication and equipped with the attention mechanism, Amorpheus clearly outperforms SMP, the state-of-the-art algorithm for incompatible continuous control. Huang et al. [HMP20] report that training SMP on `Cheetah++` together with other environments makes SMP unstable. By contrast, Amorpheus has no trouble learning in this regime (Figure 6.4g and 6.4h).

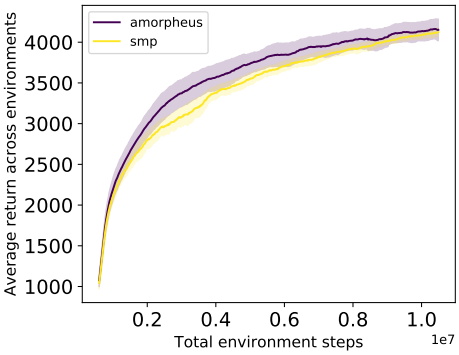
Our experiments demonstrate that node features have enough information for Amorpheus to perform the task and limb discrimination needed for successful MTRL continuous control policies. For example, a model can distinguish left from right, not from structural biases as in SMP, but from the relative position of the limb w.r.t. the root node provided in the node features.

While the total number of tasks in the SMP benchmarks is high, they all share one key characteristic. All tasks in a benchmark are built using subsets of the limbs from an archetype (e.g., `Walker++` or `Cheetah++`). To verify that our results hold more broadly, we adapted the `Walkers` benchmark [Wan+18] and compared Amorpheus with SMP and NerveNet on it. This benchmark includes five agents with different morphologies: a Hopper, a HalfCheetah, a FullCheetah, a Walker, and an Ostrich. The results in Figure 6.5 are consistent with our previous experiments, demonstrating the benefits of Amorpheus' fully-connected graph with attentional aggregation. Note that the performance of NerveNet is not directly comparable, as the observational features and the learning algorithm differ from Amorpheus and SMP. We do not test NerveNet on SMP benchmarks because the codebases are not compatible and comparing NerveNet and SMP is not the focus of the chapter. Even if we implemented NerveNet in the SMP training loop, it is unclear how the critic of NerveNet would perform in a new setting without extensive hyperparameter tuning. The original paper considers two options for the critic: one GNN-based and one MLP-based. We use the latter in Figure 6.1c as the former takes only the root node output labels as an input and is thus most likely to face difficulty learning

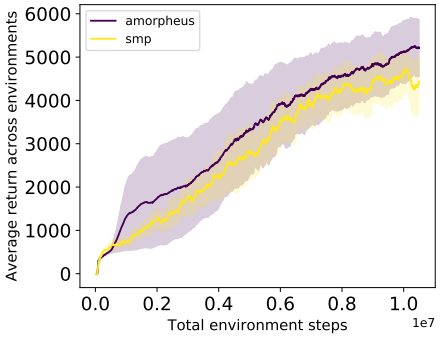
multi-hop message-passing. The MLP critic should perform better because training an MLP is easier, though it might be sample-inefficient when the number of tasks is large. For example, in `Cheetah++`, an agent would need to learn 12 different critics. Finally, NerveNet learns a separate MLP encoder per task, partially defeating the purpose of using GNN for incompatible environments.



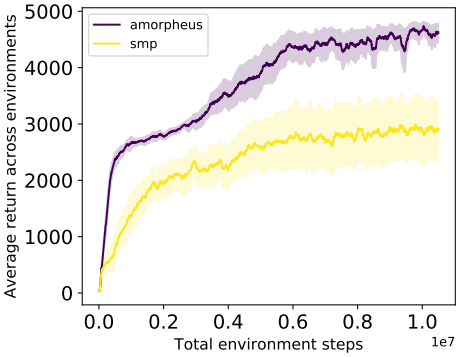
(a) Walker++



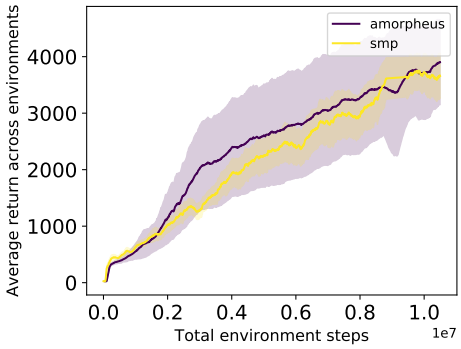
(b) Cheetah++



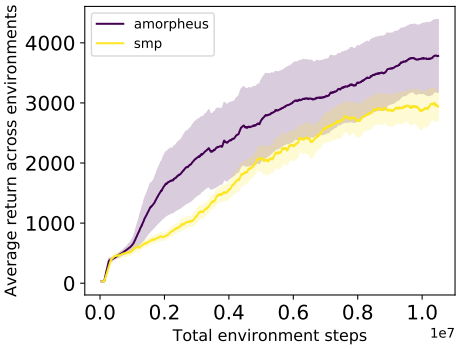
(c) Humanoid++



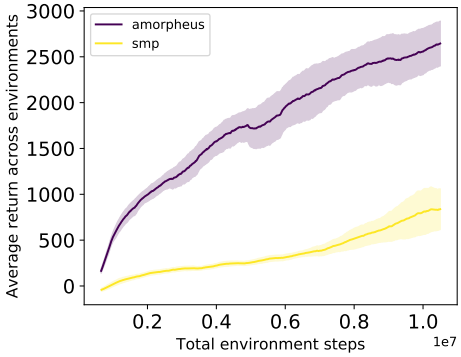
(d) Hopper++



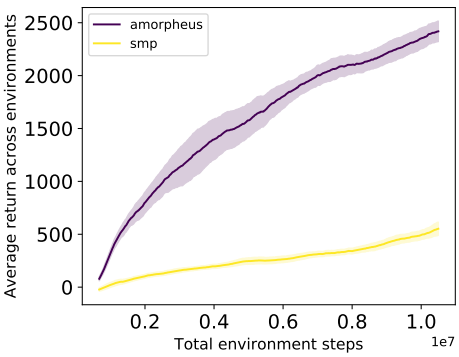
(e) Walker-Humanoid++



(f) Walker-Human-Hopper++



(g) Cheetah-Walker-Human++



(h) Cheetah-Walker-Human-Hopper++

Figure 6.4: Amorpheus consistently outperforms SMP, supporting our hypothesis that no explicit structural information is needed to learn a successful MTRL policy and that facilitated message-passing procedure results in faster learning.

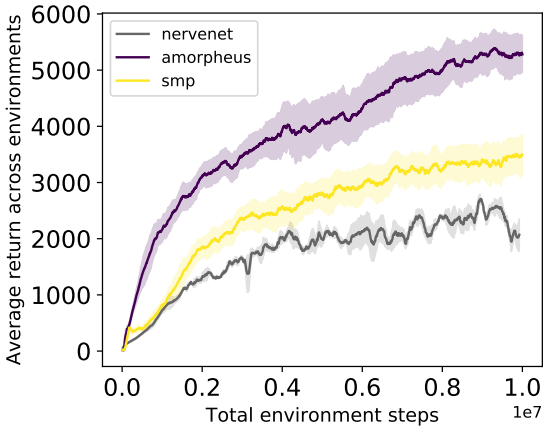


Figure 6.5: MTRL performance on `Walkers` [Wan+18].

6.5.2 Attention Mask Analysis

GNN-based policies, especially those that use attention, are more interpretable than monolithic MLP policies. We now analyse the attention masks that Amorpheus learns, showing the attention weights for different Transformer layers.

Having an implicit structure that is state-dependent is one of the benefits of Amorpheus (every node has access to other nodes' annotations, and the aggregation weights depend on the input as shown in Equation 2.17). By contrast, NerveNet and SMP have a rigid message-passing structure that does not change throughout training or a rollout. Indeed, Figure 6.6 shows a variety of masks a `Walker++` model exhibits within a `Walker-7` rollout, confirming that Amorpheus attends to different parts of the state space based on the input. Moreover, we can see that some nodes attend to the nodes distant in the morphological graph, thus using the benefits of the reduced graph diameter. For instance, the nodes from the left part of the body attend to the right.

Both Wang et al. [Wan+18] and Huang et al. [HMP20] notice periodic patterns arising in their models. Similarly, Amorpheus demonstrates cycles in attention masks, usually arising for the first layer of the Transformer. Figure 6.7 shows the column-wise sum of the attention masks coordinated with an upper-leg limb of a `Walker-7` agent. Intuitively, the column-wise sum shows how much other

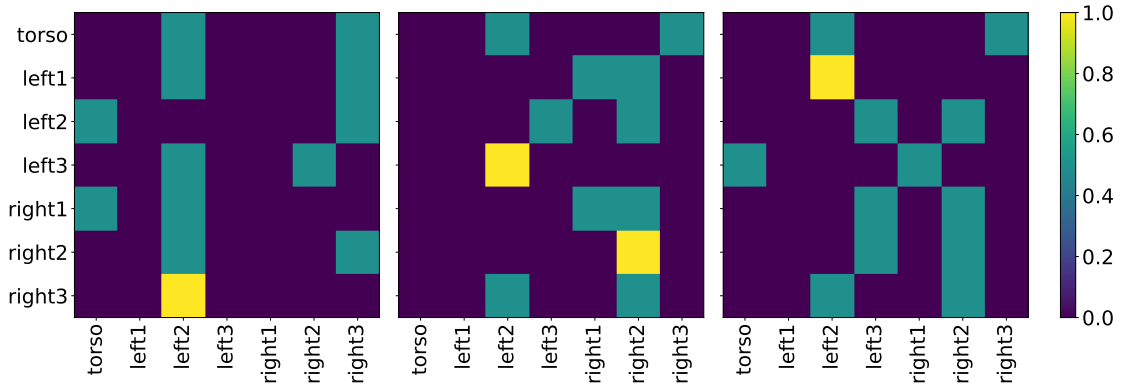


Figure 6.6: State-dependent masks of Amorpheus (3rd attention layer) within a `Walker-7` rollout.

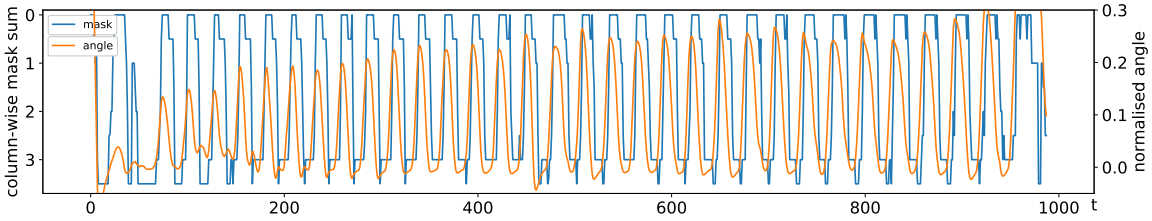


Figure 6.7: In the first attention layer of a `Walker-7` rollout, nodes attend to an upper leg (column-wise mask sum ~ 3) when the leg is near the ground (normalised angle ~ 0).

nodes are interested in the node corresponding to that column. Interestingly, attention masks in earlier layers change more slowly within a rollout than those of the downstream layers. Figure 6.8 demonstrates this phenomenon for three different `Walker++` models tested on `Walker-7`. This shows that Amorpheus might, in principle, learn a rigid structure (as in GNNs) if needed. It would be interesting to see if the model exhibits hierarchical behaviour with early layers masks being a limited set of discrete modes.

Figure 6.9 shows that actor and critic networks might attend to different elements of the observation space. Critic pays attention to the state of the torso and hip limbs. This is reasonable because the environment terminates the episode as soon as the torso position and angle go beyond some allowed region, and the hips and the torso are the most indicative of the termination event.

Figure 6.10 shows an interesting case when Amorpheus learnt the bilateral symmetry of the `Humanoid2d-9-full` agent, however, the structure is still different

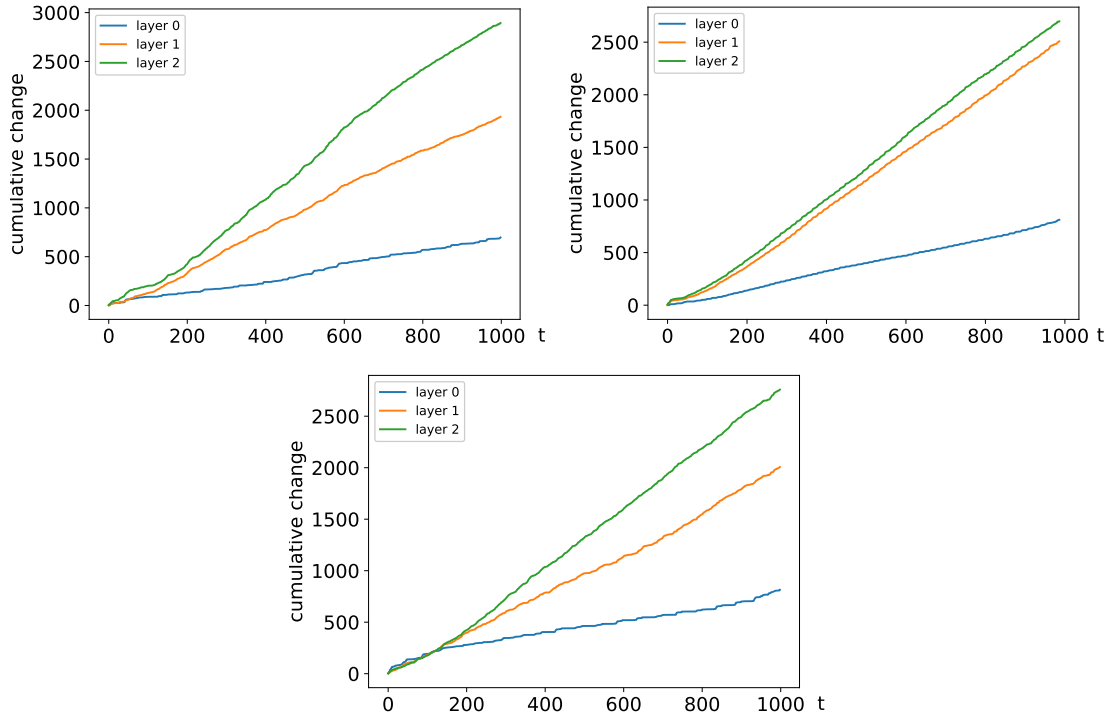


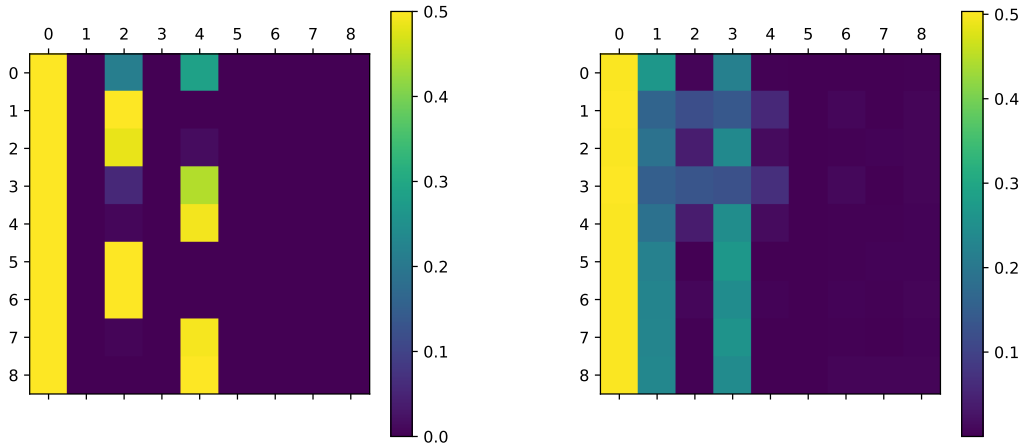
Figure 6.8: Absolute cumulative change in the attention masks for three different models on `Walker-7`. The x-axis represents a rollout in time, the y-axis shows the accumulated change in the mask values. The masks in successive layers are more prone to change compared to the early layers.

from the morphology (Figure 6.2a). Figure 6.10 shows the attention masks drawn as a directed graph: if node A attends to node B, we draw a directed edge from B to A. We can see that all the nodes attend to the root node. Apart from that, the left arm nodes attend to the left leg nodes, and the same happens with the right part of the body. We want to point out that only one run exhibited such behaviour, and this case is more of a curiosity rather than an established pattern.

Finally, we investigate how attention masks evolve over time. Early in training, the masks are spread across the whole graph. Later on, the mask weights distributions become less uniform. Figure 6.11 demonstrates this phenomenon on `Walker-7`.

6.5.3 Zero-shot generalisation

While we focused on multi-task learning in this work, we also evaluated Amorpheus in a zero-shot generalisation setting. Table 6.1 provides zero-shot generalisation



(a) Actor **(b)** Critic

Figure 6.9: In Actor and Critic, learnt implicit structures are different.

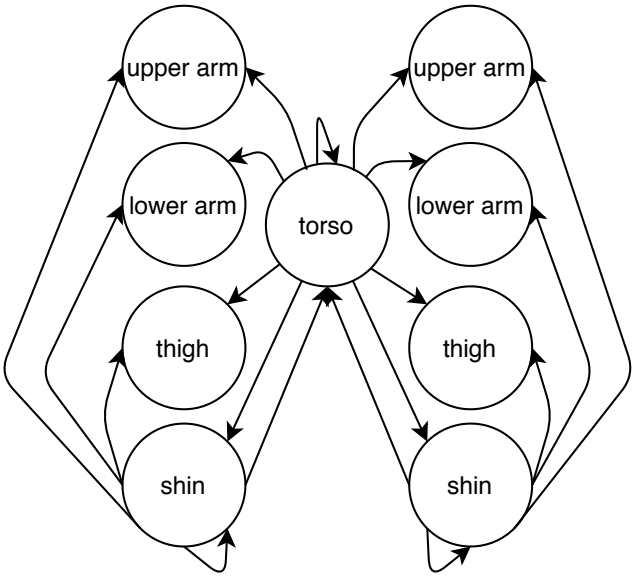
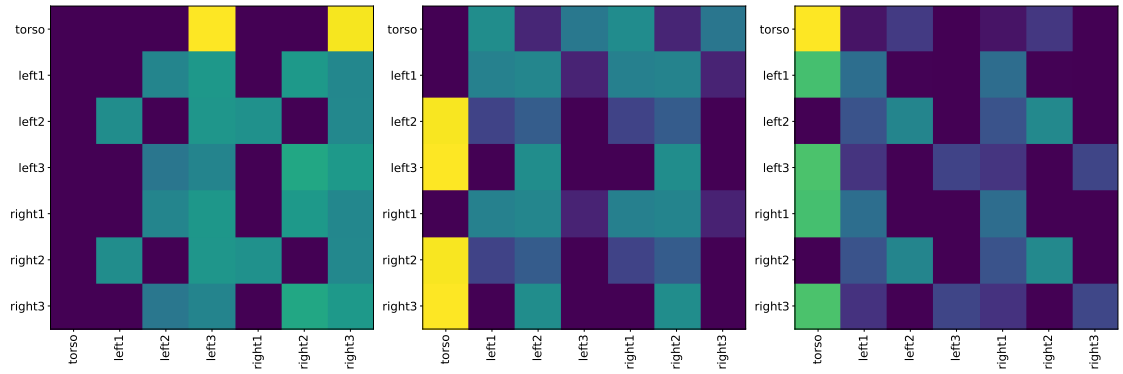
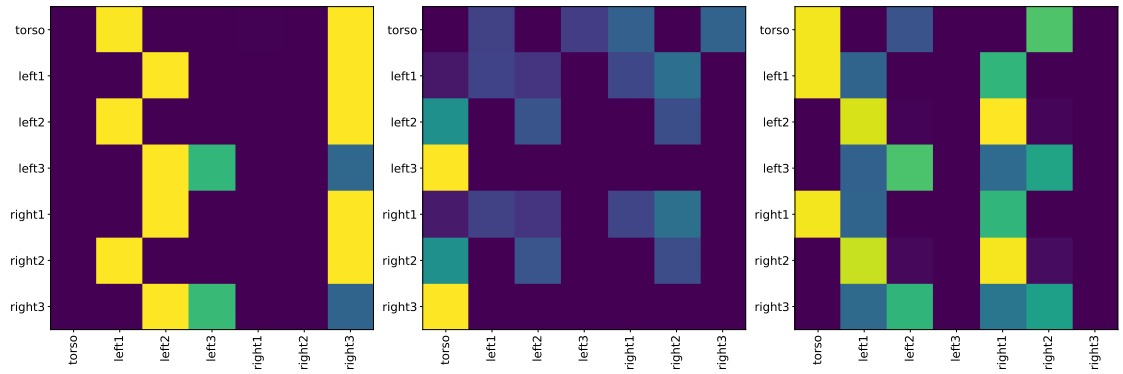


Figure 6.10: One of the seeds learns to imitate the bilateral symmetry of a `Humanoid-9`.

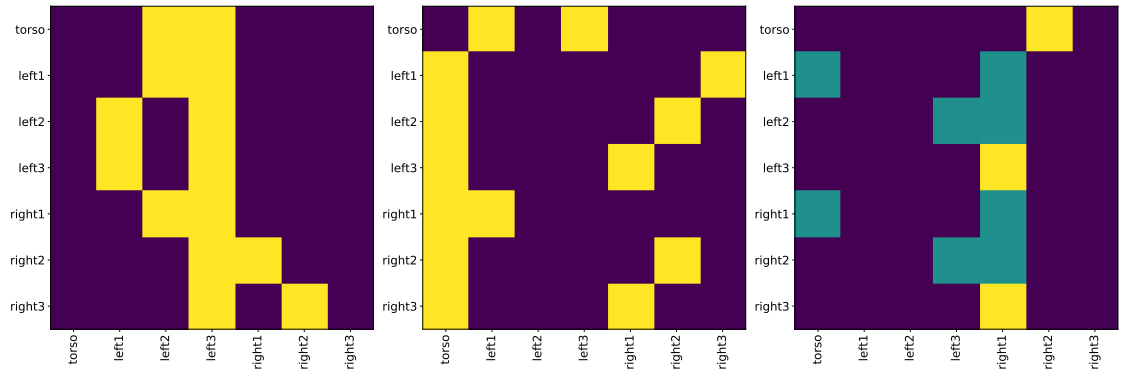
results demonstrating Amorpheus's potential. While the average values are higher for Amorpheus on 5 out of 7 benchmarks, the high variance of both methods might indicate instabilities in generalisation behaviour due to significant differences between the training and testing tasks. We believe the leading cause of these is that zero-shot generalisation behaviour is more a stroke of luck rather than an emerging property. We hypothesise that it depends on the optimality of the



(a) Beginning of training.



(b) After 2.5 mil frames.



(c) End of training.

Figure 6.11: `Walker++` masks on `Walker-7` for the 3 attention layers

training task gaits for the target task.

We further support our hypothesis with a simple experiment. First, we evaluate the models on Walker 6 with a different size of a foot limb (0.14 and 0.3 instead of the original 0.2). In this case, one SMP seed dramatically improves performance for a smaller leg (2680 vs 1950) but deteriorates significantly for a bigger one (616). In contrast, another seed gets worse for the smaller leg (1008 to 703): the effect

Table 6.1: Zero-shot generalisation results. The numbers show the average performance of three seeds evaluated on 100 rollouts and the standard error of the mean.

	Amorpheus		SMP	
walker-3-main	**666.24**	(133.66)	175.65	(157.38)
walker-6-main	**1171.35**	(832.91)	729.26	(135.60)
humanoid-2d-7-left-leg	**2821.22**	(1340.29)	2158.29	(785.33)
humanoid-2d-8-right-knee	**2717.21**	(624.80)	327.93	(125.75)
cheetah-3-balanced	**474.82**	(74.05)	156.16	(33.00)
cheetah-5-back	3417.72	(306.84)	**3820.77**	(301.95)
cheetah-6-front	5081.71	(391.08)	**6019.07**	(506.55)

is the opposite indicating that different seeds learn different gaits with different generalisation properties, depending on a target environment.

6.6 Conclusions and Future work

In this chapter, we investigated the role of explicit morphological information in graph-based continuous control. We ablated existing methods SMP and NerveNet, providing evidence against the belief that these methods improve performance by exploiting explicit morphological structure encoded in graph edges. Motivated by our findings, we presented Amorpheus, a Transformer-based method for MTRL in incompatible environments. Amorpheus obviates the need to propagate messages far away in the graph and can attend to different regions of the observations depending on the input and the particular point in training. As a result, Amorpheus clearly outperforms existing work in incompatible continuous control. In addition, Amorpheus exhibits non-trivial behaviour such as periodic cycles of attention masks coordinated with the gait. The results show that information in the node features alone is enough to learn a successful MTRL policy. We believe our results further push the boundaries of incompatible MTRL and provide valuable insights for further progress.

We believe that, in problems with rich feature spaces, transformers are a better choice due to their flexibility. Compared to many graph structure learning works [Zhu+21], transformers can operate in a setting when there is no single optimal graph, adjusting the mask based on the inputs. However, computational overhead is one major drawback of transformers and, as a result, of our method. Transformers suffer from quadratic complexity in the number of nodes, with a growing body of work addressing this issue [Tay+20]. However, the number of nodes in continuous control problems is relatively low compared to much longer sequences used in NLP [Dev+19]. Moreover, transformers are highly parallelisable, compared to SMP with the data dependency across tree levels (the tree is processed level by level, with each level taking the previous level's output as an input).

In current approaches, generalisation across structures means generalisation across different robot morphologies, e.g. across different graph topologies. We believe this overlooks another essential kind of generalisation across real-world structures: varying limb weights and lengths. This kind of information is not represented in the graph structure and should be a part of the feature space of a node. We believe that adding this kind of information might lead to better zero-shot generalisation properties making generalisation more about inferring the physical properties of the body rather than a pure stroke of luck. To further extend the work in this direction, one might infer the physical properties either using a recurrent model or an approach proposed in Yu et al. [YLT19].

We focused on investigating the effect of injecting explicit morphological information into the model. However, there are also opportunities to improve the learning algorithm itself. Potential directions of improvement include averaging gradients instead of performing sequential task updates or balancing the data for the updates based on cross-task similarities.

Truth is ever to be found in simplicity, and not in the multiplicity and confusion of things

— Isaac Newton

7

Generalisable Branching Heuristic for a SAT Solver

Contents

7.1 Introduction

This chapter studies the generalisation and transfer properties of incompatible MTRL using the SAT problem as a testbed. SAT is a flexible problem which allows us to vary state and action set sizes, and family distributions and has a well-tuned established baseline – Variable State Independent Decaying Sum (VSIDS) heuristic [Mos+01].

7.1.1 Motivation

SAT is a fundamental problem for both industry and academia that impacts various fields, including circuit design, computer security, artificial intelligence and automatic theorem proving. As a result, modern SAT solvers are well-crafted, sophisticated, reliable pieces of software that can scale to problems with hundreds of thousands of variables [OSC09].

SAT is known to be NP-complete [Kar72], and most state-of-the-art open-source and commercial solvers rely on multiple *heuristics* to speed up the exhaustive search, which is otherwise intractable. These heuristics are usually meticulously crafted using expert domain knowledge and are often iteratively refined via trial and error. This chapter investigates how we can use machine learning to improve upon an existing branching heuristic without leveraging domain expertise.

7.1.2 Contribution

We present Graph-Q-SAT, a branching heuristic in a Conflict Driven Clause Learning [SS99; JS97, p. CDCL] SAT solver trained with value-based reinforcement learning (RL), based on deep Q-networks [Mni+15, DQN]. Graph-Q-SAT uses a graph representation of SAT problems similar to Selsam et al. [Sel+19], which provides permutation and variable relabeling invariance. Graph-Q-SAT uses a GNN [GMS05; Bat+18] as a function approximator to provide generalisation as well as support for a dynamic state-action space. Graph-Q-SAT uses a simple state representation and a binary reward that requires no feature engineering or problem

domain knowledge. Graph-Q-SAT modifies only part of the CDCL-based solver, keeping it *complete*, i.e., always yielding a correct solution.

We demonstrate that Graph-Q-SAT outperforms VSIDS [Mos+01], the most frequently used CDCL branching heuristic, reducing the number of iterations required to solve SAT problems by 2-3x. Graph-Q-SAT is trained to examine the structure of the particular problem instance to make better decisions at the beginning of the search, whereas the VSIDS heuristic suffers from poor decisions during the warm-up period.

Our work primarily focuses on the machine learning perspective, and thus more work would be required to apply Graph-Q-SAT in industrial-scale SAT settings. However, Graph-Q-SAT exhibits intriguing properties which might eventually be useful for practical applications. We show that our method generalises to problems five times larger than those it was trained on. We also show that Graph-Q-SAT generalises across problem types from satisfiable (SAT) to unsatisfiable instances (unSAT). We show that reducing the number of iterations, in turn, could reduce wall clock time, the ultimate goal when designing heuristics. We also show positive zero-shot transfer properties of Graph-Q-SAT when the testing task family is different from the training one. Finally, we show that some improvements are achieved even when training is limited to a single SAT problem, demonstrating data efficiency.

This chapter is based on Kurin et al. [Kur+20] in which the first author contributed the majority of the ideas, figures, text, implementation and experiments. All the other authors played significant advisory roles and contributed to writing individual sections of the paper.

7.2 Boolean Satisfiability Problem

A SAT problem involves finding variable assignments that satisfy a propositional logic formula or showing that such an assignment does not exist. A propositional formula is a Boolean expression, including Boolean variables, ANDs, ORs and negations. x or 'NOT x' makes up a literal. It is convenient to represent Boolean

formulas in conjunctive normal form (CNF), i.e., conjunctions (AND) of clauses, where a clause is a disjunction (OR) of literals. An example of a CNF is $(x_1 \vee \neg x_2) \wedge (x_2 \vee \neg x_3)$, where $\wedge, \vee, \neg$ are AND, OR, and negation respectively. This CNF has two clauses: $(x_1 \vee \neg x_2)$ and $(x_2 \vee \neg x_3)$. In this work, we use SAT to denote both the Boolean Satisfiability problem and a satisfiable instance, which should be clear from the context. We use unSAT to denote unsatisfiable instances.

There are many types of SAT solvers. We focus on CDCL solvers, MiniSat [SE05] in particular because it is an open-source, minimal, but powerful implementation. A CDCL solver repeats the following steps: every iteration, it chooses a literal and assigns a variable a binary value. This is called a *decision*. Then, the solver simplifies the formula, building an implication graph, and checks whether a conflict emerged. Given a conflict, the solver can infer (learn) new clauses and backtrack to the variable assignments where the newly learned clause becomes unit (consisting of a single literal), forcing a variable assignment which avoids the previous conflict. Sometimes, the CDCL solver undoes all the variable assignments keeping the learned clauses to escape futile regions of the search space. This is called a restart.

We focus on the *branching* heuristic because it is one of the most heavily used during the solution procedure. The branching heuristic is responsible for picking the variable and assigning some value to it. VSIDS [Mos+01] is one of the most used CDCL branching heuristics. It is a counter-based heuristic that keeps a scalar value for each literal or variable (MiniSat uses the latter). These values are increased every time a variable is involved in a conflict. The algorithm behaves greedily with respect to these values called *activities*. Activities are usually initialised with zeros [Lia+15].

7.3 Graph-Q-SAT

We use the MDP formalism for our purposes. Each SAT problem is an MDP sampled from a distribution of SAT problems of a specific family (e.g., random 3-SAT or graph colouring). Moreover, each problem is either satisfiable or unsatisfiable. Hence, a task is defined as follows: $\tau \sim \mathcal{D}(\phi, (un)SAT, n_{vars}, n_{clauses})$, where $\mathcal{D}$

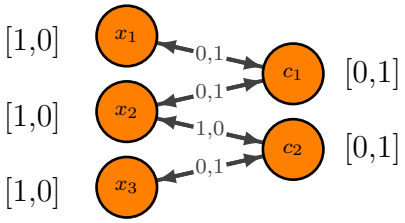


Figure 7.1: Bipartite graph representation of the Boolean formula $(x_1 \vee x_2) \wedge (\neg x_2 \vee x_3)$. The numbers next to the vertices distinguish variables and clauses. Edge labels encode literal polarities.

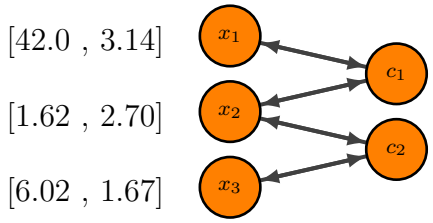


Figure 7.2: Q-function values for setting variables to *true* and *false* respectively. Taking arg max across all Q-values of variable nodes gives an action.

is the distribution of SAT problems with ϕ defining the task family, the second argument defining the problem satisfiability and n_{vars} and $n_{clauses}$ are the number of variables and clauses respectively. Each state of the MDP consists of unassigned variables and unsatisfied clauses containing these variables. The MDP is episodic, and a terminal state is reached when a satisfying assignment is found, or all possible options have been exhausted, proving unSAT. The action space includes two actions for each unassigned variable: assigning it to true or false. We build upon the MiniSat-based environment of [WR18]. It takes the actions, modifies its implication graph internally and returns a new state containing newly learned clauses without the variables removed during propagation.

Strictly speaking, this state is not fully observable. In the case of a conflict, the solver undoes the assignments for variables not observed by the agent. However, in practice, this should not inhibit the goal of quickly pruning the search tree: the information in the state is enough to pick a variable that leads to more propagation in the remaining formula.

We use a simple reward function: the agent gets a negative reward of p for each non-terminal transition and 0 for reaching the terminal state. This reward encourages an agent to finish an episode as quickly as possible and does not require elaborate reward shaping.

7.3.1 State Representation

We represent a SAT problem as a graph similar to Selsam et al. [Sel+19]. We make it more compact, using vertices to denote variables instead of literals. We use vertices to encode clauses as well. As Figure 7.1 shows, our state representation is simple and does not require feature engineering. An edge (x_i, c_i) means that a clause c_i contains literal x_i. If a literal contains a negation, a corresponding edge has a $[1, 0]$ label and $[0, 1]$ otherwise. GNNs process directed graphs, so we create two directed edges with the same labels: from a variable to a clause and vice-versa. Vertex features are two-dimensional one-hot vectors, denoting either a variable or a clause. We do not provide any other information to the model. The global attribute input is empty and is only used for message passing.

7.3.2 Q-Function Representation

We use the Encode-Process-Decode architecture from Battaglia et al. [Bat+18]. Encoder and decoder are independent graph networks, i.e. MLPs taking the whole vertex or edge feature matrix as a batch without message passing. We call the middle part 'the core'. The core output is concatenated with the output of the encoder and gets fed to the core again. Like Bapst et al. [Bap+19], our GNN labels variable vertices with Q-values. Each variable vertex has two actions: set the variable to true or false, as shown in Figure 7.2. We choose the action that gives the maximal Q-value across all variable vertices (see Algorithm 1). The graph contains only unassigned variables, so all actions are valid. We use DQN with standard techniques such as memory replay, target network, and ϵ-greedy exploration. To expose the agent to more episodes and prevent it from getting stuck, we cap the maximum number of actions per episode using an *episode length* parameter as in *gym* [Bro+16].

7.3.3 Training and Evaluation

We train our agent using Random 3-SAT instances from the SATLIB benchmark [HS00]. We split these data into training, validation, and test sets to measure

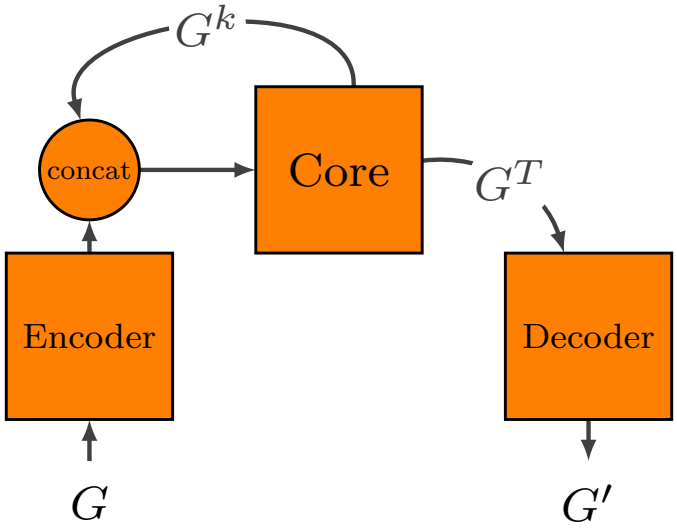


Figure 7.3: Encode-Process-Decode architecture. Encoder and decoder are independent graph networks, i.e. MLPs taking the whole vertex/edge data array as a batch. k is the index of a message passing iteration. When concatenating for the first time, encoder output is concatenated with zeros.

Algorithm 1 Graph-Q-SAT Action Selection

Input: graph network GNN_θ, state graph $G_s := (V, E, \mathcal{E})$,
with vertex features $V = [V_{vars}, V_{clauses}]$, edge features E, and graph topology $\mathcal{E}$.

$V', E', U' = GN(V, E, U)$;
VarIndex,VarPolarity $= \arg\max_{ij} V'_{vars}$;
Return VarIndex,VarPolarity;

generalisation. To illustrate the problem complexities, Table 7.1 provides the number of steps it takes MiniSat to solve the problem. Each random 3-SAT problem is denoted as SAT-X-Y or unSAT-X-Y, where SAT means that all problems are satisfiable, and unSAT means all problems are unsatisfiable. X and Y stand for the number of variables and clauses in the initial formula. We provide more details about the datasets in Appendix A.4.1.

While random 3-SAT problems have relatively few variables and clauses, they have an interesting property that makes them more challenging for a solver. For this dataset, the ratio of clauses to variables is close to 4.3 to 1, which is near the *phase transition* at which it is hard to say whether the problem is SAT or

Table 7.1: Number of MiniSat iterations (no restarts) to solve random 3-SAT instances.

dataset	median	mean
SAT 50-218	38	42
SAT 100-430	232	286
SAT 250-1065	62,192	76,120
unSAT 50-128	68	68
unSAT 100-430	587	596
unSAT 250-1065	178,956	182,799

unSAT [CKT91]. In 3-SAT problems, each clause has precisely three variables. However, learned clauses might be of arbitrary size.

We use Median Relative Iteration Reduction (MRIR) w.r.t. MiniSat as our main performance metric: the number of iterations it takes MiniSat to solve a problem divided by Graph-Q-SAT's number of iterations. Similarly to the *median human normalised score* adopted in the Atari domain [Hes+18], we use the median instead of the mean to avoid skew from outliers. By one iteration, we mean one *decision*, i.e., choosing a variable and setting it to a value. We compare ourselves with the best MiniSat results, having run MiniSat with and without restarts. We cap the number of decisions our method takes at the beginning of the solution procedure, and then we give control to MiniSat.

When training, we evaluate the model every 1000 batch updates on the validation instances and pick the model with the best validation results. After that, we evaluate this model on the test set and report the results. For each model, we do five training runs and report the average MRIR results, the maximum, and the minimum. Algorithm 2 denotes our training procedure. We provide all the hyperparameters needed to reproduce our results in Appendix A.4. We released our experimental code as well as the MiniSat *gym* environment: https://github.com/nvidia/graphqsat.

7.4 Experiments

In this section, we present empirical results for Graph-Q-SAT.

Table 7.2: Graph-Q-SAT MRIR trained on SAT-50-218. SAT-50-218 results are for a separate test set.

dataset	mean	min	max
SAT 50-218	2.46	2.26	2.72
SAT 100-430	3.94	3.53	4.41
SAT 250-1065	3.91	2.88	5.22
unSAT 50-128	2.34	2.07	2.51
unSAT 100-430	2.24	1.85	2.66
unSAT 250-1065	1.54	1.30	1.64

7.4.1 Improving upon VSIDS

In our first experiment, we consider whether it is possible to improve upon VSIDS using no domain knowledge, a simple state representation, and a simple reward function. The first row in Table 7.2 gives a positive answer to that question. DQN equipped with a GNN solves the problems in fewer than half the iterations of MiniSat. Graph-Q-SAT makes decisions resulting in more propagations, i.e., inferring variable values based on other variable assignments and clauses. This helps Graph-Q-SAT prune the search tree faster. For SAT-50-218, Graph-Q-SAT does on average 2.44 more propagations than MiniSat (6.62 versus 4.18). Figure 7.4 shows that on average using Graph-Q-SAT leads to more propagations per step than VSIDS.

These results raise the question: Why does Graph-Q-SAT outperform VSIDS? VSIDS is a counter-based heuristic that takes time to warm up. On the other hand, our model perceives the whole problem graph and can make more informed decisions from the beginning. To test this hypothesis, we vary the number of decisions our model makes at the beginning of the solution procedure before we hand the control back to VSIDS. The results in Figure 7.5 support this hypothesis. Even if our model is used for only the first ten iterations, it still improves performance over VSIDS.

One strength of Graph-Q-SAT is that VSIDS keeps being updated while the decisions are made with Graph-Q-SAT. We believe that Graph-Q-SAT complements VSIDS by providing better quality decisions in the initial phase while VSIDS is

Algorithm 2 Graph-Q-SAT Training Procedure

Input: Set of tasks $\mathcal{S} \sim \mathcal{D}(\phi, (un)SAT, n_{vars}, n_{clauses})$ split into $\{\mathcal{S}_{train}, \mathcal{S}_{validation}, \mathcal{S}_{test}\}$, ϕ is the task family (e.g. random 3-SAT, graph coloring). All hyperparameters are from Table A.1.

Randomly Initialise Q-network GNN_θ;

UPDATES = 0;

TOTALENVSTEPS = 0;

repeat

  repeat

    Sample a SAT problem $p \sim \mathcal{S}_{train}$;

    Initialise the environment ENV = SATENV(P);

    Reset the environment S = ENV.RESET();

    take action:

    $$\text{A} = \begin{cases} \text{RANDOM}(\mathcal{A}), \text{with probability } \epsilon \\ \text{SELECTACTION(S)}, \text{with probability } 1-\epsilon \end{cases}$$

    Take env step S', R, DONE = ENV.STEP(A);

    TOTALENVSTEPS +=1;

    dump experience BUFFER.ADD(S, S', R, DONE, A);

    if TOTALENVSTEPS mod UPDATEFREQ == 0; **then**

      Do a DQN update;

    end if

    if TOTALENVSTEPS mod VALIDATEFREQ == 0; **then**

      Evaluate GNN_θ on $\mathcal{S}_{validation}$;

    end if

  until Proved SAT/unSAT (DONE is *True*)

until UPDATES == TOTALBATCHUPDATES

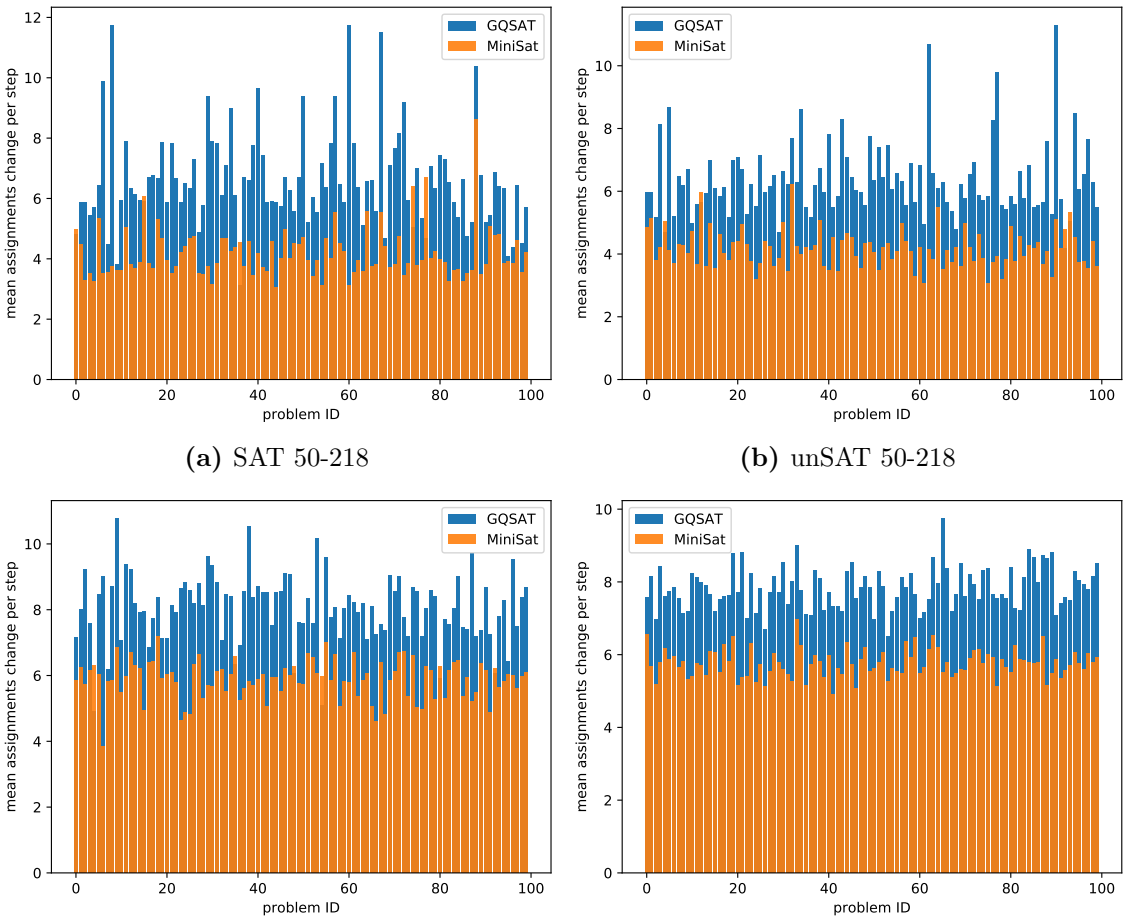
Pick the best model GNN_θ given validation scores;

Test the model GNN_θ on $\mathcal{S}_{test}$;

warming up. Capping the number of model calls also significantly reduces the main bottleneck of our approach – wall clock time spent on model evaluation.

7.4.2 Generalisation Properties of Graph-Q-SAT

Next, we consider Graph-Q-SAT's generalisation properties.



(a) SAT 50-218 **(b)** unSAT 50-218

(c) SAT 100-430 **(d)** unSAT 100-430

Figure 7.4: Average number of variable assignments change per step for (un)SAT-50-218 and (un)SAT-100-430.

7.4.2.1 Generalisation across Problem Sizes

Table 7.2 shows that Graph-Q-SAT has no difficulty generalising to larger problems, showing almost 4x improvement in iterations for a dataset 5 times bigger than the training set. Graph-Q-SAT on average leads to more variable assignments changes per step, e.g., 7.58 vs 5.89 on SAT-100-430 (see Figure 7.4 for details). It might seem surprising that the model performs better for larger problems. However, increasing scores for different problem sizes might also mean that the base solver scales worse than our method for this benchmark.

7.4.2.2 Generalisation from SAT to unSAT

An essential characteristic of Graph-Q-SAT is that the problem formulation and representation make it possible to solve unSAT problems when training only on SAT, which is problematic for some existing approaches [Sel+19]. The performance is, however, worse than the performance on satisfiable problems. On the one hand, SAT and unSAT problems are different. When the solver finds one satisfying assignment, the problem is solved. For unSAT, the algorithm must exhaust all possible options to prove that there is no such assignment. On the other hand, there is one crucial similarity between the two: an algorithm has to prune the search tree as fast as possible. Our measurements of the average number of propagations per step demonstrate that Graph-Q-SAT learns how to prune the tree more efficiently than VSIDS (6.36 vs 4.17 for unSAT-50-218, see Figure 7.4 for details).

7.4.2.3 Transfer across Task Families

So far, we have examined the generalisation properties of Graph-Q-SAT varying only the last three arguments of the task distribution defined in § 7.3 (SAT or unSAT, $n_{vars}, n_{clauses}$). In this section we go one step further and study Graph-Q-SAT's *zero-shot transfer* to a new task family ϕ.

This is a challenging problem. SAT problems have distinct structures, e.g., the graph representation of a random 3-SAT problem looks different than that of a graph colouring problem. GNNs learn local graph properties, i.e. how neighbouring entities' features have a global implication on Q-values. It is reasonable to expect a performance drop when changing the task family ϕ. However, the magnitude of the drop indicates the method's ability to transfer across task families. Therefore, we evaluate a model trained on SAT-50-218 on the flat graph colouring benchmark from SATLIB [HS00]. All the problems in the benchmark are satisfiable.

Table 7.3 shows positive transfer for Graph-Q-SAT on the graph colouring benchmark, with MRIR above 1 in five out of eight cases. As expected, MRIR is

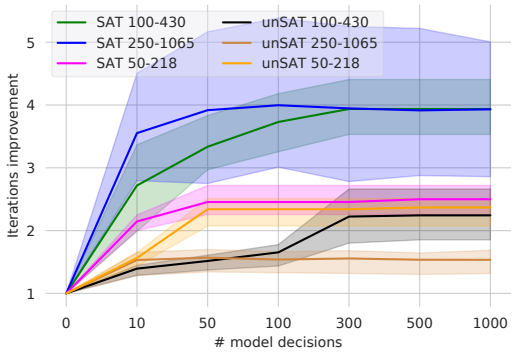


Figure 7.5: Graph-Q-SAT number of maximum first decisions vs performance. Graph-Q-SAT shows improvement starting from 10 iterations confirming our hypothesis of the VSIDS initialisation problem. The shades mark *min* and *max* values.

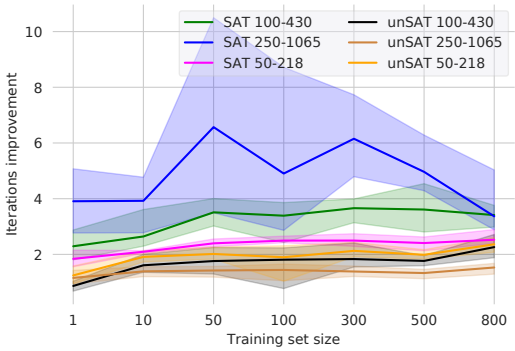


Figure 7.6: Dataset size effect on generalisation. While Graph-Q-SAT profits from more data in most cases, it can generalise even from one problem instance. The model is trained on SAT-50-218. The shades mark *min* and *max* values.

lower if than in Table 7.2, where the model was evaluated on the tasks sampled from the same distribution.

Training directly on the graph colouring benchmark indeed improves performance. Graph colouring benchmarks have only 100 problems each, so we do not split them into training/validation/test sets using *flat-75-180* for training and *flat-100-239* to do model selection. Table 7.4 shows that Graph-Q-SAT, trained on *flat75-180* shows higher MRIR compared to the transferred model.

Another intriguing property of Graph-Q-SAT generalisation is that sometimes Graph-Q-SAT shows better performance when generalising in comparison to training from scratch. Learning on SAT-100-430 requires more resources, does not generalise as well, and is generally less stable than training on SAT-50-218 and then transferring to SAT-100-430 and SAT-250-1065. Hence, generalising with Graph-Q-SAT not only reduces the time and samples spent on training but yields models hardly achievable by learning. We suppose the reason is that transfer is not directly affected by all the issues an RL agent faces when training: higher variance in the returns caused by longer episodes, challenges for temporal credit assignment, and difficulties with exploration.

Table 7.3: SAT-50 model's performance on SATLIB flat graph colouring benchmark. The comparison is w.r.t. MiniSat with restarts since MiniSat performs better in this mode for this benchmark.

dataset	variables	clauses	Graph-Q-SAT MRIR		
			average	min	max
30-60	90	300	1.51	1.25	1.65
50-115	150	545	1.36	0.47	1.80
75-180	225	840	1.40	0.31	2.06
100-239	300	1117	1.44	0.31	2.38
125-301	375	1403	1.02	0.32	1.87
150-360	450	1680	0.76	0.37	1.40
175-417	525	1951	0.67	0.44	1.36
200-479	600	2237	0.67	0.54	0.87

7.4.3 Data Efficiency

We design our next experiment to understand how many different SAT problems Graph-Q-SAT needs to learn from. We varied the SAT-50-218 training set from a single problem to 800 problems. Figure 7.6 shows that Graph-Q-SAT is extremely data efficient. Having more data helps in most cases, but even with a single problem, Graph-Q-SAT generalises across problem sizes and to unSAT instances. This should allow Graph-Q-SAT to generalise to new benchmarks without access to many problems from them. We assume that Graph-Q-SAT's data efficiency is one of the benefits of using RL. The environment allows the agent to explore diverse regions of state-action space, making it possible to learn valuable policies even from a single instance. In supervised learning, data diversity is addressed at the training data generation step.

7.4.4 Wall-Clock Time Bottleneck

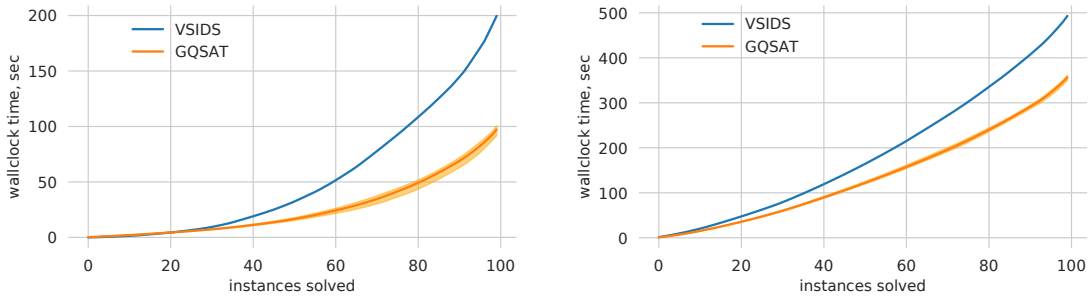
The main goal of this work is to show that RL can learn a value function used as a branching heuristic in a SAT solver and to study this model's generalisation properties. In its current form, more work would be required to apply Graph-Q-SAT in an industrial setting, where wall-clock time is the metric of success,

Table 7.4: Graph-Q-SAT MRIR on graph coloring (five training runs on 75-180, model selection with 100-239).

dataset	Graph-Q-SAT MRIR		
	average	min	max
75-180	2.44	2.25	2.70
100-239	2.89	2.77	2.98
30-60	1.74	1.33	2.00
50-115	2.08	2.00	2.13
125-301	2.43	2.20	2.66
150-360	2.07	2.00	2.11
175-417	1.98	1.69	2.21
200-479	1.70	1.38	1.98

and problem sizes are extremely large. However, we believe that Graph-Q-SAT should be of interest to the SAT community because the reduction of iterations can reduce wall clock time when the number of saved iterations is large enough to tolerate the network inference timings.

Figure 7.7 demonstrates that reducing the number of iterations together and limiting the number of model calls results in wall clock time improvements for the datasets where the number of saved iterations is large enough to tolerate the network inference timings (SAT-250 and unSAT-250 in our case). More generally, we can



(a) SAT-250 **(b)** unSAT-250

Figure 7.7: Graph-Q-SAT's MRIR improvement (10 model calls) results in the wall clock time reduction. The curves show the averaged performance across five runs, with the shade denoting the worst and the best runs.

anticipate the settings in which Graph-Q-SAT will yield an improvement in wall clock

time over VSIDS by analysing the factors contributing to their runtime performance. Assuming VSIDS's computational cost is negligible, we can compute the wall clock time of MiniSat with VSIDS as follows: $W_{\text{VSIDS}} = \sum_{t=1}^{T} P(t)$, where $P(t)$ is the unit propagation time (a procedure of formula simplification after the branching decision is made). Similarly, Graph-Q-SAT saves some fraction of iterations at the cost of added neural network inference time: $W_{\text{Graph-}Q\text{-SAT}} = \sum_{t=1}^{T/S} P(t) + \sum_{t=1}^{K} I(t)$, where S is the reduction of the number of iterations, K is the number of our model forward passes ($K \ll T$ for larger problems), and $I(t)$ is the network inference time. Thus, Graph-Q-SAT leads to wall clock speed-ups when the total inference time stays below the time spent on propagation for the reduced number of VSIDS decisions. This seems plausible, assuming that T's growth is unbounded, $K << T$ and linear dependence of $I(t)$ on the number of vertices. To check the linear dependence, we generated 10^5 graphs with characteristics similar to random 3-SAT problems (bipartite graph, each variable is connected to 13 clauses, and clause/variable ratio is 4). Figure 7.8 demonstrates that the dependence is, indeed, linear.

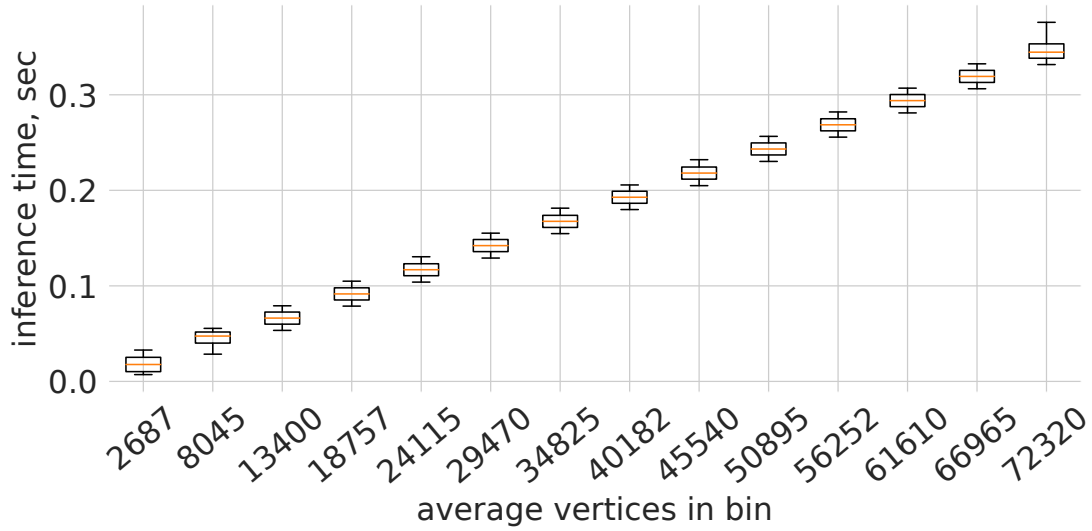


Figure 7.8: Graph-Q-SAT inference time linearly depends on the number of vertices in the graph.

7.5 Related Work

Using machine learning for the SAT problem is not a new idea [GP14; HW09; FB12; Sin+09; Xu+08; Lia+16]. Recently, SAT has attracted interest in the deep learning community. There are two main approaches: solving a problem end-to-end or learning heuristics while keeping the algorithm backbone the same. Selsam et al. [Sel+19, NeuroSAT] take the end-to-end supervised learning approach demonstrating that GNN can generalise to SAT problems bigger than those used for training. NeuroSAT finds satisfying assignments for the SAT formulae and thus cannot generalise from SAT to unSAT problems. Moreover, the method is incomplete and might generate incorrect results, which is extremely important, especially for unSAT problems. Selsam et al. [SB19] modify NeuroSAT and integrate it into popular SAT solvers to improve timing on the SATCOMP-2018 benchmark. While the approach shows its potential to scale to large problems, it requires an extensive training set, including over 150,000 data points. Amizadeh et al. [AMW19] propose an end-to-end GNN architecture to solve circuit-SAT problems. Unfortunately, while their model never produces false positives, it cannot solve unSAT problems.

The following methods take the other approach: learning a branching heuristic instead of learning an algorithm end-to-end. Jaszczur et al. [JLM20] take the supervised learning approach using the same graph representation as Selsam et al. [Sel+19]. The authors show a positive effect of combining the DPLL/CDCL solver with the learnt model. As in Selsam et al. [Sel+19], their approach requires diligent crafting of the test set. Unfortunately, the authors do not compare their approach to the VSIDS heuristic, a crucial component of CDCL [KSS11]. Wang et al. [WR18], whose environment we took as a starting point, show that DQN does not generalise for 20-91 3-SAT problems, whereas Alpha(Go) Zero [Sil+17] does. Our results show that the issue is related to state representation. They use CNNs, which are not invariant to variable renaming or permutations. Moreover, this architecture requires a fixed input size making it infeasible when applied to problems with different numbers of variables or clauses.

Yolcu et al. [YP19] use REINFORCE [Wil92] to learn the variable selection heuristic of a local search SAT solver [SKC93]. Their algorithm is an incomplete solver and cannot work with unsatisfiable instances. They also investigate the generalisation over problem sizes on random instances near the phase transition. However, in this experiment, the training problems have only ten variables, and the number of variables in the test set does not exceed 80, with the success ratio of the algorithm staying below the baseline for the latter case.

Lederman et al. [Led+20] train a REINFORCE [Wil92] agent applying GNNs to replace the branching heuristic for Quantified Boolean Formulas, a problem that allows existential and universal quantifiers. Lederman et al. [Led+20] note positive generalisation properties across problem sizes for problems from similar distributions. Our work focuses more on the generalisation and transfer properties of a GNN value-based RL algorithm. We investigate data efficiency properties and merge VSIDS with a trained RL agent, looking into the trade-off between the model use and its effect on the final solution. Apart from that, we show that achieving good performance and generalisation properties is possible with a more straightforward state representation. Finally, doing more message propagations per step and using a GNN as a Q-function (in their case, a GNN only computes node embeddings) allows us to consider more subtle dependencies in the graph.

Look-ahead SAT solvers [HvM09] perform more computations compared to VSIDS to evaluate the consequences of their decisions. This pays off in some cases, e.g. random k-SAT. Difference heuristics used for making a decision measure reduction in the formulae before and after the decision. LRB heuristic [Lia+16] uses multi-armed bandits to explicitly optimise for the ability of the variables' to generate learnt clauses. We hypothesise that Graph-Q-SAT might have learnt some aspects of those heuristics (see Figure 7.4). We believe that integrating Graph-Q-SAT with other types of solvers is a promising direction for future research.

Vinyals et al. [VFJ15] introduce a recurrent architecture for approximately solving complex problems, such as the Traveling Salesman Problem, approaching it in a supervised way. Bello et al. [Bel+17] consider combinatorial optimisation

problems with RL. Khalil et al. [Kha+17] approach combinatorial optimisation using GNNs and DQN, learning a heuristic that is later used greedily. It differs from our approach in that their heuristic is effectively the algorithm itself. The environment dynamics in Khalil et al. [Kha+17] is straightforward, with the following state easily inferred, given the current state and the chosen action. However, in the case of SAT, there are CDCL steps after the decision, and the next state might be totally different from the current one making the problem harder in terms of learning the Q-function. In addition, we use Battaglia et al. [Bat+18] which is more expressive than `structure2vec` used in Khalil et al. [Kha+17]. The global attribute in Battaglia et al. [Bat+18] can facilitate message passing in case of a bigger graph. Having separate updaters for edges and nodes leads to more powerful representations. Finally, an edge updater of Battaglia et al. [Bat+18] can learn better pairwise interaction between the sender and the receiver, enabling sending different messages to different nodes.

Paliwal et al. [Pal+20] use GNNs with imitation learning for theorem proving. Carbune et al. [Car+18] propose a general framework for injecting an RL agent into existing algorithms. Cai et al. [Cai+19] use RL to find a suboptimal solution that is further refined by another optimisation algorithm, in their case, simulated annealing [KJV83]. It is not restricted to simulated annealing only, and this modularity is valuable. However, it is also a drawback because the second optimisation algorithm might benefit more from the first if they were interleaved. For instance, Graph-Q-SAT can guide the search before VSIDS overcomes its initialisation bias.

7.6 Conclusions and Future work

In this chapter, we demonstrated that Q-learning could be used to learn the branching heuristic of a SAT solver. Graph-Q-SAT uses a simple state representation and does not require elaborate reward shaping. We show empirically that Graph-Q-SAT causes more variable propagations per step, solving the SAT problem in fewer iterations than VSIDS. For larger problems, we showed that fewer iterations

could, in turn, reduce wall-clock time. We demonstrated its generalisation abilities, showing more than a 2-3x reduction in iterations for problems up to 5x larger and 1.5-2x from SAT to unSAT. We showed how Graph-Q-SAT improves VSIDS and that it is data-efficient. We also demonstrated positive transfer properties when changing the task family and showed that training on data from other distributions could lead to further performance improvements.

Although we showed the powerful generalisation properties of graph-based RL on SAT, we believe the problem is still far from solved. More work is needed before Graph-Q-SAT is ready to compete with branching heuristics in a modern industrial setting. The two main directions of future applied research are scaling and wall-clock time reduction. Some possible ways of tackling these issues include combining the machine learning improvements from above with an efficient C++ implementation, using a smaller network, reducing the network polling frequency, and replacing the variable activities with Graph-Q-SAT's output, similarly to Selsam et al. [SB19].

From the machine learning perspective, it is intriguing to study how combining benchmarks from different domains might improve the transfer behaviour. Graph-Q-SAT can benefit from using the latest stabilising techniques [Hes+18] and more sophisticated exploration methods. Building an efficient curriculum is another crucial step towards further scaling the method, motivated by Bapst et al. [Bap+19]. Newsham et al. [New+14] show that the graph structure of SAT problems affects the problem complexity. We are interested in understanding how the structure influences the performance of Graph-Q-SAT and how we can exploit this knowledge to improve Graph-Q-SAT. Modern industrial SAT problems are so large in terms of the number of variables and clauses, that they are often hard to fit into memory. We believe that scaling ideas from the GNN community [Jos22], e.g., graph coarsening, are of utmost importance for real-life applications of Graph-Q-SAT.

While being implementation-specific (MiniSAT is a Boolean SAT solver), Graph-Q-SAT's ideas can be extended to other related problems: other SAT extensions, constraint satisfaction problems and integer programming. These problems will have a different graph featurisation, but the solution will have a similar backbone:

keep the underlying solver intact, replace one of the heuristics and train an RL agent on the resulting environment [Car+18]. We refer the reader to Cappart et al. [Cap+21] for an extensive review on combinatorial optimisation and GNNs.

Complex models are rarely useful (unless for those writing their dissertations).

— V.I. Arnold

8 Afterword

This thesis demonstrated that minimalism is an attractive alternative to the ever-growing complexity of multi-task learning. In Chapter 4, we demonstrated that the naive summation of per-task gradients performs on par with specialised multi-task optimisers while being conceptually simple, faster, and easier to implement. In Chapter 5, we showed that freezing parameters of a GNNs, another minimalist approach, allows keeping the performance at the MLP level while preserving desired generalisation and transfer abilities of GNNs. Chapter 6 showed that biasing a GNN by a fixed input graph is not necessarily the optimal way to solve a downstream task. Amorpheus, a Transformer-based method we proposed, does not require a prior graph structure to be known. Moreover, it performs better while preserving the ability to work across incompatible environments. In the final Chapter 7, we proposed Graph-Q-SAT, a simple method that uses GNNs to learn a generalisable branching heuristic of a boolean SAT solver.

In each of the content chapters (§§4-7), we give immediate direction one could explore to further investigate the topic. Below, we provide three broad research avenues that our work can steer.

Graph representation. The majority of this thesis deals with graph-based representations, and GNNs in particular. We believe that graph representation and structured computation problems in GNNs are far from being solved. In GNNs, the computation graph depends on the input, e.g., the direction of the edges affects the gradient computation. As we show in Chapter 6, we do not necessarily know an optimal graph structure for the downstream task, and this dependency can be harmful. Transformers provide a way of circumventing this issue by learning the inductive biases from data. However, transformers suffer a quadratic computational complexity in the number of nodes, and they will fail in the poor feature regime, making the dot products meaningless. We also believe that thinking of graphs as a general abstraction is harmful to further progress. We need to account for the problem specifics defining the graph semantics. As we showed in Chapter 6, learning inductive biases from data can be beneficial (wrong prior is worse than uninformative[1]). However, learning inductive biases from data is not always possible due to computational constraints or limited data. We believe future work will focus on designing problem-specific architectures and data representations that exploit domain knowledge to facilitate learning and reduce computational complexity. For example, designing a suitable data representation and models for trees is of utmost importance for tasks involving programming code: finding and fixing bugs, generating code, and compiler optimisation. We believe that hyperbolic representations have great potential for learning on trees [Yan+22; KNS22]

If multi-task learning is the answer, what is the question? This thesis clearly illustrates that additional complexity is often unnecessary in multi-task learning leading to two alternative hypotheses. According to the first hypothesis, multi-task learning is a variant of a single-task problem; there is no significant difference between the two. The alternative conjecture is that our current benchmarks are not hard enough to show the difficulty of multi-task learning. While

[1] Jaynes [Jay03] uses a beautiful quote by Thomas Jefferson as an epigraph to the 'Ignorance priors and transformation groups' chapter: `Ignorance is preferable to error and he is less remote from the truth who believes nothing than he who believes what is wrong.`

the current thesis, and especially Chapter 4, is primarily aligned with the first hypothesis, we equally believe in the second one and look forward to the next generation of more complex multi-task problems. Further formalising the setting, e.g. exploring different objectives, and putting constraints on model capacity, will help develop the second hypothesis.

Interpretability is another exciting direction for future MTL research. Even if we assume that the MTL is no different from single-task optimisation-wise, algorithmically the two approaches are quite different. On the one hand, MTL adds more moving parts to the learning system, and it might be harder to interpret because of that. On the other hand, there appear new directions to pursue: for example, how exactly are the weights shared? While working on this thesis, we ran some experiments showing that weight sharing is not always happening in MTL, and most of the weights are specialised on one of the tasks instead. However, more work is needed to falsify this hypothesis, and it would be extremely valuable to run controlled experiments for the tasks for which we can quantify the metric of similarity. Providing task relatedeness in a form of a graph [Nas+20] is a principled way to define similarity that is also interesting from the interpretability viewpoint.

Another important direction to pursue is quantifying the task synergies we discussed in Chapter 1. In this work, as common in deep learning and deep RL, we circumvented this issue by comparing the performance of MTL algorithms without attributing performance improvements to exploiting particular synergies. We believe, it is of great practical importance to address this limitation in the future.

Applications. When working on real-life applications, we not only profit from solving real-life problems but understand what our tools are lacking. Multi-task learning and, especially, incompatible multi-task learning open up many possibilities to expand real-life application areas of machine learning and, especially, reinforcement learning. These include using machine learning models as a part of a complex high-fidelity simulator [CBL19], learning heuristics in combinatorial optimisation problems [Cap+21], and merging traditional computer science algorithms [Cor+09]

with machine learning models [VB21; Iba+22].

We want to conclude with a quote by Socrates: 'The only true wisdom is in knowing you know nothing.' We humbly admit that our work provides only a tiny glimpse of what minimalist approaches are capable of. However, we hope that we provided food for thought for future generations of researchers to question the status quo and go further in their endeavour to expand human knowledge.

Appendices

A

Reproducibility

Contents

A.1 In Defense of the Unitary Scalarisation for Deep Multi-Task Learning

A.1.1 Experimental Setting

We now present details concerning the experimental settings from §4.3, including details on the employed open-source software, dataset information, hardware specifications, and hyperparameters.

A.1.1.1 Supervised Learning

All the experiments were run under Ubuntu 18.04 LTS on a single GPU per run (using two 8-GPU machines in total). Timing experiments were all run on Nvidia GeForce GTX 1080 Ti GPUs with an Intel Xeon E5-2650 CPU. The remaining experiments were run on either Nvidia GeForce RTX 2080 Ti GPUs or Nvidia GeForce GTX 1080 Ti GPUs, respectively, using an Intel Xeon Gold 6230 CPU or an Intel Xeon E5-2650 CPU.

MultiMNIST Multi-MNIST, originally introduced by Sabour et al. [SFH17] and as modified by Sener et al. [SK18], is a simple two-task supervised learning benchmark dataset constructed by uniformly sampling MNIST [LeC+98] images and placing one in the top-left corner, the other in the bottom-right corner. Each of the two overlaid images corresponds to a 10-class classification task. Using the above procedure, we generate the Multi-MNIST training set from the first 50000 MNIST training images, the validation set from the last 10000 training images, and the test set from the original MNIST test set. For consistency with the experimental setup of Sener et al. [SK18], we employ a modified encoder-decoder version of the LeNet architecture [LeC+98]. Specifically, the last layer is omitted from the encoder, and two fully-connected layers are employed as task-specific predictive heads. The cross-entropy loss is used for both tasks. All methods are trained for 100 epochs using Adam [KB15] in the stochastic gradient setting, with an initial learning rate of $\eta = 10^{-2}$ (tuned in $\eta \in \{10^{-3}, 10^{-2}, 10^{-1}\}$ and yielding the best

validation results for all considered algorithms), exponentially decayed by 0.95 after each epoch, and a mini-batch size of 256.

CelebA The CelebA [Liu+15] dataset consists of 200,000 headshots (with standard training, validation and test splits) associated with the presence or absence of 40 attributes. In the MTL literature, it is commonly treated as a 40-task classification problem, each task being a binary classification problem for an attribute. As commonly done in previous work [SK18; Yu+20; Liu+21b], we employ an encoder-decoder architecture where the encoder is a ResNet-18 [He+16] (without the final layer), and the per-task decoders are linear classifiers. The cross-entropy loss is used for all tasks. The training is performed from scratch for 50 epochs using Adam, with a mini-batch size of 128 and a per-epoch exponential decay factor of 0.95. As common on this network-dataset combination [LYZ21; Che+20], the initial learning rate is $\eta = 10^{-3}$ for all methods except for MGDA and IMTL, for which $\eta = 5 \times 10^{-4}$ yielded a better validation performance. As done by the respective authors, for PCGrad, RLW and GradDrop, we use the same learning rate as the unitary scalarisation [Yu+20; LYZ21; Che+20].

Cityscapes We rely on the version of the dataset pre-processed by Liu et al. [LJD19], which consists of 2,975 training and 500 test images and presents two tasks: semantic segmentation on 7 classes and depth estimation. We further split the original training set into a validation set of 595 images, employed to tune hyperparameters and a training set of 2380 images. Consistently with recent work [LYZ21], we rely on a dilated ResNet-50 architecture pre-trained on ImageNet [YKF17] for the encoder and on the Atrous Spatial Pyramid Pooling [Che+18a] as decoders. While more powerful encoders might lead to better performance on Cityscapes, like the SegNet [BKC17] used in [JV22; Liu+21a; Nav+22], we aim to provide a fair comparison of MTL optimisers rather than maximise overall task performance. Cross-entropy loss is employed on the semantic segmentation task, whereas the ℓ_1 loss is used for the depth estimation. The training is performed by using Adam

with a mini-batch size of 32 for 100 epochs, with an initial step size $\eta = 5 \times 10^{-4}$ resulting in the best validation performance for all algorithms, exponentially decayed by 0.95 at each epoch.

Cityscapes is a heterogeneous MTL problem: it contains tasks of different types whose validation metrics cannot be averaged to perform model selection. Considering the lack of an established model selection procedure in this context, we potentially evaluate a different model for each metric, chosen as the one with the best validation performance across the epochs. This procedure maximises per-task performance at the cost of increased inference time. If inference time is a priority, an alternative model selection procedure could rely on relative task improvement [JV22; Nav+22; Liu+21a], assuming that per-metric improvements are to be weighted linearly. Nevertheless, any consistently applied model selection scheme serves the primary goal of our work: evaluating all SMTOs on a fairground.

A.1.1.2 Reinforcement Learning

Similarly to the supervised learning experiments, we ran all the experiments under Ubuntu 18.04 LTS using one GPU per run (using six 8-GPU machines in total). Timing experiments were all run using NVIDIA GeForce RTX 2080 Ti GPUs with an Intel Xeon Gold 6230 CPU. The main bulk of the remaining experiments was run on Nvidia GeForce RTX 2080 Ti GPUs with either Intel Xeon Gold 6230 or Intel Xeon Silver 4216. We utilised NVIDIA GeForce RTX 3080 GPUs with Intel Xeon Gold 6230 CPUs for a small fraction of experiments.

We use Meta-World's MT10/MT50 for our experiment. The benchmark consists of ten/fifty tasks in which a simulated robot manipulator has to perform various actions, e.g., pressing a button, opening a door, or pushing the block. We use Sodhani et al. [SZP21] for most of the hyperparameters and list them in Table A.1. We use bold font for hyperparameters different from Sodhani et al. [SZP21]. Similarly to Sodhani et al. [SZP21], we use the v1 version of Metaworld for our experiments[1]. Sodhani et al. [SZP21] use a shared entropy loss weight α for PCGrad and separate

[1] https://github.com/rlworkgroup/metaworld.git@af8417bfc82a3e249b4b02156518d775f29eb289

α for unitary scalarisation[2]. For fairness, we use shared α for all methods in our experiments. Since it is a single number (rather than a vector), we used unitary scalarisation to update α for all SMTOs apart from PCGrad, which was already implemented in [SZP21].

We use the same network architecture as in Sodhani et al. [SZP21], i.e. a three-layered feedforward fully-connected network with 400 hidden units per layer for both the actor and the critic. The actor is shared across all tasks as well as the critic.

To normalize rewards, we keep track of first and second moments in the buffer and normalise the rewards by their standard deviation: $r'_i = {}^{r_i}/_{\hat{\sigma}_i}$, where $\hat{\sigma}_i$ is the sample standard deviation of the rewards for environment i.

Sodhani et al. [SZP21] average the gradient for unitary scalarisation and PCGrad, whereas our SMTO implementations sum the gradients, i.e. effectively using larger learning rates (apart from MGDA that assures that all the aggregation weights sum to 1). We tried reducing the learning rate for SMTOs that sum (RLW Norm., RLW Diri., and GradDrop) both for MT10 and MT50, but it worked worse for these methods, and we kept the default learning rate for them as well. We had to use a lower learning rate for IMTL because the default one crashed at the beginning of training due to numerical overflow. A smaller learning rate did not prevent it from crashing, but this happened much later.

We tried 10^6, 2×10^6, and 4×10^6 for the replay buffer size, with the last being superior in terms of stability. Additionally, for l_2 actor regularisation, we tried 0.0001 and 0.0003, with the latter slightly superior for the baseline. We tried the same options for other SMTOs and picked the best option for each of the method. For MGDA, no regularisation works best, most likely due to a strong regularisation effect of the method itself, which is mirrored by our supervised learning results. PCGrad and Graddrop work best with the regularisation coefficient of 0.0001. Both RLW variants use the same coefficient as the baseline (0.0003).

For MT50, we took the best MT10 hyperparameters, and we believe one could obtain even better results for unitary scalarisation since it is much faster

[2] https://mtrl.readthedocs.io/en/latest/pages/tutorials/baseline.html

Table A.1: Hyperparameters of the RL experiments. Hyperparameters different from Sodhani et al. [SZP21] are in bold.

Hyperparameter	**Value**
All methods	
– training steps	2,000,000
– batch size	1280
– **Replay buffer size**	**4,000,000**
– actor learning rate	0.0003
– critic learning rate	0.0003
– entropy α learning rate	0.0003
– shared entropy α	True
– runs	10
– discounting γ	0.99
Unit. Scal.	
– **actor l_2 coeff.**	**0.0003**
PCGrad	
– **actor l_2 coeff.**	**0.0001**
RLW Norm.	
– normal mean	0
– normal std	1
– actor l_2 coeff.	0.0003
RLW Diri.	
– α	1
– actor l_2 coeff.	0.0003
GradDrop	
– k	1
– p	0.5
– actor l_2 coeff.	0.0001
MGDA	
– gradient normalisation	L_2
– actor l_2 coeff.	0.0
IMTL	
– actor learning rate	0.00003
– critic learning rate	0.00003
– entropy α learning rate	0.00003
– actor l_2 coeff.	0.0

to tune compared to other SMTOs (e.g. 15 hours for unitary scalarisation vs nine days for PCGrad).

A.1.2 Software Acknowledgments and Licenses

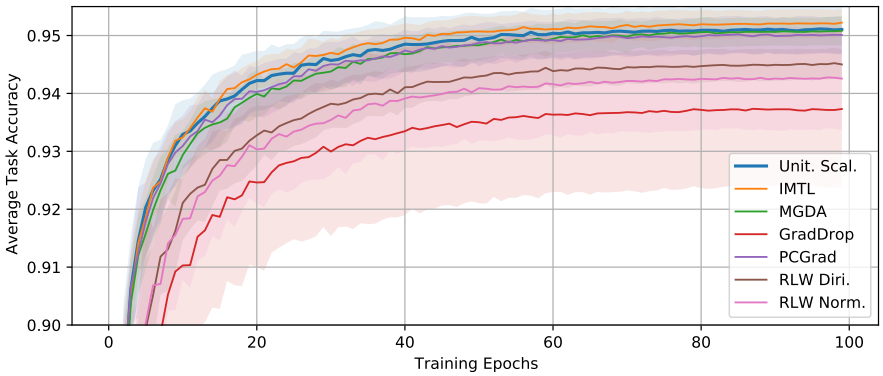
Our codebase is built upon several prior works: [SK18], [LJD19], [LYZ21] and [SZP21]: all of them were released under the MIT license. We also acknowledge Tseng [Tse20], upon which we built some of our code. Multi-MNIST is based on the MNIST dataset released under the Creative Commons Attribution-Share Alike 3.0 license. The code for generating the Multi-MNIST dataset was taken from Sener et al. [SK18] released under the MIT license. CelebA dataset has a custom license allowing non-commercial research purposes. More details can be found on the project website: `http://mmlab.ie.cuhk.edu.hk/projects/CelebA.html`. Cityscapes also has a custom license allowing non-commercial research purposes. The full text of the license can be found on the project website: `https://www.cityscapes-dataset.com/license/`. Meta-world used for RL experiments is released under the MIT license.

A.1.3 Supplementary Supervised Learning Experiments

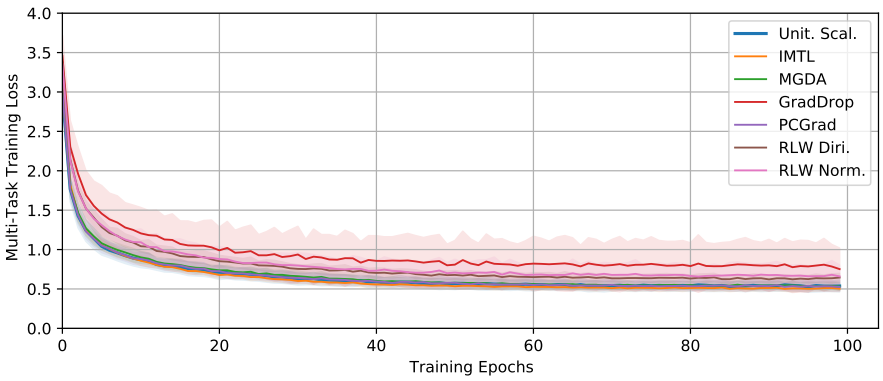
This section presents supervised learning results omitted from §4.3.1. In particular, we show additional plots for the experiments of §4.3.1, then present an analysis of the regularising effect of SMTOs in the absence of single-task regularisation (§A.1.3.2), and conclude with an ablation study on GradDrop's dependency on the sign of per-task gradients (§A.1.4.1).

A.1.3.1 Addendum

This section complements the plots presented in §4.3.1. In particular, we show the test and runtime results in table form, along with the behaviour of the validation metrics and the training loss over the training epochs. Plots for Multi-MNIST, CelebA, and Cityscapes are reported in Figures A.1, A.2 and A.9, respectively. The



(a) Mean (and 95% CI) average task validation accuracy per training epoch.



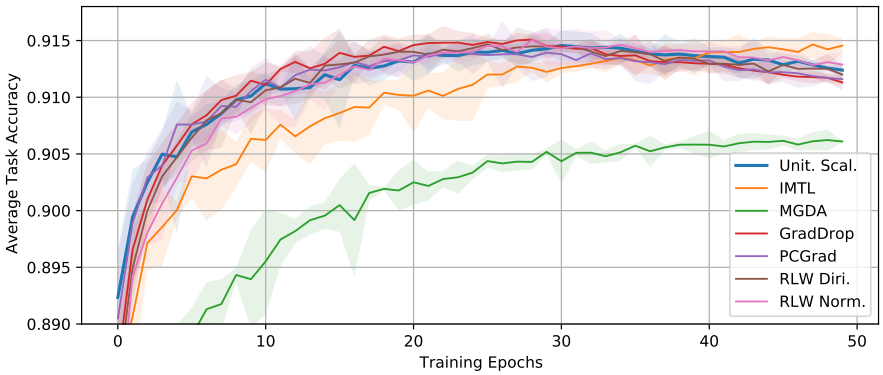
(b) Mean (and 95% CI) training multi-task loss $\mathcal{L}^{\text{MT}}$ per epoch.

MTO	Average Task Accuracy	Epoch Runtime [s]
Unit. Scal.	9.476e-01 ± 4.368e-03	[3.510e+00, 3.617e+00]
IMTL	9.487e-01 ± 2.533e-03	[3.695e+00, 3.996e+00]
MGDA	9.478e-01 ± 1.977e-03	[3.491e+00, 3.617e+00]
GradDrop	9.347e-01 ± 1.282e-02	[3.508e+00, 3.589e+00]
PCGrad	9.479e-01 ± 3.578e-03	[3.807e+00, 3.928e+00]
RLW Diri.	9.430e-01 ± 2.973e-03	[3.790e+00, 4.005e+00]
RLW Norm.	9.399e-01 ± 8.929e-03	[3.894e+00, 4.225e+00]

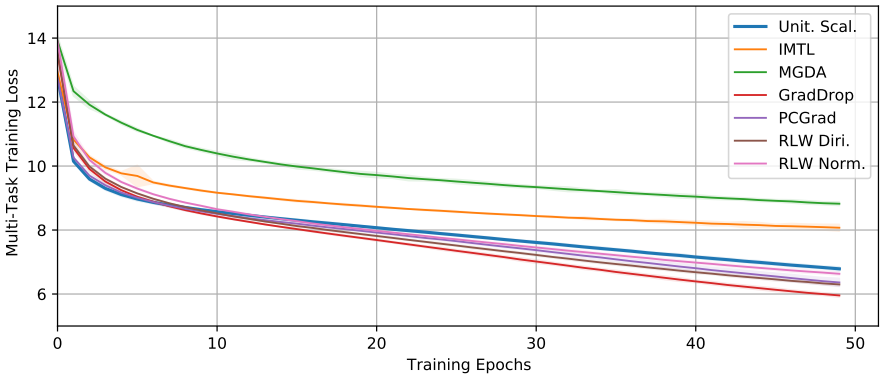
(c) Mean and 95% CI of the avg. task test accuracy across runs and interquartile range for the training time per epoch.

Figure A.1: Additional figures for the comparison of various SMTOs with the unitary scalarisation on the MultiMNIST dataset [SK18].

behavior of the CelebA training loss demonstrates heavier regularisation (compare with the unregularized plot in Figure A.3a). Except IMTL and MGDA, for which the tuned values of the weight decay prevent overfitting, the other optimisers display



(a) Mean (and 95% CI) average task validation accuracy per training epoch.



(b) Mean (and 95% CI) training multi-task loss $\mathcal{L}^{\text{MT}}$ per epoch.

MTO	Average Task Accuracy	Epoch Runtime [s]
Unit. Scal.	9.090e-01 ± 7.568e-04	[2.869e+02, 2.878e+02]
IMTL	9.093e-01 ± 7.631e-04	[3.600e+02, 3.621e+02]
MGDA	9.022e-01 ± 9.687e-04	[6.859e+02, 7.194e+02]
GradDrop	9.098e-01 ± 3.383e-04	[3.001e+02, 3.008e+02]
PCGrad	9.093e-01 ± 1.108e-03	[1.015e+04, 1.016e+04]
RLW Diri.	9.099e-01 ± 7.845e-04	[3.040e+02, 3.054e+02]
RLW Norm.	9.095e-01 ± 1.012e-03	[3.028e+02, 3.037e+02]

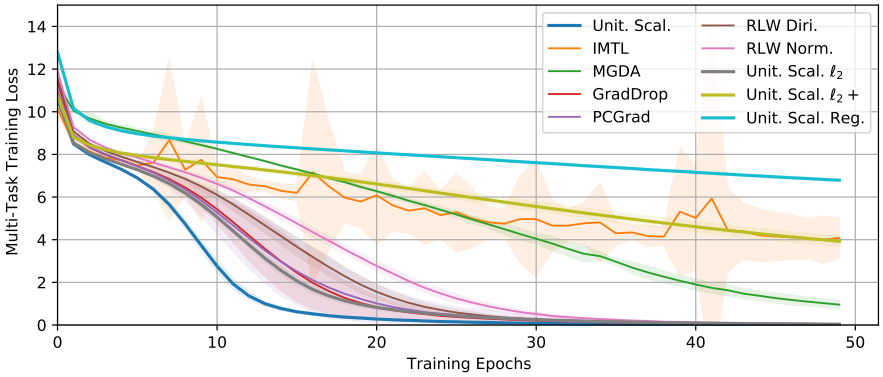
(c) Mean and 95% CI of the avg. task test accuracy across runs and interquartile range for the training time per epoch.

Figure A.2: Additional figures for the comparison of various SMTOs with the unitary scalarisation on the CelebA [Liu+15] dataset.

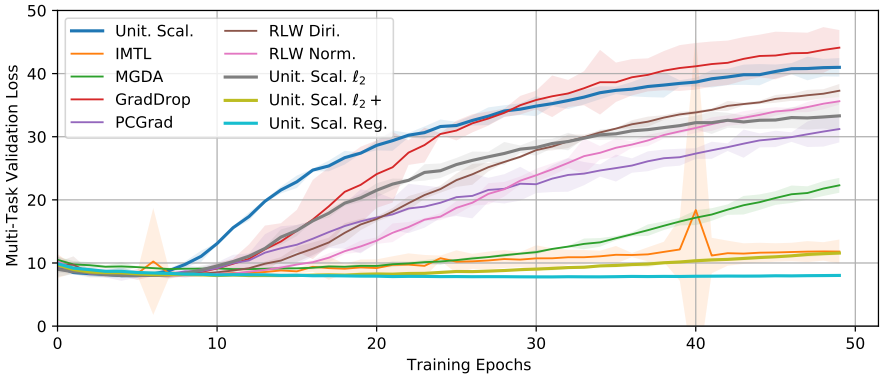
very similar validation and training curves, and start overfitting around epoch 30. Considering that most SMTOs required less regularisation (see §4.3.1.2), the results are consistent with our interpretation of SMTOs as regularisers in §4.4. The Cityscapes plots display a certain instability across training epochs, as demonstrated

by the various peaks and valleys in the metrics. Nevertheless, despite a factor 10 difference in scale, both training losses are similarly decreased by most optimisers.

A.1.3.2 Unregularised Experiments



(a) Mean and 95% CI (3 runs) multi-task training loss per epoch.



(b) Mean and 95% CI (3 runs) multi-task validation loss per training epoch.

Figure A.3: Additional figures for the unregularised comparison of various SMTOs with the unitary scalarisation on CelebA. SMTOs provide varying degrees of regularisation.

Figures 4.7, A.3a and A.3b respectively report the average task validation accuracy, the multi-task training loss, and the multi-task validation loss at each training epoch. The regularising effect of SMTOs compared to unitary scalarisation is shown by: (i) the delay of the onset of overfitting on the validation data in Figure 4.7, (ii) the reduction of the convergence rate on the training loss in Figure A.3a (compare with Figure A.2b), and (iii) the fact that validation and training losses remain positively correlated for larger numbers of epochs. In fact, the behaviour of both the training and validation loss for the SMTOs closely parallels

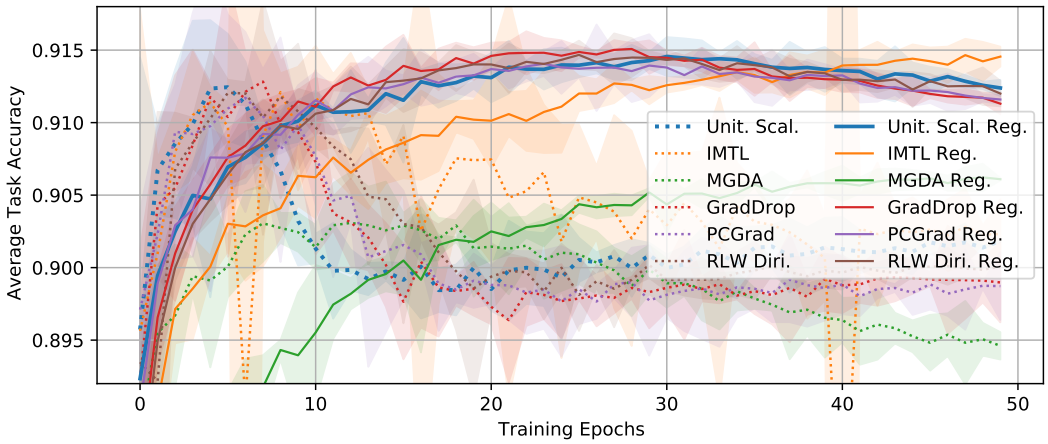


Figure A.4: Effect of regularisation (Dropout layers and weight decay) on the average task validation accuracy for all considered optimisers on the CelebA dataset: regularisation improves the average performance of all algorithms.

that of ℓ_2-regularised unitary scalarisation, with differing degrees of regularisation. We further note that unregularised IMTL displays a certain instability (compare with the regularised version in Figure A.2a). Adding Dropout layers further reduces

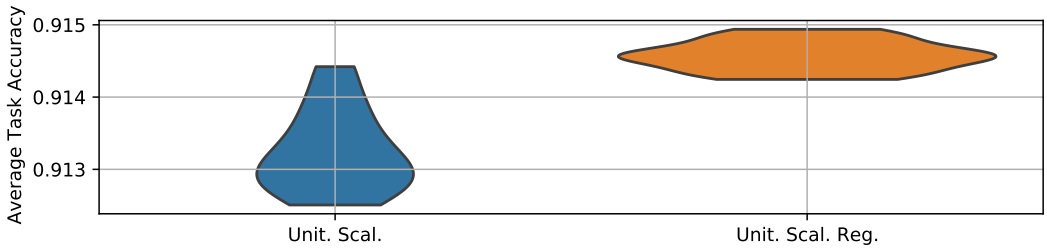


Figure A.5: Effect of regularisation (Dropout layers and weight decay) on unitary scalarisation on the CelebA dataset: violin plots (20 runs) for the best avg. task validation accuracy over epochs. The width at a given value represents the proportion of runs yielding that result. Regularisation improves the average performance while decreasing its variability.

overfitting, improves stability (reduced confidence intervals) and pushes the average validation curve upwards, motivating its use on all optimisers for the experiments of §4.3.1.2. Nevertheless, confidence intervals in Figure 4.7 still overlap due to the instability of the unregularised unitary scalarisation. Figure A.5 provides a more detailed comparison over 20 repetitions, confirming that the combined use of Dropout layers and ℓ_2 regularisation improves average performance and reduces the

empirical variance for unitary scalarisation. Furthermore, Figure A.4 shows that regularisation improves the peak average validation performance for all algorithms, demonstrating the need for tuning λ also for SMTOs. We conclude by pointing out that even without regularisation when carefully tuned, the maximal performance over epochs of unitary scalarisation is comparable to SMTOs in Figure 4.7.

A.1.4 Multi-Task Optimisers and Under-Optimisation

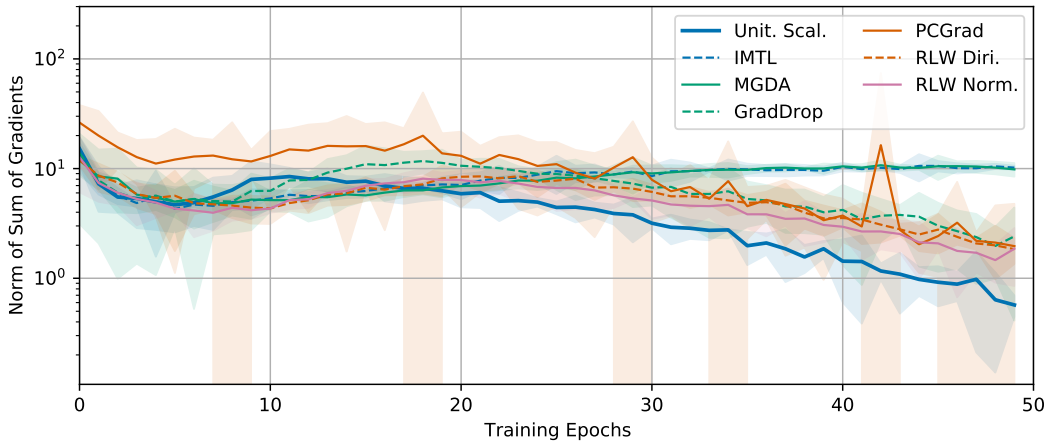


Figure A.6: Mean and 95% CI (three runs) per-epoch average of the ℓ_2 norm of the sum of per-task gradients of the mini-batch loss, computed every 100 updates. The experimental setup mirrors the CelebA experiments in §4.4.1 and appendix A.1.3.2. SMTOs converge away from stationary points of unitary scalarisation, empirically supporting our analysis from §4.4.2.

The experiments in §4.4.1 and appendix A.1.3.2 show that SMTOs empirically act as regularisers for Equation 2.1. The analysis from §4.4.2 provides a partial explanation for the results: due to their larger convergence sets, MGDA, IMTL and PCGrad might under-optimise Equation 2.1 compared to unitary scalarisation. In order to empirically evaluate our analysis, we estimate $\left\|\sum_{i\in\mathcal{T}} \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i\right\|_2$, the norm of the unitary scalarisation update on shared parameters $\boldsymbol{\theta}_{\|}$, for all optimisers throughout the unregularised CelebA experiment (§4.4.1). Large magnitudes for $\left\|\sum_{i\in\mathcal{T}} \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i\right\|_2$ towards convergence would indicate that SMTOs steer optimisation far from stationary points of unitary scalarisation, resulting in under-optimisation.

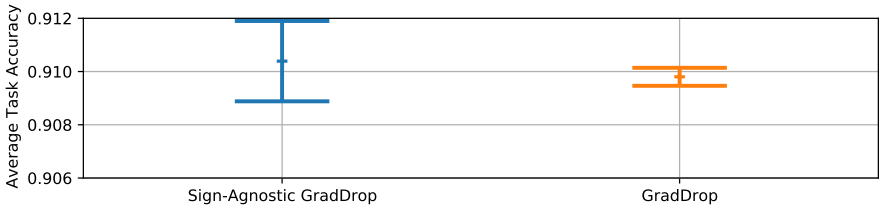
For Figure A.6, we compute the update norm on the mini-batch loss every 100 updates, and report the per-epoch average. Compared with unitary scalarisation, most SMTOs have smaller or comparable update magnitude in the first 15 epochs. However, towards convergence, SMTOs display larger $\left\|\sum_{i\in\mathcal{T}} \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i\right\|_2$ compared to unitary scalarisation. In particular, IMTL and MGDA, both of which were shown to have a larger convergence set in §4.4.2, display the largest norm. The additional stochasticity of RLW and GradDrop also appears to lead to larger norm values than unitary scalarisation, yet to a lesser degree. Given that SMTOs incur a larger loss than unitary scalarisation (see Figure A.3a) in the later epochs, we can conclude that SMTOs guide optimisation towards regions of the parameter space that under-optimise the loss, providing empirical support for our analysis.

A.1.4.1 Sign-Agnostic GradDrop

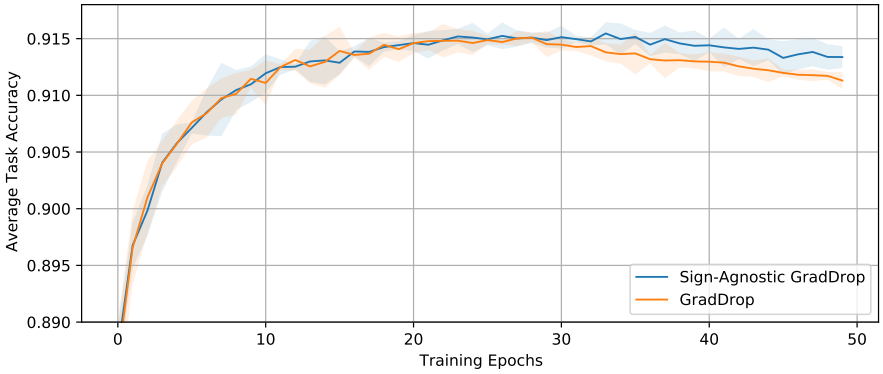
We will now present an ablation study on GradDrop, investigating the effect of the sign of per-task gradients on the SMTO's performance. Specifically, we compare the performance of GradDrop with a sign-agnostic version of its stochastic gradient masking (which we refer to as "Sign-Agnostic GradDrop"), whose update direction is defined as follows:

$$\mathbf{g} = -\left(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_{\|}}\right)^T \left(\sum_{i\in\mathcal{T}} \mathbf{u}_i \odot \nabla_{\mathbf{z}} \mathcal{L}_i\right),$$

where $\mathbf{u}_i, \nabla_{\mathbf{z}}\mathcal{L}_i \in \mathbb{R}^{n\times r}$ and, for all $i \in \mathcal{T}$, $\mathbf{u}_i$ is i.i.d. according to $\mathbf{u}_i[j,k] \sim \text{Bernoulli}(p)\ \forall j \in \{1,\dots,n\}, k \in \{1,\dots,r\}$. Differently from a similar study carried out by Chen et al. [Che+20], we tuned the hyperparameter of the sign-agnostic masking in the following range: $p \in \{0.1, 0.25, 0.5, 0.75, 0.9\}$. The experimental setup complies with the one described in Appendix A.1.1.1. Figure A.7, plotting test and validation results for the CelebA dataset [Liu+15], shows that the performance of Sign-Agnostic GradDrop closely matches the original algorithm. Therefore, sign conflicts across per-task gradients do not seem to play a significant role in GradDrop's performance.

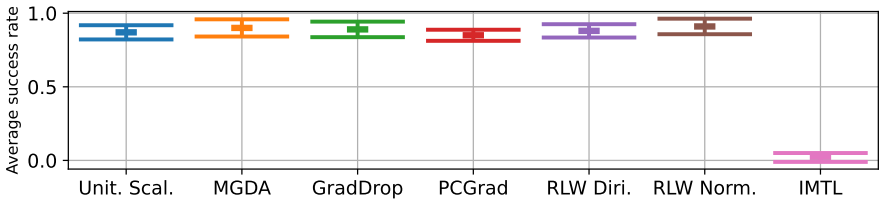


(a) Mean and 95% CI (3 runs) avg. task test accuracy.

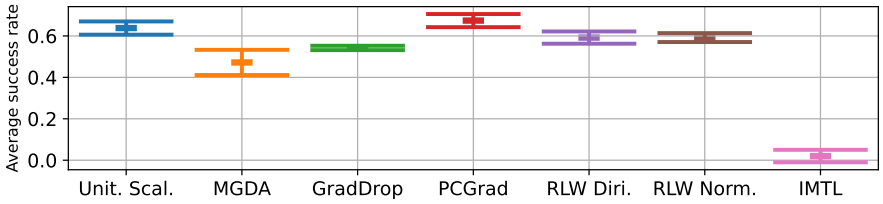


(b) Mean and 95% CI (3 runs) avg. task validation accuracy per training epoch.

Figure A.7: Comparison of GradDrop [Che+20] with sign-agnostic masking of the shared-representation gradients on the CelebA dataset [Liu+15]. No statistically relevant difference between the two methods can be observed for the majority of the epochs.



(a) MT10.

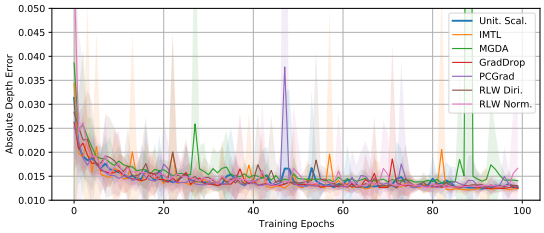


(b) MT50.

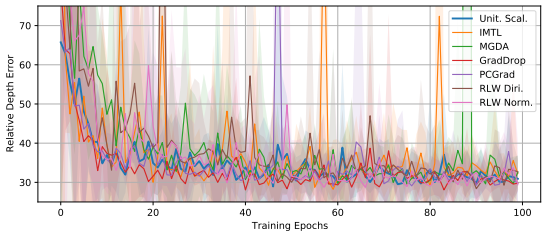
Figure A.8: Mean and 95% CI (10 runs) for the best avg. success rate on Meta-world. None of the SMTOs significantly outperforms unitary scalarisation.

(a) Mean and 95% CI of the test metrics across runs and interquartile range for the training time per epoch.

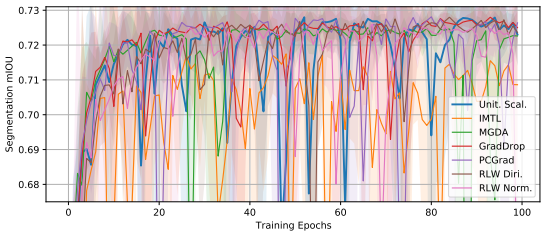
MTO	Absolute Depth Error	Relative Depth Error	Segmentation Accuracy	Segmentation mIOU	Epoch Runtime [s]
Unit. Scal.	1.301e-02 ± 2.342e-04	4.761e+01 ± 5.148e+00	9.196e-01 ± 2.913e-04	7.012e-01 ± 6.001e-04	[3.228e+02, 3.241e+02]
IMTL	1.281e-02 ± 7.521e-04	4.389e+01 ± 6.984e-01	9.164e-01 ± 2.828e-03	6.967e-01 ± 4.785e-03	[7.329e+02, 7.373e+02]
MGDA	1.418e-02 ± 2.331e-04	4.750e+01 ± 1.466e+01	9.189e-01 ± 2.636e-04	6.999e-01 ± 3.124e-03	[7.251e+02, 7.269e+02]
GradDrop	1.293e-02 ± 2.757e-04	4.674e+01 ± 7.709e+00	9.193e-01 ± 1.282e-03	7.024e-01 ± 3.628e-03	[5.196e+02, 5.215e+02]
PCGrad	1.294e-02 ± 2.284e-04	4.380e+01 ± 5.165e+00	9.198e-01 ± 9.119e-04	7.025e-01 ± 6.531e-04	[4.202e+02, 4.212e+02]
RLW Diri.	1.305e-02 ± 4.155e-04	4.810e+01 ± 2.259e+00	9.199e-01 ± 1.247e-03	7.037e-01 ± 1.989e-03	[3.161e+02, 3.164e+02]
RLW Norm.	1.301e-02 ± 5.528e-04	4.630e+01 ± 2.751e+00	9.192e-01 ± 4.962e-04	7.006e-01 ± 4.580e-03	[3.194e+02, 3.210e+02]



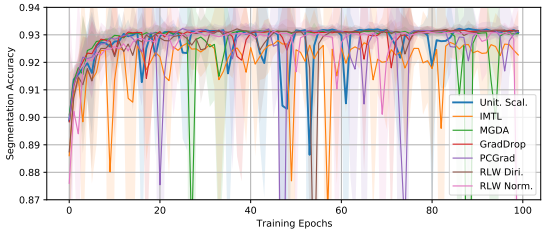
(b) Mean (and 95% CI) absolute depth validation error per training epoch.



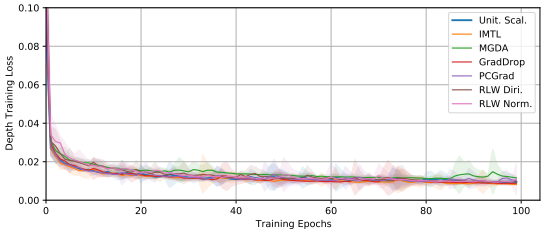
(c) Mean (and 95% CI) relative depth validation error per training epoch.



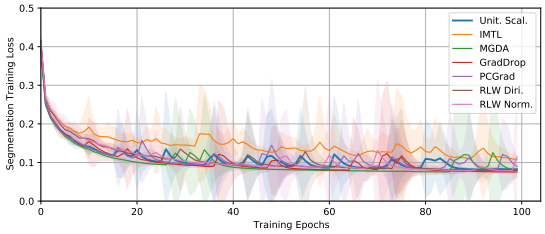
(d) Mean (and 95% CI) validation segmentation mIOU per training epoch.



(e) Mean (and 95% CI) validation segmentation accuracy per training epoch.



(f) Mean (and 95% CI) training depth loss per epoch.



(g) Mean (and 95% CI) training segmentation loss per epoch.

Figure A.9: Additional figures for the comparison of SMTOs with the unitary scalarisation on the Cityscapes [Cor+16] dataset.

A.1.5 Supplementary Reinforcement Learning Experiments

A.1.5.1 Addendum

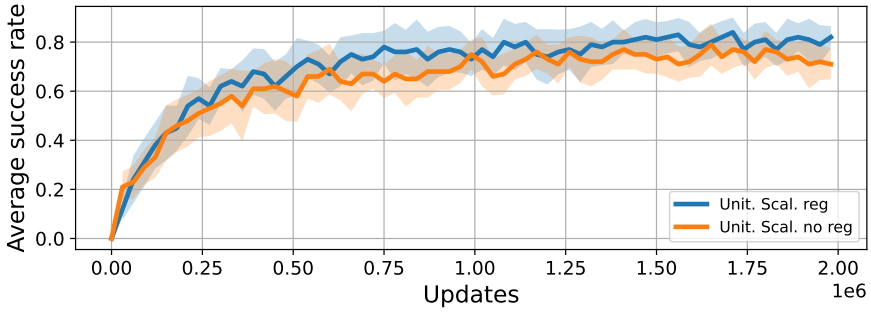
This section presents additional plots for the RL experiments in §4.3.2. Specifically, Figure A.8 re-plots Figure 4.5a and 4.5b with the omitted IMTL results. As

pointed out in §4.3.2, none of the IMTL runs successfully terminated due to numerical instability. Indeed, Liu et al. [Liu+21b] show that, in supervised settings, coefficients do not fluctuate much across epochs [Liu+21b, Figure 4, Appendix B] and never become negative. By contrast, up to 50% of the scaling coefficients, α, are negative in our experiments, thus reversing subtask gradient directions. MGDA, which constrains the weights, is more stable and is comparable to unitary scalarisation. In order to avoid incomplete curves and unfair calculations of the mean, Figure 4.6 plots the highest value ever achieved by *any* seed as a dashed horizontal line. The IMTL results in Figure A.8 instead report each seed's best average success rate until its termination.

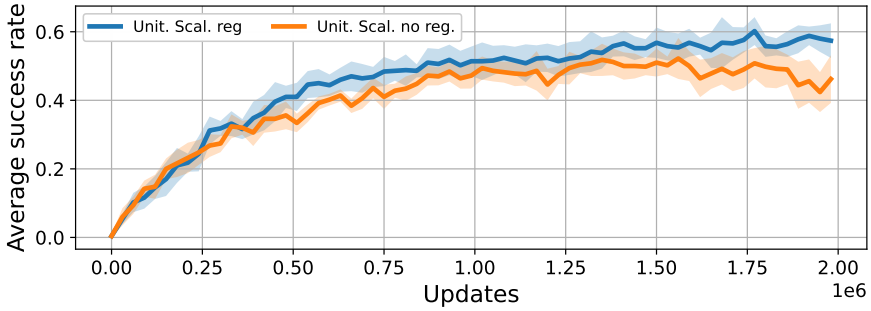
A.1.5.2 Ablation studies

Figure A.11 presents our ablations for MT10 experiments. Due to computational constraints, we ran ablations on the unitary scalarisation and PCGrad since these are the two methods previously tested in the RL setting.

Figure A.10 shows ablation studies on the effect of regularisation on MT10 and MT50. Despite CI overlaps, actor l_2 regularisation pushes the average higher on both benchmarks, motivating our use of regularisation for the experiments in §4.3.2. Furthermore, the gap between the averages tends to widen with the number of updates on MT50, suggesting improved stabilisation.



(a) MT10 average performance (10 runs) and 95% CI.



(b) MT50 average performance (5 runs) and 95% CI.

Figure A.10: For both MT10 and MT50, actor l_2 regularisation pushes the average higher for unitary scalarisation.

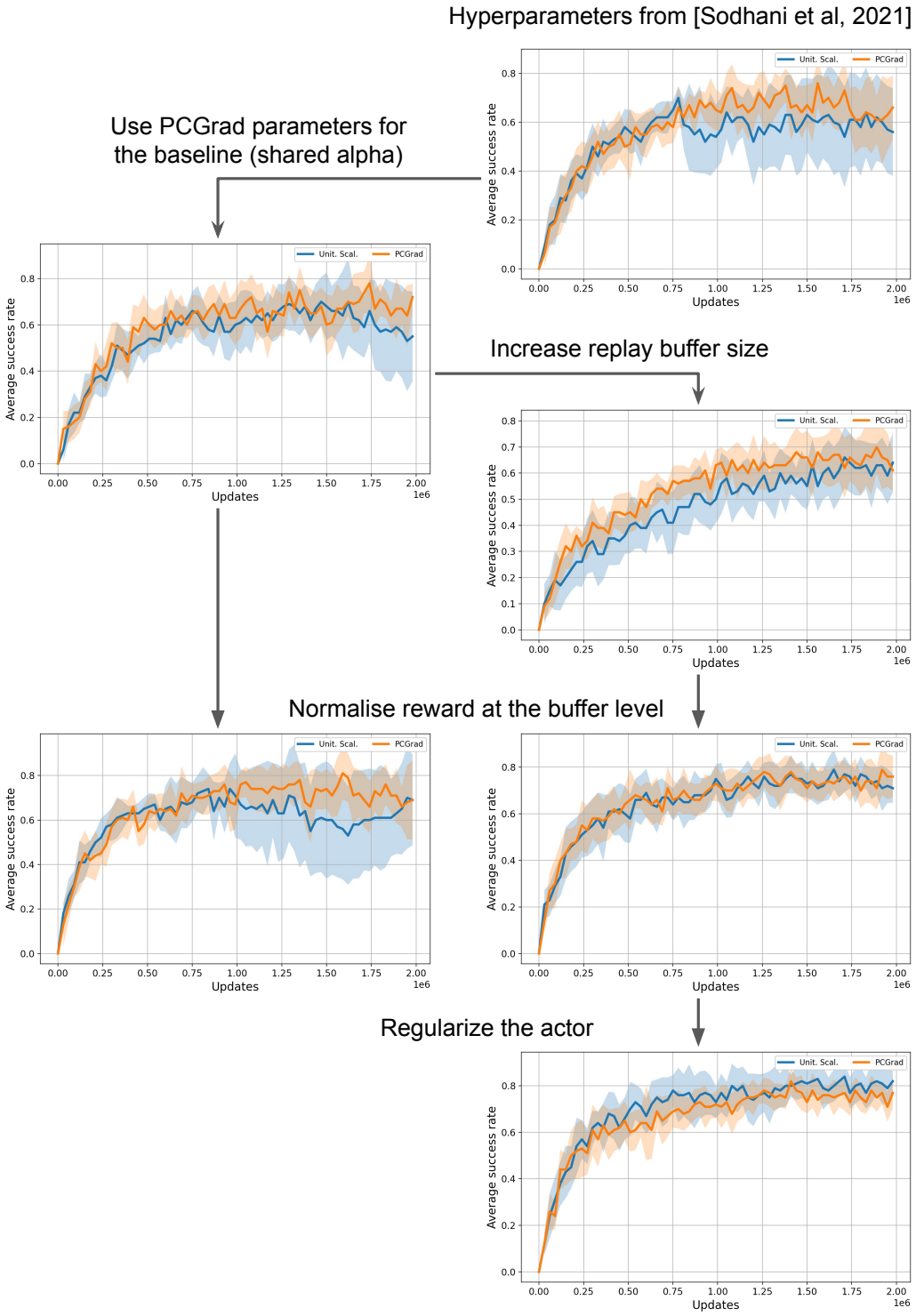


Figure A.11: Meta-world's MT10 ablation experiments.

A.2 Scaling GNNs to High-Dimensional Continuous Control

Here, we outline further details of our experimental approach to supplement those given in §5.5.

A.2.0.1 Data Generation

As typical when training PPO in simulated environments, we train a policy by interleaving two processes. First, we perform repeated rollouts of the current policy in the environment to generate on-policy training data. Second, we optimise the policy with respect to the training data collected to generate a new policy, and repeat.

To improve wall-clock training time, for larger agents, we perform rollouts in parallel over multiple CPU threads, scaling from a single thread for `Centipede-6` to five threads for `Centipede-20`. Rollouts terminate once the sum of timesteps experienced across all threads reaches the training batch size.

For optimisation, we shuffle the training data randomly and split the batch into eight mini-batches. We perform ten optimisation epochs over these mini-batches, in the manner defined by the PPO algorithm [Sch+17] (see §2.2.1.2).

Each experiment is performed six times, and results are averaged across runs. Figure 5.5 is an exception, where results are an average of three runs.

A.2.1 Hyperparameter Search

Our starting point for selecting hyperparameters is the hyperparameter search performed by Wang et al. [Wan+18], whose codebase ours is derived from.

To ensure that we have the best set of hyperparameters for training on large agents, we ran our own hyperparameter search on `Centipede-20` for SF, as seen in Table A.2.

Hyperparameter	Values
Batch size	512, 1024, **2048**, 4096
Learning rate	1e-4, **3e-4**, 1e-5
Learning rate scheduler	adaptive, **constant**
ϵ clipping	0.02, 0.05, **0.1**, 0.2
GNN layers	2, **4**, 10
Gated Recurrent Unit hidden state size	**64**, 128
Learned action std	shared, **separate**

Table A.2: Hyperparameter search for SF on `Centipede-20`. Values in bold resulted in the best performance.

Across the range of agents tested, we conducted a secondary search over the batch size, learning rate and ϵ clipping value for each model. For the latter two hyperparameters, we found that the values in Table A.2 did not require adjusting.

For the batch size, we used the lowest value possible until training deteriorated. Using NerveNet, a batch size of 2048 was required throughout, whereas using SF a batch size of 1024 was best for `Walker`, 512 was best for `Centipede-8` and `Centipede-6`, and 2048 for all other agents.

Wang et al. [Wan+18] provide experimental results for the NerveNet model, which we use as a baseline for our experiments. Out of the `Centipede-n` models, they provide direct training results for `Centipede-8` (see the non-pre-trained agents in their Figure 5). Our performance results are comparable but taken over many more timesteps. Their final MLP results appear slightly different to ours at the same point (they attain roughly 500 more reward), likely due to hyperparameter tuning for performance over a different time frame.

They also provide performance metrics for trained `Centipede-4` and `Centipede-6` agents across the models compared (their Table 1). The results reported there are significantly less than the best performance we attain for both MLP and NerveNet on `Centipede-6`. We suspect this discrepancy is due to running for fewer timesteps in their case, but precise stopping criteria are not provided.

A.2.2 Computing Infrastructure

Our experiments were run on four different machines during the project, depending on availability. These machines use variants of the Intel Xeon E5 processor (models 2630, 2699 and 2680), containing between 44 and 88 CPU cores. As running the agent in the MuJoCo environment is CPU-intensive, we observed little decrease in training time when using a GPU; hence the experiments reported here are only run on CPUs.

Runtimes for our results vary significantly depending on the number of threads allocated and batch size used. Our standard runtime for `Centipede-6` (single thread) for ten million timesteps is around 24 hours, scaling up to 48 hours for our standard `Centipede-20` configuration (five threads). Our experiments on the default MuJoCo agents also take approximately 24 hours for a single thread.

A.2.3 Sources

We release source code to replicate the experiments on GitHub: https://github.com/thecharlieblake/snowflake. Our code is an extension of the NerveNet codebase: https://github.com/WilsonWangTHU/NerveNet. This repository contains the original code/schema defining the `Centipede-n` agents.

The other standard agents are taken from the Gym [Bro+16]: https://github.com/openai/gym. The specific hopper, walker and humanoid versions used are `Hopper-v2`, `Walker2d-v2` and `Humanoid-v2`.

For our MLP results on the Gym agents, as state-of-the-art performance baselines have been well established in this case, we use the OpenAI Baselines codebase (https://github.com/openai/baselines) to generate results, to ensure the most rigorous and fair comparison possible.

The MuJoCo [TET12] simulator can be found at: http://www.mujoco.org/. At the time this work was done, a paid license was required to use MuJoCo, the use of free alternatives was not viable in our case as our key benchmarks are all defined for MuJoCo. In 2021, DeepMind made MuJoCo freely available.

A.3 My Body is a Cage: the Role of Morphology in Graph-Based Incompatible Control

We initially took the Transformer implementation from the Official Pytorch Tutorial [Seq] which uses `TransformerEncoderLayer` from Pytorch [Pas+19]. We modified it for the regression task instead of classification and removed the causal masking and the positional encoding. Table A.3 provides all the hyperparameters needed to replicate our experiments.

Table A.3: Hyperparameters of our experiments

Hyperparameter	**Value**	**Comment**
Amorpheus		
– Learning rate	0.0001	
– Gradient clipping	0.1	
– Normalisation	LayerNorm	As an argument to `TransformerEncoder` in `torch.nn`
– Attention layers	3	
– Attention heads	2	
– Attention hidden size	256	
– Encoder output size	128	
Training		
– runs	3	per benchmark

Amorpheus uses gradient clipping and a lower learning rate. We found that SMP also performs better with the decreased learning rate (0.0001), and we use it throughout the work. Figure A.12 demonstrates the effect of a lower learning rate on `Walker++`. All other SMP hyperparameters are as reported in the original paper with the two-directional message passing.

Wang et al. [Wan+18] add an artificial return limit of 3800 for their Walkers environment. We remove this limit and compare the methods without it. For NerveNet, we plot the results with the option best for it. Figure A.13 compares the two options.

Table A.4: Full list of environments used in Chapter 6.

Environment	Training	Zero-shot testing
`Walker++`		
	`walker_2_main`	`walker_3_main`
	`walker_4_main`	`walker_6_main`
	`walker_5_main`	
	`walker_7_main`	
`humanoid++`		
	`humanoid_2d_7_left_arm`	`humanoid_2d_7_left_leg`
	`humanoid_2d_7_lower_arms`	`humanoid_2d_8_right_knee`
	`humanoid_2d_7_right_arm`	
	`humanoid_2d_7_right_leg`	
	`humanoid_2d_8_left_knee`	
	`humanoid_2d_9_full`	
`Cheetah++`		
	`cheetah_2_back`	`cheetah_3_balanced`
	`cheetah_2_front`	`cheetah_5_back`
	`cheetah_3_back`	`cheetah_6_front`
	`cheetah_3_front`	
	`cheetah_4_allback`	
	`cheetah_4_allfront`	
	`cheetah_4_back`	
	`cheetah_4_front`	
	`cheetah_5_balanced`	
	`cheetah_5_front`	
	`cheetah_6_back`	
	`cheetah_7_full`	
`Cheetah-Walker-Humanoid`		
	All in the column above	All in the column above
`Hopper++`		
	`hopper_3`	
	`hopper_4`	
	`hopper_5`	
`Cheetah-Walker-Humanoid-Hopper`		
	All in the column above	All in the column above
`Walkers` from Wang et al. [Wan+18]		
	`Ostrich`	
	`HalfCheetah`	
	`FullCheetah`	
	`Hopper`	
	`HalfHumanoid`	

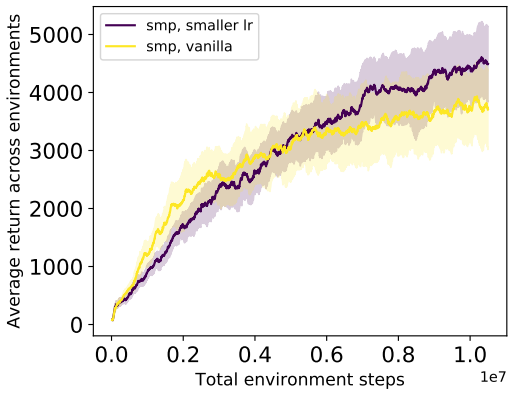


Figure A.12: Smaller learning rate yields better results for SMP on `Walker++`.

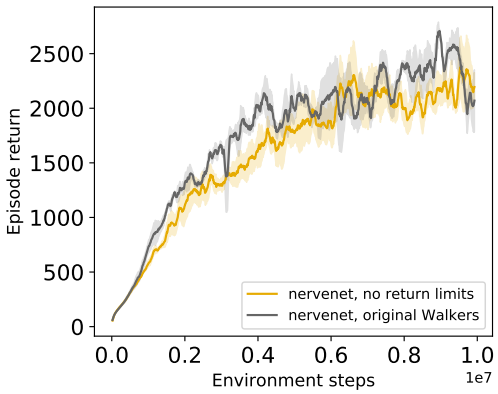
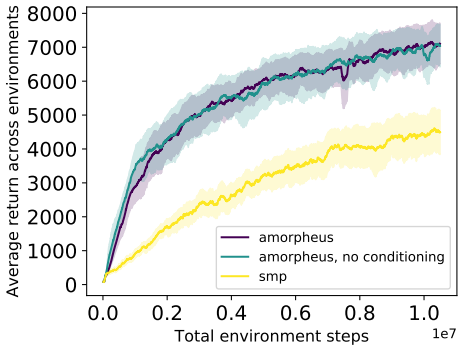


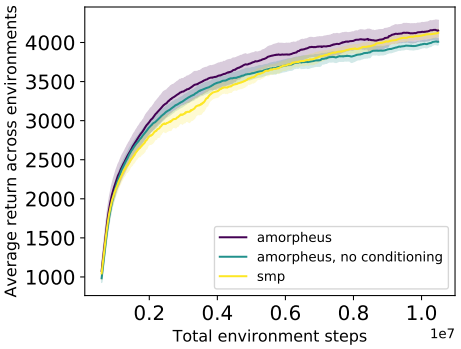
Figure A.13: Removing the return limit slightly deteriorates the performance of NerveNet on Walkers.

A.3.1 Residual Connection Ablation

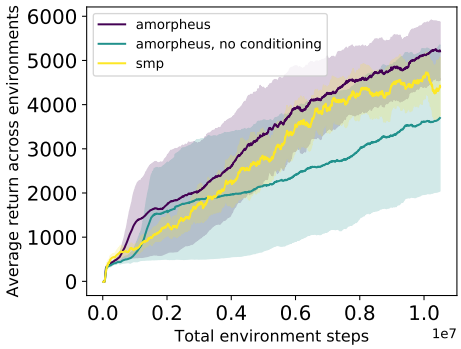
We use the residual connection in Amorpheus as a safety mechanism to prevent nodes from forgetting their own observations. We performed the ablation to check that Amorpheus's improvements do not come from the residual connection alone. As one can see in Figure A.14, we cannot attribute the success of our method to this improvement alone. High variance on `Humanoid++` is related to the fact that one seed started to improve much later, and the average performance suffered as a result.



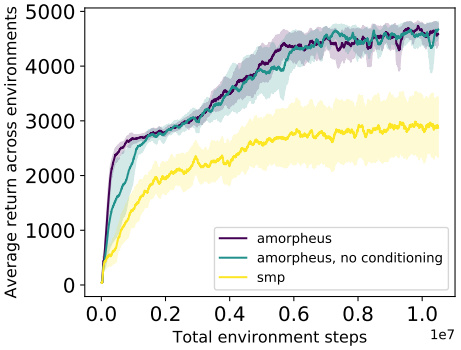
(a) Walker++



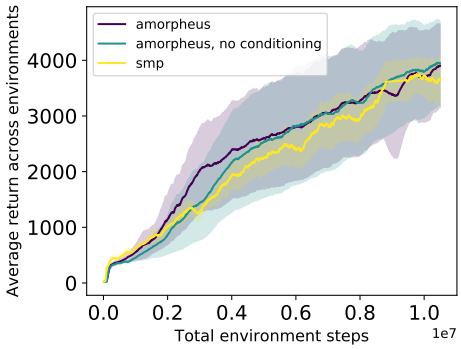
(b) Cheetah++



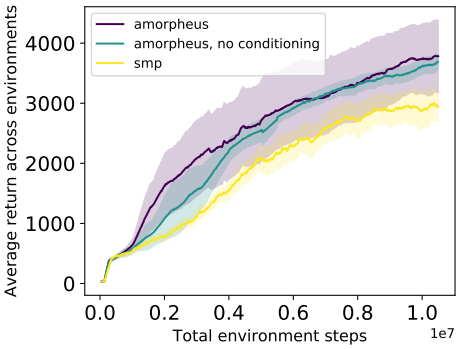
(c) Humanoid++



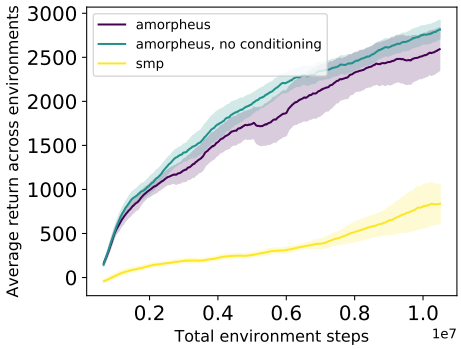
(d) Hopper++



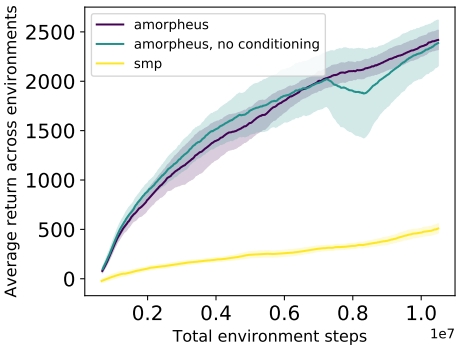
(e) Walker-Humanoid++



(f) Walker-Hum-Hopper++



(g) Cheetah-Walker-Hum++



(h) Cheetah-Walker-Hum-Hopper++

Figure A.14: Residual connection ablation experiment.

A.4 Generalisable Branching Heuristic for a SAT Solver

We implement our models using Pytorch [Pas+19] and Pytorch Geometric [FL19].

Table A.5 contains all the hyperparameters necessary to replicate our results.

Table A.5: Graph-Q-SAT hyperparameters.

Hyperparameter	**Value**	**Comment**
DQN		
– Batch updates	50 000	
– Learning rate	0.00002	
– Batch size	64	
– Memory replay size	20 000	
– Initial exploration ϵ	1.0	
– Final exploration ϵ	0.01	
– Exploration decay	30 000	Environment steps.
– Initial exploration steps	5000	Environment steps, filling the buffer, no training.
– Discounting γ	0.99	
– Update frequency	4	Every 4th environment step.
– Target update frequency	10	
– Max decisions allowed for training	500	Used a safety against being stuck at the episode.
– Max decisions allowed for testing	500	Varied among [0, 10, 50, 100, 300, 500, 1000] for the experiment on Figure 7.5.
– Step penalty size p	-0.1	
Optimisation		
– Optimiser	Adam	
– Adam betas	0.9, 0.999	Pytorch default.
– Adam eps	1e-08	Pytorch default.
– Gradient clipping	1.0	0.1 for training on the graph colouring dataset.
– Gradient clipping norm	L_2	
– Evaluation frequency	1000	
Graph Network		
– Message passing iterations	4	
– Number of hidden layers for GNN core	1	
– Number of units in GNN core	64	
– Encoder output dimensions	32	For vertex, edge and global updater.
– Core output dimensions	64,64,32	For vertex, edge and global respectively.
– Decoder output dimensions	32	For vertex updater, since only Q values are used, no need for edge/global updater.
– Activation function	ReLU	For everything but the output transformation.
– Edge to vertex aggregator $\rho_{e\to v}$	sum	
– Variable to global aggregator $\rho_{v\to u}$	average	
– Edge to global aggregator $\rho_{e\to u}$	average	
– Normalisation	Layer Normalisation	After each GNN updater

A.4.1 Dataset

We split SAT-50-218 into three subsets: 800 training problems, 100 validation and 100 test problems. For generalisation experiments, we use 100 problems from all the other benchmarks.

For graph colouring experiments, we train our models using all problems from `flat-75-180` dataset. We select a model, given the performance on all 100 problems from `flat-100-239`. So, evaluation on these two datasets should not be used to judge the method's performance, and they are shown separately in Table 7.4. All the data from the second part of the table was not seen by the model during training (`flat-30-60`, `flat-50-115`, `flat-125-301`, `flat-150-360`, `flat-175-417`, `flat-200-479`).

B

Supplement to the Overview of Multi-Task Optimisers

Contents

This section presents the proofs and the technical results omitted from §4.4, along with a description of the use of per-task gradients with respect to the last shared activation for encoder-decoder architectures (usually less expensive than per-task gradients with respect to shared parameters).

B.1 MGDA

Proposition 1. *The MGDA SMTO [SK18] converges to a superset of the convergence points of unitary scalarisation. More specifically, it converges to any point* $\boldsymbol{\theta}^*_{||}$ *such that:* $\mathbf{0} \in Conv(\{\nabla_{\boldsymbol{\theta}^*_{||}} \mathcal{L}_i \mid i \in \mathcal{T}\})$.

Proof. As shown by Désidéri [Dés12], Equation (4.2) is a simplex-constrained norm-minimisation problem. In other words, the argument of the minimum is the projection of $\mathbf{0}$ onto the feasible set. Therefore:

$$\mathbf{g} = \mathbf{0} \iff \mathbf{0} \in \text{Conv}(\{\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i \mid i \in \mathcal{T}\}).$$

It then suffices to point out that $\sum_{i\in\mathcal{T}} \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i = \mathbf{0} \iff \sum_{i\in\mathcal{T}} \frac{1}{|\mathcal{T}|} \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i = \mathbf{0} \Rightarrow \mathbf{0} \in \text{Conv}(\{\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i \mid i \in \mathcal{T}\})$ to conclude the proof. □

Due to the cost of computing per-task gradients, Sener et al. [SK18] propose MGDA-UB, which replaces the gradients wrt the parameters $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i$ with the gradients wrt the shared activation $\nabla_{\mathbf{z}} \mathcal{L}_i$ in the computation of the coefficients of $\mathbf{g} = -\sum_i \alpha_i \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i$. This yields an upper bound on the objective of Equation (4.2), thus restricting the set of points the algorithm convergences to. Rather than directly relying on $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i$, $\mathbf{g}$ can then be obtained by computing the gradient of $\sum_{i\in\mathcal{T}} \alpha_i \mathcal{L}_i$ via reverse-mode differentiation, hence saving memory and compute.

Corollary 3. *The MGDA-UB SMTO by Sener et al. [SK18] converges to any point such that:* $\mathbf{0} \in \text{Conv}(\{\nabla_{\mathbf{z}} \mathcal{L}_i \mid i \in \mathcal{T}\})$. *Furthermore, if* $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_{\|}}$ *is non-singular, it converges to a superset of the convergence points of the unitary scalarisation.*

Proof. The first part of the proof proceeds as the proof of Proposition 1, noting that the MGDA-UB update is associated with the following problem:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2} \|\mathbf{g}\|_2^2 \\ \text{s.t.} \quad & \sum_i \alpha_i \nabla_{\mathbf{z}} \mathcal{L}_i = -\mathbf{g}, \quad \sum_{i\in\mathcal{T}} \alpha_i = 1, \\ & \alpha_i \geq 0 \qquad \forall\, i \in \mathcal{T}. \end{aligned}$$

In order to show that a stationary point of the unitary scalarisation satisfies $\mathbf{0} \in \text{Conv}(\{\nabla_{\mathbf{z}^*} \mathcal{L}_i \mid i \in \mathcal{T}\})$, we will assume $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_{\|}}$ is non-singular, as done by Sener

et al. [SK18, Theorem 1]. Then, relying on the chain rule, the result follows from:

$$\begin{aligned}
\sum_{i\in\mathcal{T}} \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i = \mathbf{0} &\iff \sum_{i\in\mathcal{T}} \frac{1}{|\mathcal{T}|} \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i = \mathbf{0} \\
&\iff \sum_{i\in\mathcal{T}} \frac{\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_\parallel}}{|\mathcal{T}|} \nabla_{\mathbf{z}} \mathcal{L}_i = \mathbf{0} \\
&\iff \left(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_\parallel}\right)^{-1} \frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_\parallel} \sum_{i\in\mathcal{T}} \frac{1}{|\mathcal{T}|} \nabla_{\mathbf{z}} \mathcal{L}_i = \mathbf{0} \\
&\iff \sum_{i\in\mathcal{T}} \frac{1}{|\mathcal{T}|} \nabla_{\mathbf{z}} \mathcal{L}_i = \mathbf{0} \\
&\Rightarrow \mathbf{0} \in \operatorname{Conv}(\{\nabla_{\mathbf{z}} \mathcal{L}_i \mid i \in \mathcal{T}\})
\end{aligned}$$

□

B.2 IMTL

Proposition 2. *IMTL by Liu et al. [Liu+21b] updates $\boldsymbol{\theta}_\parallel$ by taking a step in the* steepest descent direction whose *cosine similarity with per-task gradients is the same across tasks.*

Proof. First, Equation (4.4) solves the linear system in $\boldsymbol{\alpha} := [\alpha_1, \ldots, \alpha_m]$ given by:

$$\begin{aligned}
&\mathbf{g}^T \left(\frac{\nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_1}{\left\| \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_1 \right\|} - \frac{\nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i}{\left\| \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i \right\|} \right) = \mathbf{0} \qquad \forall\, i \in \mathcal{T} \setminus \{1\}, \\
&\mathbf{g} = -\sum_i \alpha_i \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i, \quad \sum_{i\in\mathcal{T}} \alpha_i = 1,
\end{aligned}$$

which corresponds to finding a point of $\mathcal{A}' := \operatorname{Aff}(\left\{ \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i \mid i \in \mathcal{T} \right\})$ which is orthogonal to $\mathcal{A} := \operatorname{Aff}\left(\left\{ \frac{\nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i}{\left\| \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i \right\|} \mid i \in \mathcal{T} \right\} \right)$. By definition of the vector orthogonality to a subspace, any vector orthogonal to $\mathcal{A}$ is also orthogonal to the vector subspace spanned by differences of vectors belonging to $\mathcal{A}$. As this subspace has $m-1$ dimensions, any vector orthogonal to $\left(\frac{\nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_1}{\left\| \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_1 \right\|} - \frac{\nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i}{\left\| \nabla_{\boldsymbol{\theta}_\parallel} \mathcal{L}_i \right\|} \right)$ for each $i \in \mathcal{T} \setminus \{1\}$ is orthogonal to the entire subspace.

Second, consider the problem of finding a point in $\mathcal{A}$ that is orthogonal to the linear subspace spanned by differences of vectors in $\mathcal{A}$. In other words, we seek the

projection of $\mathbf{0}$ onto $\mathcal{A}$. Recalling the definition of $\mathcal{A}$, we can write:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\left\|\mathbf{g}'\right\|_2^2 \\ \text{s.t.} \quad & \sum_i \alpha_i \frac{\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i}{\left\|\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i\right\|} = -\mathbf{g}', \quad \sum_i \alpha_i = 1. \end{aligned} \tag{B.1}$$

The solution of equation (B.1) is always collinear to the solution of equation (4.4). In fact, if a vector $\mathbf{g} \in \mathcal{A}'$ is orthogonal to the affine subspace $\mathcal{A}$ (or to the linear subspace spanned by differences of its members), then $\gamma\mathbf{g} = \left(-\gamma \sum_i \left(\alpha_i \left\|\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i\right\|\right) \frac{\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i}{\left\|\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i\right\|}\right)$ is orthogonal to $\mathcal{A}$ as well, and $\gamma = \frac{1}{\sum_i \left(\alpha_i \left\|\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i\right\|\right)} \implies \gamma\mathbf{g} \in \mathcal{A}$.

Finally, Equation (B.1) differs from Equation (4.2) in two aspects: $\boldsymbol{\alpha}$ is not constrained to be non-negative (hence the convex hull is replaced by the affine hull), and the task vectors are normalised. Therefore, Equation (B.1) is the dual of:

$$\begin{aligned} \min_{\mathbf{g},\epsilon} \quad & \epsilon + \frac{1}{2}\left\|\mathbf{g}\right\|_2^2 \\ \text{s.t.} \quad & \frac{\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i^T}{\left\|\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i\right\|} \mathbf{g} = \epsilon \qquad \forall\ i = 1, \ldots, m. \end{aligned} \tag{B.2}$$

The Proposition then follows by comparing Equation (B.2) with Equation (4.1) and recalling that IMTL-L only adds a scaling factor to the chosen update direction. □

Corollary 1. *IMTL by Liu et al. [Liu+21b] converges to a superset of the Pareto-optimal points for $\boldsymbol{\theta}_{\|}$ (and hence of the convergence points of the unitary scalarisation). More specifically, it converges to any point $\boldsymbol{\theta}_{\|}^*$ such that:*

$$\mathbf{0} \in \mathit{Aff}\left(\left\{\frac{\nabla_{\boldsymbol{\theta}^*_{\|}} \mathcal{L}_i}{\left\|\nabla_{\boldsymbol{\theta}^*_{\|}} \mathcal{L}_i\right\|} \mid i \in \mathcal{T}\right\}\right).$$

Proof. Inspecting Equation (B.2), which yields a collinear point to the IMTL update, reveals that IMTL might converge to non Pareto-stationary points: due to the restrictive equality constraints, the minimiser of Equation (B.2) might be $\mathbf{0}$ even if a descent direction exists. Furthermore, its dual, Equation (B.1), implies that:

$$\begin{aligned} \mathbf{g} = \mathbf{0} &\iff \mathbf{0} \in \operatorname{Aff}\left(\left\{\frac{\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i}{\left\|\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i\right\|} \mid i \in \mathcal{T}\right\}\right) \\ &\iff \mathbf{0} \in \operatorname{Aff}\left(\left\{\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i \mid i \in \mathcal{T}\right\}\right), \end{aligned}$$

which, noting that $\operatorname{Conv}(\mathcal{A}) \subseteq \operatorname{Aff}(\mathcal{A})$ for any $\mathcal{A}$, concludes the proof. □

Similarly to MGDA-UB, Liu et al. [Liu+21b] advocate using $\nabla_{\mathbf{z}}\mathcal{L}_i$ in place of $\nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_i$ while solving Equation (4.4), typically reducing the cost of computing the coefficients of $\mathbf{g} = -\sum_i \alpha_i \nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_i$.

Corollary 4. *When employing the approximation of problem (4.4) that relies on $\nabla_{\mathbf{z}}\mathcal{L}_i$, IMTL by Liu et al. [Liu+21b] converges to $\mathbf{0} \in \mathit{Aff}\left(\left\{\frac{\nabla_{\mathbf{z}}\mathcal{L}_i}{\|\nabla_{\mathbf{z}}\mathcal{L}_i\|} \mid i \in \mathcal{T}\right\}\right)$. If $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_\parallel}$ is non-singular, this is a superset of the convergence points of the unitary scalarisation.*

Proof. Following the proof of Proposition 2, the following problem yields a collinear point to the $\nabla_{\mathbf{z}}\mathcal{L}_i$-approximate IMTL update:

$$\begin{aligned} \max_{\boldsymbol{\alpha}} \quad & -\frac{1}{2}\|\mathbf{g}'\|_2^2 \\ \text{s.t.} \quad & \sum_i \alpha_i \frac{\nabla_{\mathbf{z}}\mathcal{L}_i}{\|\nabla_{\mathbf{z}}\mathcal{L}_i\|} = -\mathbf{g}', \quad \sum_i \alpha_i = 1. \end{aligned}$$

Therefore:

$$\mathbf{g} = \mathbf{0} \iff \mathbf{0} \in \operatorname{Aff}\left(\left\{\frac{\nabla_{\mathbf{z}}\mathcal{L}_i}{\|\nabla_{\mathbf{z}}\mathcal{L}_i\|} \mid i \in \mathcal{T}\right\}\right).$$

Finally, assuming $\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_\parallel}$ is non-singular, we can replicate the procedure in the proof of Corollary 3 to get:

$$\begin{aligned} &\sum_{i\in\mathcal{T}} \nabla_{\boldsymbol{\theta}_\parallel}\mathcal{L}_i = \mathbf{0} \iff \sum_{i\in\mathcal{T}} \frac{1}{|\mathcal{T}|}\nabla_{\mathbf{z}}\mathcal{L}_i = \mathbf{0} \\ &\iff \sum_{i\in\mathcal{T}} \frac{\|\nabla_{\mathbf{z}}\mathcal{L}_i\|}{|\mathcal{T}|}\frac{\nabla_{\mathbf{z}}\mathcal{L}_i}{\|\nabla_{\mathbf{z}}\mathcal{L}_i\|} = \mathbf{0} \\ &\iff \left(\frac{|\mathcal{T}|}{\sum_{i\in\mathcal{T}}\left(\|\nabla_{\mathbf{z}}\mathcal{L}_i\|\right)}\right)\sum_{i\in\mathcal{T}} \frac{\|\nabla_{\mathbf{z}}\mathcal{L}_i\|}{|\mathcal{T}|}\frac{\nabla_{\mathbf{z}}\mathcal{L}_i}{\|\nabla_{\mathbf{z}}\mathcal{L}_i\|} = \mathbf{0} \\ &\Rightarrow \mathbf{0} \in \operatorname{Conv}\left(\left\{\frac{\nabla_{\mathbf{z}}\mathcal{L}_i}{\|\nabla_{\mathbf{z}}\mathcal{L}_i\|} \mid i \in \mathcal{T}\right\}\right) \\ &\Rightarrow \mathbf{0} \in \operatorname{Aff}\left(\left\{\frac{\nabla_{\mathbf{z}}\mathcal{L}_i}{\|\nabla_{\mathbf{z}}\mathcal{L}_i\|} \mid i \in \mathcal{T}\right\}\right), \end{aligned}$$

which shows that $\operatorname{Aff}\left(\left\{\frac{\nabla_{\mathbf{z}}\mathcal{L}_i}{\|\nabla_{\mathbf{z}}\mathcal{L}_i\|} \mid i \in \mathcal{T}\right\}\right)$ contains the convergence points of the unitary scalarisation. □

B.3 PCGrad

Proposition 3. *PCGrad is equivalent to a dynamic, possibly stochastic, loss rescaling for $\boldsymbol{\theta}_{\|}$. At each iteration, per-task gradients are rescaled as follows:*

$$\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i \leftarrow \left(1+\textstyle\sum_{j\in\mathcal{T}\setminus\{i\}} d_{ji}\right)\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i,\ \ d_{ji}\in\left[0,\frac{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j\right\|}{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i\right\|}\right].$$

Furthermore, if $|\mathcal{T}|>2$, d_{ji} is a random variable, and the above range contains its support.

Proof. We start by pointing out that:

$$\begin{aligned}\left[\frac{-\mathbf{g}_i^T\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j(\mathbf{x})}{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j\right\|^2}\right]_+ &= \left[\frac{-\mathbf{g}_i^T\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j(\mathbf{x})}{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j\right\|}\right]_+\frac{1}{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j\right\|}\\ &= \left[-\cos(\mathbf{g}_i,\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j)\left\|\mathbf{g}_i\right\|\right]_+\frac{1}{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j\right\|}\\ &\in\left[0,\frac{\|\mathbf{g}_i\|}{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j\right\|}\right].\end{aligned}$$

As $\mathbf{g}_i$ is obtained by iterative projections of $\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i$ onto the normals of $\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j\ \forall j\in\mathcal{T}\setminus\{i\}$, and the norm of a vector can only decrease or remain unvaried after projections, we can write the coefficient of each $\mathbf{g}_i$ update as:

$$d_{ij}:=\left[\frac{-\mathbf{g}_i^T\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j(\mathbf{x})}{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j\right\|^2}\right]_+\in\left[0,\frac{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i\right\|}{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j\right\|}\right],\ \ \forall i\neq j.$$

Furthermore, if $|\mathcal{T}|>2$ the contraction factor $\frac{\|\mathbf{g}_i\|}{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i\right\|}$ for the norm of g_i depends on the ordering of the projections, which is stochastic by design [Yu+20]. Therefore, d_{ij} a random variable whose support is contained in $\left[0,\frac{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i\right\|}{\left\|\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j\right\|}\right]$. Finally, exploiting the definition of d_{ij}, we can re-write Equation (4.5) as:

$$\begin{aligned}-\mathbf{g}&=\sum_{i\in\mathcal{T}}\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i+\sum_{i\in\mathcal{T}}\sum_{j\in\mathcal{T}\setminus\{i\}}d_{ij}\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j=\sum_{i\in\mathcal{T}}\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i+\sum_{j\in\mathcal{T}}\sum_{i\in\mathcal{T}\setminus\{j\}}d_{ji}\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i\\ &=\sum_{j\in\mathcal{T}}\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j+\sum_{j\in\mathcal{T}}\sum_{i\in\mathcal{T}\setminus\{j\}}d_{ji}\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i=\sum_{j\in\mathcal{T}}\left(\sum_{i\in\mathcal{T}\setminus\{j\}}d_{ji}\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_i+\nabla_{\boldsymbol{\theta}_{\|}}\mathcal{L}_j\right).\end{aligned}$$

Introducing (and then removing, using their definition) dummy variables $d_{jj} = 1$:

$$-\mathbf{g} = \sum_{j\in\mathcal{T}} \left(\sum_{i\in\mathcal{T}\setminus\{j\}} d_{ji} \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i + d_{jj} \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_j \right) = \sum_{j\in\mathcal{T}} \left(\sum_{i\in\mathcal{T}} d_{ji} \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i \right) = \sum_{i\in\mathcal{T}} \left(\sum_{j\in\mathcal{T}} d_{ji} \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i \right)$$
$$= \sum_{i\in\mathcal{T}} \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i \left(\sum_{j\in\mathcal{T}} d_{ji} \right) = \sum_{i\in\mathcal{T}} \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i \left(1 + \sum_{j\in\mathcal{T}\setminus\{i\}} d_{ji} \right),$$

from which the result trivially follows. □

Corollary 2. *If $|\mathcal{T}| = 2$, PCGrad will stop at any point where $\cos(\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1, \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2) = -1$. Furthermore, if $\mathcal{L}_1$ and $\mathcal{L}_2$ are differentiable, and $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}^{MT}$ is L-Lipschitz with $L > 0$, PCGrad with step size $t < \frac{1}{L}$ converges to a superset of the convergence points of the unitary scalarisation.*

Proof. Let us start from the first statement, which does not require any assumption on the loss landscape. From Proposition 3, we get:

$$\begin{aligned} -\mathbf{g} &= \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1 \left(1 + d_{21}\right) + \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2 \left(1 + d_{12}\right) \\ &= \left(1 + \left[\frac{-\cos(\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1, \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2) \left\| \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2 \right\|}{\left\| \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1 \right\|} \right]_+ \right) \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1 \\ &+ \left(1 + \left[\frac{-\cos(\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1, \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2) \left\| \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1 \right\|}{\left\| \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2 \right\|} \right]_+ \right) \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2, \end{aligned}$$

which shows that gradient norms are rebalanced proportionally to the angle between them in case of conflicting gradient directions. For $\cos(\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1, \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2) = -1$, the above evaluates to:

$$-\mathbf{g} = \left(\frac{\left\| \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1 \right\| + \left\| \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2 \right\|}{\left\| \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1 \right\|} \right) \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1 + \left(\frac{\left\| \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1 \right\| + \left\| \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2 \right\|}{\left\| \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2 \right\|} \right) \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2.$$

The first part of the result then follows by pointing out that, if $\cos(\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1, \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2) = -1$, then $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_1 = -\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_2$, and hence $\mathbf{g} = \mathbf{0}$. We remark that a similar proof appears in [Yu+20, Theorem 1 and Proposition 1]. However, our derivation relaxes the author's assumptions on $\mathcal{L}^{\text{MT}}$ and is therefore applicable to the training of piecewise-linear networks (e.g. neural networks with ReLU activation function).

Finally, given the assumptions on differentiability and smoothness, we need to prove that PCGrad converges to the stationary points of the unitary scalarisation: this directly follows from [Yu+20, Proposition 1]. □

B.4 GradDrop

Proposition 4. *Let* $\mathcal{L}^{RGD}(\boldsymbol{\theta}_{\|}) := \sum_{i\in\mathcal{T}} u_i \mathcal{L}_i(\boldsymbol{\theta}_{\|})$*, where* $u_i \sim$ *Bernoulli*(p) $\forall i \in \mathcal{T}$ *and* $p \in (0, 1]$*. The gradient* $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}^{RGD}$ *is always zero if and only if* $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i = \mathbf{0}$ $\forall i \in \mathcal{T}$*. In other words, the result from [Che+20, Proposition 1] can be obtained without any information on the sign of per-task gradients.*

Proposition 4 can be proved by adapting the proof from Chen et al. [Che+20, Proposition 1]: it suffices to replace $f(\mathcal{P})$ with the Bernoulli parameter p, which is non-negative by definition. In our opinion, this seriously undermines the conflicting gradient hypothesis that motivated GradDrop. We now provide a straightforward and self-contained proof for the reader's convenience.

Proof. Let us start from the statement on $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}^{\text{RGD}}$. Looking at the definition of $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}^{\text{RGD}}$, we can see that if $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i = \mathbf{0}$ $\forall i \in \mathcal{T}$, then $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}^{\text{RGD}}$ is surely $\mathbf{0}$. On the other hand, if $\exists j : \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_j \neq \mathbf{0}$, then:

$$\begin{aligned}\mathbb{P}\left[\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}^{\text{RGD}} \neq \mathbf{0}\right] &\geq \mathbb{P}\left[\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}^{\text{RGD}} = \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_j\right] \\ &= p(1-p)^{|\mathcal{T}|-1} > 0,\end{aligned}$$

where the first inequality comes from the fact that $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}^{\text{RGD}} = \nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_j$ is only one of the many instances of a non-null $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}^{\text{RGD}}$. □

Proposition 5. *Let us assume, as often demonstrated in the single-task case [MBB18; ALS19], that the multi-task network has the capacity to interpolate the data on all tasks at once:* $\min_{\boldsymbol{\theta}} \mathcal{L}^{MT} = \sum_{i\in\mathcal{T}} \min_{\boldsymbol{\theta}} \mathcal{L}_i$*, and that its training by gradient descent attains such global minimum. Then, if* $\inf_{\boldsymbol{\theta}} \mathcal{L}_i > -\infty$ $\forall$ $i \in \mathcal{T}$*, unitary scalarisation converges to a joint minimum.*

Proof. It suffices to point out that if $\mathcal{L}^{\text{MT}}(\boldsymbol{\theta}^*) = \sum_{i\in\mathcal{T}} \min_{\boldsymbol{\theta}} \mathcal{L}_i$, then the globally optimal loss is attained for all tasks. In other words $\mathcal{L}_i(\boldsymbol{\theta}^*) = \min_{\boldsymbol{\theta}} \mathcal{L}_i$ $\forall i \in \mathcal{T}$, and hence $\nabla_{\boldsymbol{\theta}^*} \mathcal{L}_i = \mathbf{0}$ $\forall$ $i \in \mathcal{T}$ (joint minimum). Furthermore, running gradient descent on $\min_{\boldsymbol{\theta}} \mathcal{L}^{\text{MT}}$ corresponds to the unitary scalarisation (§4.2), which concludes the proof. In other words, by the assumption, the unitary scalarisation reaches a global minimum, which is a joint minimum. □

On encoder-decoder architectures, similarly to MGDA and IMTL (see appendices B.1 and B.2), the authors do not apply GradDrop on $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}_i$, but rather on a the usually less expensive $\nabla_{\mathbf{z}} \mathcal{L}_i$. Let $\text{sign}(\mathbf{x})$ stand for the element-wise sign operator applied on $\mathbf{x}$. In more detail, they compute the GradDrop sign purity scores $\mathbf{p}$ from Equation (4.6) on $\sum_{i=1}^{n} \left(\text{sign}(\mathbf{z}) \odot \nabla_{\mathbf{z}} \mathcal{L}_i\right)[i] \in \mathbb{R}^r$, and then apply Equation (4.6) on the $\nabla_{\mathbf{z}} \mathcal{L}_i$ gradients, yielding a vector $\mathbf{g}_z \in \mathbb{R}^{n \times r}$. Then, relying on reverse-mode differentiation, the update direction in the space of the parameters $\boldsymbol{\theta}_{\|}$ is obtained via a Jacobian-vector product: $\mathbf{g} = -\left(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_{\|}}\right)^T \mathbf{g}_z$. Such a computation replaces the similar $\nabla_{\boldsymbol{\theta}_{\|}} \mathcal{L}^{\text{MT}} = \left(\frac{\partial \mathbf{z}}{\partial \boldsymbol{\theta}_{\|}}\right)^T \nabla_{\mathbf{z}} \mathcal{L}^{\text{MT}}$ from the unitary scalarisation.

C

Training Graph Networks in RL: tricks of the trade

Over the course of this work, we developed an intuition for what is essential when training graph-based policies and value functions in incompatible MTRL. We provide those as advice, and an extensive empirical study is needed to confirm whether our intuition is correct across different problem settings and hyperparameter choices.

C.1 Use lower learning rates and gradient clipping

The first couple of updates usually has extremely large gradients. A large update will knock out ReLUs and oversaturate hyperbolic tangent activation often used with MuJoCo benchmarks.

C.2 Use bigger batches

GNNs tend to overfit. Using larger batches has a positive effect on training such models.

C.3 Debugging a model in a supervised setting and moving it to the RL setup saves a lot of time

This is a common technique in RL with a illustrative example of Silver et al. [Sil+16] and Vinyals et al. [Vin+19]. Often, the architecture that works in a supervised setting will also work in the RL setup.

C.4 Not all graphs are the same

Semantics matters when designing an architecture of a GNN. For some of the applications, the locality is a crucial inductive bias. Locality bias was a hurdle for locomotion environments we describe in Chapter 6. However, we believe that locality has played a significant role in our work in Chapter 7.

C.5 Normalisation is extremely important

Normalising the outputs of message-passing layers significantly improves the stability and performance of GNNs. There is no research rigorously investigating this phenomenon, but many existing works use normalisation, supporting the importance of this technique.

From our experience, Layer Normalisation [BKH16] works best and is the easiest to work with since its behaviour does not differ in inference and training modes.

References

[Abd+18a] Abbas Abdolmaleki, Jost Tobias Springenberg, Jonas Degrave, Steven Bohez, Yuval Tassa, Dan Belov, Nicolas Heess, and Martin A. Riedmiller. "Relative Entropy Regularized Policy Iteration". In: *CoRR* abs/1812.02256 (2018). arXiv: 1812.02256. URL: http://arxiv.org/abs/1812.02256 (cit. on p. 24).

[Abd+18b] Abbas Abdolmaleki, Jost Tobias Springenberg, Jonas Degrave, Steven Bohez, Yuval Tassa, Dan Belov, Nicolas Heess, and Martin A. Riedmiller. "Relative Entropy Regularized Policy Iteration". In: *CoRR* abs/1812.02256 (2018). arXiv: 1812.02256. URL: http://arxiv.org/abs/1812.02256 (cit. on p. 30).

[Abd+18c] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Rémi Munos, Nicolas Heess, and Martin A. Riedmiller. "Maximum a Posteriori Policy Optimisation". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=S1ANxQW0b (cit. on p. 69).

[Aga+20] Akshat Agarwal, Sumit Kumar, Katia P. Sycara, and Michael Lewis. "Learning Transferable Cooperative Behavior in Multi-Agent Teams". In: *International Conference on Autonomous Agents and Multiagent Systems, AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 1741–1743. URL: https://dl.acm.org/doi/abs/10.5555/3398761.3398967 (cit. on p. 44).

[AKL17] Jacob Andreas, Dan Klein, and Sergey Levine. "Modular Multitask Reinforcement Learning with Policy Sketches". In: *International Conference on Machine Learning, ICML*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 166–175. URL: http://proceedings.mlr.press/v70/andreas17a.html (cit. on p. 24).

[ALS19] Zeyuan Allen-Zhu, Yuanzhi Li, and Zhao Song. "A Convergence Theory for Deep Learning via Over-Parameterization". In: *International Conference on Machine Learning, ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 242–252. URL:

http://proceedings.mlr.press/v97/allen-zhu19a.html (cit. on p. 165).

[Ama96] Shun-ichi Amari. "Neural Learning in Structured Parameter Spaces - Natural Riemannian Gradient". In: *Advances in Neural Information Processing Systems*. MIT Press, 1996, pp. 127–133. URL: http://papers.nips.cc/paper/1248-neural-learning-in-structured-parameter-spaces-natural-riemannian-gradient (cit. on p. 69).

[AMW19] Saeed Amizadeh, Sergiy Matusevych, and Markus Weimer. "Learning To Solve Circuit-SAT: An Unsupervised Differentiable Approach". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=BJxgz2R9t7 (cit. on p. 121).

[Ant+17] Rika Antonova, Silvia Cruciani, Christian Smith, and Danica Kragic. "Reinforcement Learning for Pivoting Task". In: *CoRR* abs/1703.00472 (2017). arXiv: 1703.00472. URL: http://arxiv.org/abs/1703.00472 (cit. on p. 29).

[Aro+20] Simran Arora, Avner May, Jian Zhang, and Christopher Ré. "Contextual Embeddings: When Are They Worth It?" In: *Annual Meeting of the Association for Computational Linguistics, ACL*. Association for Computational Linguistics, 2020, pp. 2650–2663. URL: https://doi.org/10.18653/v1/2020.acl-main.236 (cit. on p. 83).

[Aue+02] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire. "The Nonstochastic Multiarmed Bandit Problem". In: *SIAM J. Comput.* 32.1 (2002), pp. 48–77. URL: https://doi.org/10.1137/S0097539701398375 (cit. on p. 36).

[AY20] Uri Alon and Eran Yahav. "On the Bottleneck of Graph Neural Networks and its Practical Implications". In: *CoRR* abs/2006.05205 (2020). arXiv: 2006.05205. URL: https://arxiv.org/abs/2006.05205 (cit. on p. 92).

[Bak+20] Bowen Baker, Ingmar Kanitscheider, Todor M. Markov, Yi Wu, Glenn Powell, Bob McGrew, and Igor Mordatch. "Emergent Tool Use From Multi-Agent Autocurricula". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=SkxpxJBKwS (cit. on p. 41).

[Ban+19] Kshitij Bansal, Sarah M. Loos, Markus N. Rabe, Christian Szegedy, and Stewart Wilcox. "HOList: An Environment for Machine Learning of Higher Order Logic Theorem Proving". In: *International Conference on Machine Learning, ICML*. Vol. 97. Proceedings of Machine Learning Research.

PMLR, 2019, pp. 454–463. URL: http://proceedings.mlr.press/v97/bansal19a.html (cit. on p. 37).

[Bap+19] Victor Bapst, Alvaro Sanchez-Gonzalez, Carl Doersch, Kimberly L. Stachenfeld, Pushmeet Kohli, Peter W. Battaglia, and Jessica B. Hamrick. "Structured agents for physical construction". In: *International Conference on Machine Learning, ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 464–474. URL: http://proceedings.mlr.press/v97/bapst19a.html (cit. on pp. 43, 110, 124).

[Bar+17] André Barreto, Will Dabney, Rémi Munos, Jonathan J. Hunt, Tom Schaul, David Silver, and Hado van Hasselt. "Successor Features for Transfer in Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4055–4065. URL: https://proceedings.neurips.cc/paper/2017/hash/350db081a661525235354dd3e19b8c05-Abstract.html (cit. on p. 31).

[Bas+20] Andrea Bassich, Francesco Foglino, Matteo Leonetti, and Daniel Kudenko. "Curriculum Learning with a Progression Function". In: *CoRR* abs/2008.00511 (2020). arXiv: 2008.00511. URL: https://arxiv.org/abs/2008.00511 (cit. on p. 35).

[Bat+18] Peter W. Battaglia et al. "Relational inductive biases, deep learning, and graph networks". In: *CoRR* abs/1806.01261 (2018). arXiv: 1806.01261. URL: http://arxiv.org/abs/1806.01261 (cit. on pp. 15, 17, 39, 88, 93, 106, 110, 123).

[BCN06] Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. "Model compression". In: *Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2006, pp. 535–541. URL: https://doi.org/10.1145/1150402.1150464 (cit. on p. 25).

[Bea+16] Charles Beattie et al. "DeepMind Lab". In: *CoRR* abs/1612.03801 (2016). arXiv: 1612.03801. URL: http://arxiv.org/abs/1612.03801 (cit. on pp. 24, 39).

[Bel+13] Marc G. Bellemare, Yavar Naddaf, Joel Veness, and Michael Bowling. "The Arcade Learning Environment: An Evaluation Platform for General Agents". In: *J. Artif. Intell. Res.* 47 (2013), pp. 253–279. URL: https://doi.org/10.1613/jair.3912 (cit. on pp. 24, 36).

[Bel+17] Irwan Bello, Hieu Pham, Quoc V. Le, Mohammad Norouzi, and Samy Bengio. "Neural Combinatorial Optimization with Reinforcement Learning". In: *International Conference on Learning Representations,*

ICLR, Workshop Track Proceedings. OpenReview.net, 2017. URL: https://openreview.net/forum?id=Bk9mxlSFx (cit. on pp. 43, 122).

[Bel+19] Marc G. Bellemare, Will Dabney, Robert Dadashi, Adrien Ali Taiga, Pablo Samuel Castro, Nicolas Le Roux, Dale Schuurmans, Tor Lattimore, and Clare Lyle. "A Geometric Perspective on Optimal Representations for Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. 2019, pp. 4360–4371. URL: https://proceedings.neurips.cc/paper/2019/hash/3cf2559725a9fdfa602ec8c887440f32-Abstract.html (cit. on p. 39).

[Ber+19] Christopher Berner et al. "Dota 2 with Large Scale Deep Reinforcement Learning". In: *CoRR* abs/1912.06680 (2019). arXiv: 1912.06680. URL: http://arxiv.org/abs/1912.06680 (cit. on p. 12).

[BH03] Bart Bakker and Tom Heskes. "Task Clustering and Gating for Bayesian Multitask Learning". In: *J. Mach. Learn. Res.* 4 (2003), pp. 83–99. URL: http://jmlr.org/papers/v4/bakker03a.html (cit. on p. 46).

[BKC17] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. "SegNet: A Deep Convolutional Encoder-Decoder Architecture for Image Segmentation". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 39.12 (2017), pp. 2481–2495. URL: https://doi.org/10.1109/TPAMI.2016.2644615 (cit. on p. 133).

[BKH16] Lei Jimmy Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. "Layer Normalization". In: *CoRR* abs/1607.06450 (2016). arXiv: 1607.06450. URL: http://arxiv.org/abs/1607.06450 (cit. on pp. 94, 168).

[BKW20] Wendelin Boehmer, Vitaly Kurin, and Shimon Whiteson. "Deep Coordination Graphs". In: *International Conference on Machine Learning, ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 980–991. URL: http://proceedings.mlr.press/v119/boehmer20a.html (cit. on p. 40).

[Bla+21] Charles Blake, Vitaly Kurin, Maximilian Igl, and Shimon Whiteson. "Snowflake: Scaling GNNs to high-dimensional continuous control via parameter freezing". In: *Advances in Neural Information Processing Systems*. 2021, pp. 23983–23992. URL: https://proceedings.neurips.cc/paper/2021/hash/c952ce98517ac529c60744ac28364b03-Abstract.html (cit. on p. 69).

[BM01] Ella Bingham and Heikki Mannila. "Random projection in dimensionality reduction: applications to image and text data". In: *ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM,

2001, pp. 245–250. URL: https://doi.org/10.1145/502512.502546 (cit. on p. 83).

[Bro+16] Greg Brockman, Vicki Cheung, Ludwig Pettersson, Jonas Schneider, John Schulman, Jie Tang, and Wojciech Zaremba. "OpenAI Gym". In: *CoRR* abs/1606.01540 (2016). arXiv: 1606.01540. URL: http://arxiv.org/abs/1606.01540 (cit. on pp. 70, 72, 79, 110, 151).

[Bro+17a] Andrew Brock, Theodore Lim, James M. Ritchie, and Nick Weston. "FreezeOut: Accelerate Training by Progressively Freezing Layers". In: *CoRR* abs/1706.04983 (2017). arXiv: 1706.04983. URL: http://arxiv.org/abs/1706.04983 (cit. on p. 77).

[Bro+17b] Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. "Geometric Deep Learning: Going beyond Euclidean data". In: *IEEE Signal Process. Mag.* 34.4 (2017), pp. 18–42. URL: https://doi.org/10.1109/MSP.2017.2693418 (cit. on p. 39).

[Bro+20] Tom B. Brown et al. "Language Models are Few-Shot Learners". In: *Advances in Neural Information Processing Systems*. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html (cit. on p. 27).

[BS07] Bikramjit Banerjee and Peter Stone. "General Game Learning Using Knowledge Transfer". In: *International Joint Conference on Artificial Intelligence*. 2007, pp. 672–677. URL: http://ijcai.org/Proceedings/07/Papers/107.pdf (cit. on p. 31).

[Cai+19] Qingpeng Cai, Will Hang, Azalia Mirhoseini, George Tucker, Jingtao Wang, and Wei Wei. "Reinforcement Learning Driven Heuristic Optimization". In: *CoRR* abs/1906.06639 (2019). arXiv: 1906.06639. URL: http://arxiv.org/abs/1906.06639 (cit. on p. 123).

[Cap+21] Quentin Cappart, Didier Chételat, Elias B. Khalil, Andrea Lodi, Christopher Morris, and Petar Velickovic. "Combinatorial Optimization and Reasoning with Graph Neural Networks". In: *International Joint Conference on Artificial Intelligence, IJCAI 2021*. ijcai.org, 2021, pp. 4348–4355. URL: https://doi.org/10.24963/ijcai.2021/595 (cit. on pp. 30, 66, 125, 128).

[Car+18] Victor Carbune, Thierry Coppey, Alexander Daryin, Thomas Deselaers, Nikhil Sarda, and Jay Yagnik. *SmartChoices: Hybridizing Programming and Machine Learning*. 2018. arXiv: 1810.00619 [cs.LG] (cit. on pp. 4, 43, 123, 125).

[Car97a] Rich Caruana. "Multitask learning". PhD thesis. Pittsburgh, PA 15213: School of Computer Science, Carnegie Mellon University, 1997. URL: http://reports-archive.adm.cs.cmu.edu/anon/1997/CMU-CS-97-203.pdf (cit. on p. 20).

[Car97b] Rich Caruana. "Multitask learning". In: *Machine learning* 28.1 (1997), pp. 41–75. URL: https://link.springer.com/article/10.1023/A:1007379606734 (cit. on pp. 2, 46).

[CBL19] Kyle Cranmer, Johann Brehmer, and Gilles Louppe. "The frontier of simulation-based inference". In: *CoRR* abs/1911.01429 (2019). arXiv: 1911.01429. URL: http://arxiv.org/abs/1911.01429 (cit. on p. 128).

[Cha21] Nethack Challenge. *Neurips 2021 Nethack Challenge*. [Online; accessed 23-August-2022]. 2021. URL: https://nethackchallenge.com/ (cit. on p. 43).

[Che+18a] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. "Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation". In: *European Conference on Computer Vision ECCV*. Vol. 11211. Lecture Notes in Computer Science. Springer, 2018, pp. 833–851. URL: https://doi.org/10.1007/978-3-030-01234-2%5C_49 (cit. on p. 133).

[Che+18b] Zhao Chen, Vijay Badrinarayanan, Chen-Yu Lee, and Andrew Rabinovich. "GradNorm: Gradient Normalization for Adaptive Loss Balancing in Deep Multitask Networks". In: *International Conference on Machine Learning, ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 793–802. URL: http://proceedings.mlr.press/v80/chen18a.html (cit. on pp. 20, 56).

[Che+19] Yevgen Chebotar, Ankur Handa, Viktor Makoviychuk, Miles Macklin, Jan Issac, Nathan D. Ratliff, and Dieter Fox. "Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience". In: *International Conference on Robotics and Automation, ICRA*. IEEE, 2019, pp. 8973–8979. URL: https://doi.org/10.1109/ICRA.2019.8793789 (cit. on p. 28).

[Che+20] Zhao Chen, Jiquan Ngiam, Yanping Huang, Thang Luong, Henrik Kretzschmar, Yuning Chai, and Dragomir Anguelov. "Just Pick a Sign: Optimizing Deep Multitask Models with Gradient Sign Dropout". In: *Advances in Neural Information Processing Systems*. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/16002f7a455a94aa4e91cc34ebdb9f2d-Abstract.html (cit. on pp. 9, 20, 46, 50, 52, 64, 65, 133, 143, 144, 165).

[Cho+14] Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Doha, Qatar: Association for Computational Linguistics, 2014, pp. 1724–1734. URL: https://www.aclweb.org/anthology/D14-1179 (cit. on p. 71).

[CKT91] Peter C. Cheeseman, Bob Kanefsky, and William M. Taylor. "Where the Really Hard Problems Are". In: *International Joint Conference on Artificial Intelligence*. Morgan Kaufmann, 1991, pp. 331–340. URL: http://ijcai.org/Proceedings/91-1/Papers/052.pdf (cit. on p. 112).

[CLG00] Rich Caruana, Steve Lawrence, and C. Lee Giles. "Overfitting in Neural Nets: Backpropagation, Conjugate Gradient, and Early Stopping". In: *Advances in Neural Information Processing Systems*. MIT Press, 2000, pp. 402–408. URL: https://proceedings.neurips.cc/paper/2000/hash/059fdcd96baeb75112f09fa1dcc740cc-Abstract.html (cit. on p. 62).

[CMG18] Tao Chen, Adithyavairavan Murali, and Abhinav Gupta. "Hardware Conditioned Policies for Multi-Robot Transfer Learning". In: *Advances in Neural Information Processing Systems*. 2018, pp. 9355–9366. URL: https://proceedings.neurips.cc/paper/2018/hash/b8cfbf77a3d250a4523ba67a65a7d031-Abstract.html (cit. on p. 40).

[Cob+19] Karl Cobbe, Oleg Klimov, Christopher Hesse, Taehoon Kim, and John Schulman. "Quantifying Generalization in Reinforcement Learning". In: *International Conference on Machine Learning, ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1282–1289. URL: http://proceedings.mlr.press/v97/cobbe19a.html (cit. on pp. 28, 57).

[Cob+20] Karl Cobbe, Christopher Hesse, Jacob Hilton, and John Schulman. "Leveraging Procedural Generation to Benchmark Reinforcement Learning". In: *International Conference on Machine Learning, ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 2048–2056. URL: http://proceedings.mlr.press/v119/cobbe20a.html (cit. on p. 30).

[Cor+09] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, 3rd Edition*. MIT Press, 2009. URL: http://mitpress.mit.edu/books/introduction-algorithms (cit. on p. 128).

[Cor+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE Computer Society, 2016, pp. 3213–3223. URL: https://doi.org/10.1109/CVPR.2016.350 (cit. on pp. 56, 145).

[CW08] Ronan Collobert and Jason Weston. "A unified architecture for natural language processing: deep neural networks with multitask learning". In: *International Conference on Machine Learning ICML*. Vol. 307. ACM International Conference Proceeding Series. ACM, 2008, pp. 160–167. URL: https://doi.org/10.1145/1390156.1390177 (cit. on p. 20).

[CW17] Kamil Andrzej Ciosek and Shimon Whiteson. "OFFER: Off-Environment Reinforcement Learning". In: *AAAI Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 1819–1825. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14378 (cit. on p. 28).

[CWP18] Maxime Chevalier-Boisvert, Lucas Willems, and Suman Pal. *Minimalistic Gridworld Environment for OpenAI Gym*. https://github.com/maximecb/gym-minigrid. 2018 (cit. on p. 28).

[Cza+19] Wojciech M. Czarnecki, Razvan Pascanu, Simon Osindero, Siddhant M. Jayakumar, Grzegorz Swirszcz, and Max Jaderberg. "Distilling Policy Distillation". In: *International Conference on Artificial Intelligence and Statistics, AISTATS*. Vol. 89. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1331–1340. URL: http://proceedings.mlr.press/v89/czarnecki19a.html (cit. on p. 26).

[CZY21] Shijie Chen, Yu Zhang, and Qiang Yang. "Multi-Task Learning in Natural Language Processing: An Overview". In: *CoRR* abs/2109.09138 (2021). arXiv: 2109.09138. URL: https://arxiv.org/abs/2109.09138 (cit. on p. 20).

[Dab+21] Will Dabney, André Barreto, Mark Rowland, Robert Dadashi, John Quan, Marc G. Bellemare, and David Silver. "The Value-Improvement Path: Towards Better Representations for Reinforcement Learning". In: *AAAI Conference on Artificial Intelligence*. AAAI Press, 2021, pp. 7160–7168. URL: https://ojs.aaai.org/index.php/AAAI/article/view/16880 (cit. on p. 39).

[DBV16] Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. "Convolutional Neural Networks on Graphs with Fast Localized Spectral Filtering". In: *Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, December 5-10, 2016, Barcelona, Spain*. Ed. by Daniel D. Lee, Masashi Sugiyama, Ulrike von Luxburg, Isabelle Guyon, and Roman Garnett. 2016, pp. 3837–3845. URL: https://proceedings.neurips.cc/paper/2016/hash/04df4d434d481c5bb723be1b6df1ee65-Abstract.html (cit. on p. 39).

[Deg+22] Jonas Degrave, Federico Felici, Jonas Buchli, Michael Neunert, Brendan Tracey, Francesco Carpanese, Timo Ewalds, Roland Hafner, Abbas Abdolmaleki, Diego de Las Casas, et al. "Magnetic control of tokamak plasmas through deep reinforcement learning". In: *Nature* 602.7897 (2022), pp. 414–419. URL: https://www.nature.com/articles/s41586-021-04301-9 (cit. on p. 30).

[DEr+20] Carlo D'Eramo, Davide Tateo, Andrea Bonarini, Marcello Restelli, and Jan Peters. "Sharing Knowledge in Multi-Task Deep Reinforcement Learning". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=rkgpv2VFvr (cit. on pp. 24, 40).

[Dés12] J. Désidéri. "Multiple-gradient descent algorithm (MGDA) for multiobjective optimization". In: *Comptes Rendus Mathematique* 350 (2012), pp. 313–318. URL: https://www.sciencedirect.com/science/article/pii/S1631073X12000738 (cit. on pp. 48, 49, 62, 159).

[Dev+17] Coline Devin, Abhishek Gupta, Trevor Darrell, Pieter Abbeel, and Sergey Levine. "Learning modular neural network policies for multi-task and multi-robot transfer". In: *IEEE International Conference on Robotics and Automation, ICRA*. IEEE, 2017, pp. 2169–2176. URL: https://doi.org/10.1109/ICRA.2017.7989250 (cit. on p. 24).

[Dev+19] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 4171–4186. URL: https://www.aclweb.org/anthology/N19-1423 (cit. on p. 104).

[Dhi+17] Bhuwan Dhingra, Hanxiao Liu, Ruslan Salakhutdinov, and William W. Cohen. "A Comparative Study of Word Embeddings for Reading Comprehension". In: *CoRR* abs/1703.00993 (2017). arXiv: 1703.00993. URL: http://arxiv.org/abs/1703.00993 (cit. on p. 83).

[Die95] Thomas G. Dietterich. "Overfitting and Undercomputing in Machine Learning". In: *ACM Comput. Surv.* 27.3 (1995), pp. 326–327. URL: https://doi.org/10.1145/212094.212114 (cit. on pp. 60, 62, 63).

[DK16] Finale Doshi-Velez and George Dimitri Konidaris. "Hidden Parameter Markov Decision Processes: A Semiparametric Regression Approach for Discovering Latent Task Parametrizations". In: *International Joint Conference on Artificial Intelligence, IJCAI*. IJCAI/AAAI Press, 2016, pp. 1432–1440. URL: http://www.ijcai.org/Abstract/16/206 (cit. on p. 28).

[Don+20] Xiaowen Dong, Dorina Thanou, Laura Toni, Michael M. Bronstein, and Pascal Frossard. "Graph Signal Processing for Machine Learning: A Review and New Perspectives". In: *IEEE Signal Process. Mag.* 37.6 (2020), pp. 117–127. URL: https://doi.org/10.1109/MSP.2020.3014591 (cit. on pp. 39, 40).

[Dua+17] Yan Duan, Marcin Andrychowicz, Bradly C. Stadie, Jonathan Ho, Jonas Schneider, Ilya Sutskever, Pieter Abbeel, and Wojciech Zaremba. "One-Shot Imitation Learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 1087–1098. URL: https://proceedings.neurips.cc/paper/2017/hash/ba3866600c3540f67c1e9575e213be0a-Abstract.html (cit. on p. 41).

[EMM05] Yaakov Engel, Shie Mannor, and Ron Meir. "Reinforcement learning with Gaussian processes". In: *International Conference on Machine Learning, ICML*. Vol. 119. ACM International Conference Proceeding Series. ACM, 2005, pp. 201–208. URL: https://doi.org/10.1145/1102351.1102377 (cit. on p. 22).

[EP04] Theodoros Evgeniou and Massimiliano Pontil. "Regularized multi–task learning". In: *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. ACM, 2004, pp. 109–117. URL: https://doi.org/10.1145/1014052.1014067 (cit. on p. 46).

[Esp+18] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures". In: *International Conference on Machine Learning, ICML*.

Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1406–1415. URL: http://proceedings.mlr.press/v80/espeholt18a.html (cit. on p. 24).

[Eys+21] Benjamin Eysenbach, Shreyas Chaudhari, Swapnil Asawa, Sergey Levine, and Ruslan Salakhutdinov. "Off-Dynamics Reinforcement Learning: Training for Transfer with Domain Classifiers". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=eqBwg3AcIAK (cit. on p. 28).

[FAL17] Chelsea Finn, Pieter Abbeel, and Sergey Levine. "Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks". In: *International Conference on Machine Learning, ICML*. Vol. 70. Proceedings of Machine Learning Research. PMLR, 2017, pp. 1126–1135. URL: http://proceedings.mlr.press/v70/finn17a.html (cit. on p. 33).

[FB12] Alex Flint and Matthew B. Blaschko. "Perceptron Learning of SAT". In: *Advances in Neural Information Processing Systems*. 2012, pp. 2780–2788. URL: http://papers.nips.cc/paper/4533-perceptron-learning-of-sat (cit. on p. 121).

[Fed+20] William Fedus, Dibya Ghosh, John D. Martin, Marc G. Bellemare, Yoshua Bengio, and Hugo Larochelle. "On Catastrophic Interference in Atari 2600 Games". In: *CoRR* abs/2002.12499 (2020). arXiv: 2002.12499. URL: https://arxiv.org/abs/2002.12499 (cit. on pp. 23, 37).

[FL19] Matthias Fey and Jan Eric Lenssen. "Fast Graph Representation Learning with PyTorch Geometric". In: *CoRR* abs/1903.02428 (2019). arXiv: 1903.02428. URL: http://arxiv.org/abs/1903.02428 (cit. on p. 156).

[Flo+17] Carlos Florensa, David Held, Markus Wulfmeier, Michael Zhang, and Pieter Abbeel. "Reverse Curriculum Generation for Reinforcement Learning". In: *1st Annual Conference on Robot Learning, CoRL 2017*. Vol. 78. Proceedings of Machine Learning Research. PMLR, 2017, pp. 482–495. URL: http://proceedings.mlr.press/v78/florensa17a.html (cit. on pp. 34, 37).

[Flo+18] Carlos Florensa, David Held, Xinyang Geng, and Pieter Abbeel. "Automatic Goal Generation for Reinforcement Learning Agents". In: *International Conference on Machine Learning, ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1514–1523. URL: http://proceedings.mlr.press/v80/florensa18a.html (cit. on p. 34).

[FS00] Jörg Fliege and Benar Fux Svaiter. "Steepest descent methods for multicriteria optimization". In: *Math. Methods Oper. Res.* 51.3 (2000), pp. 479–494. URL: https://doi.org/10.1007/s001860000043 (cit. on p. 48).

[FvHM18] Scott Fujimoto, Herke van Hoof, and David Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *International Conference on Machine Learning, ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1582–1591. URL: http://proceedings.mlr.press/v80/fujimoto18a.html (cit. on pp. 13, 14).

[FW86] Philip J. Fleming and John J. Wallace. "How Not To Lie With Statistics: The Correct Way To Summarize Benchmark Results". In: *Commun. ACM* 29.3 (1986), pp. 218–221. URL: https://doi.org/10.1145/5666.5673 (cit. on p. 25).

[GB10] Xavier Glorot and Yoshua Bengio. "Understanding the difficulty of training deep feedforward neural networks". In: *International Conference on Artificial Intelligence and Statistics, AISTATS*. Vol. 9. JMLR Proceedings. JMLR.org, 2010, pp. 249–256. URL: http://proceedings.mlr.press/v9/glorot10a.html (cit. on p. 78).

[GLU20] Pengsheng Guo, Chen-Yu Lee, and Daniel Ulbricht. "Learning to Branch for Multi-Task Learning". In: *International Conference on Machine Learning, ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 3854–3863. URL: http://proceedings.mlr.press/v119/guo20e.html (cit. on p. 20).

[GMS05] Marco Gori, Gabriele Monfardini, and Franco Scarselli. "A new model for learning in graph domains". In: *IEEE International Joint Conference on Neural Networks*. Vol. 2. IEEE. 2005, pp. 729–734. URL: https://ieeexplore.ieee.org/document/1555942 (cit. on p. 106).

[God+22] Jonathan Godwin, Michael Schaarschmidt, Alexander L. Gaunt, Alvaro Sanchez-Gonzalez, Yulia Rubanova, Petar Velickovic, James Kirkpatrick, and Peter W. Battaglia. "Simple GNN Regularisation for 3D Molecular Property Prediction and Beyond". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=1wVvweK3oIb (cit. on p. 38).

[Gol19] Yoav Goldberg. "Assessing BERT's Syntactic Abilities". In: *CoRR* abs/1901.05287 (2019). arXiv: 1901.05287. URL: http://arxiv.org/abs/1901.05287 (cit. on p. 88).

[GP14] Cristian Grozea and Marius Popescu. "Can Machine Learning Learn a Decision Oracle for NP Problems? A Test on SAT". In: *Fundam. Informaticae* 131.3-4 (2014), pp. 441–450. URL: https://doi.org/10.3233/FI-2014-1024 (cit. on p. 121).

[Guo+18] Michelle Guo, Albert Haque, De-An Huang, Serena Yeung, and Li Fei-Fei. "Dynamic Task Prioritization for Multitask Learning". In: *European Conference on Computer Vision, ECCV*. Vol. 11220. Lecture Notes in Computer Science. Springer, 2018, pp. 282–299. URL: https://doi.org/10.1007/978-3-030-01270-0%5C_17 (cit. on p. 20).

[Gup+22] Agrim Gupta, Linxi Fan, Surya Ganguli, and Li Fei-Fei. "MetaMorph: Learning Universal Controllers with Transformers". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=Opmqtk%5C_GvYL (cit. on p. 32).

[Haa+18] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *International Conference on Machine Learning, ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 1856–1865. URL: http://proceedings.mlr.press/v80/haarnoja18b.html (cit. on pp. 14, 57).

[Ham+21] Eric Hambro et al. "Insights From the NeurIPS 2021 NetHack Challenge". In: *NeurIPS Competitions and Demonstrations Track*. Vol. 176. Proceedings of Machine Learning Research. PMLR, 2021, pp. 41–52. URL: https://proceedings.mlr.press/v176/hambro22a.html (cit. on p. 43).

[Ham20] William L. Hamilton. *Graph Representation Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers, 2020. URL: https://doi.org/10.2200/S01045ED1V01Y202009AIM046 (cit. on pp. 75, 84).

[Han06] Nikolaus Hansen. "The CMA Evolution Strategy: A Comparing Review". In: *Towards a New Evolutionary Computation - Advances in the Estimation of Distribution Algorithms*. Vol. 192. Studies in Fuzziness and Soft Computing. Springer, 2006, pp. 75–102. URL: https://doi.org/10.1007/3-540-32494-1%5C_4 (cit. on p. 32).

[HCM15] Assaf Hallak, Dotan Di Castro, and Shie Mannor. "Contextual Markov Decision Processes". In: *CoRR* abs/1502.02259 (2015). arXiv: 1502.02259. URL: http://arxiv.org/abs/1502.02259 (cit. on p. 35).

[He+16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE Computer Society, 2016, pp. 770–778. URL: https://doi.org/10.1109/CVPR.2016.90 (cit. on p. 133).

[Hee+17] Nicolas Heess, Dhruva TB, Srinivasan Sriram, Jay Lemmon, Josh Merel, Greg Wayne, Yuval Tassa, Tom Erez, Ziyu Wang, S. M. Ali Eslami, Martin A. Riedmiller, and David Silver. "Emergence of Locomotion Behaviours in Rich Environments". In: *CoRR* abs/1707.02286 (2017). arXiv: 1707.02286. URL: http://arxiv.org/abs/1707.02286 (cit. on p. 12).

[Hen+17] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. "Deep Reinforcement Learning that Matters". In: *CoRR* abs/1709.06560 (2017). arXiv: 1709.06560. URL: http://arxiv.org/abs/1709.06560 (cit. on p. 14).

[Hes+18] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Tom Schaul, Georg Ostrovski, Will Dabney, Dan Horgan, Bilal Piot, Mohammad Gheshlaghi Azar, and David Silver. "Rainbow: Combining Improvements in Deep Reinforcement Learning". In: *Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 3215–3222. URL: https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17204 (cit. on pp. 112, 124).

[Hes+19] Matteo Hessel, Hubert Soyer, Lasse Espeholt, Wojciech Czarnecki, Simon Schmitt, and Hado van Hasselt. "Multi-Task Deep Reinforcement Learning with PopArt". In: *The Thirty-Third AAAI Conference on Artificial Intelligence*. AAAI Press, 2019, pp. 3796–3803. URL: https://doi.org/10.1609/aaai.v33i01.33013796 (cit. on pp. 23, 39).

[Hes00] Tom Heskes. "Empirical Bayes for Learning to Learn". In: *International Conference on Machine Learning ICML*. Morgan Kaufmann, 2000, pp. 367–374. URL: https://people.eecs.berkeley.edu/~russell/classes/cs294/f05/papers/heskes-2000.pdf (cit. on p. 46).

[HM20] Jessica B. Hamrick and Shakir Mohamed. "Levels of Analysis for Machine Learning". In: *CoRR* abs/2004.05107 (2020). arXiv: 2004.05107. URL: https://arxiv.org/abs/2004.05107 (cit. on p. 39).

[HMP20] Wenlong Huang, Igor Mordatch, and Deepak Pathak. "One Policy to Control Them All: Shared Modular Policies for Agent-Agnostic Control". In: *International Conference on Machine Learning, ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 4455–4464.

URL: http://proceedings.mlr.press/v119/huang20d.html (cit. on pp. 5, 23, 32, 38, 40–42, 66, 68, 73, 87–89, 91, 93–95, 98).

[Hos+22] Timothy M. Hospedales, Antreas Antoniou, Paul Micaelli, and Amos J. Storkey. "Meta-Learning in Neural Networks: A Survey". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 44.9 (2022), pp. 5149–5169. URL: https://doi.org/10.1109/TPAMI.2021.3079209 (cit. on pp. 21, 33, 66).

[Hou+19] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. "Parameter-Efficient Transfer Learning for NLP". In: *International Conference on Machine Learning ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2790–2799. URL: http://proceedings.mlr.press/v97/houlsby19a/houlsby19a.pdf (cit. on p. 84).

[HS00] Holger H Hoos and Thomas Stützle. "SATLIB: An online resource for research on SAT". In: *Sat* 2000 (2000). accessed September 18, 2019, pp. 283–292. URL: https://www.cs.ubc.ca/~hoos/SATLIB/benchm.html (cit. on pp. 110, 116).

[Hu+20] Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay S. Pande, and Jure Leskovec. "Strategies for Pre-training Graph Neural Networks". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=HJlWWJSFDH (cit. on p. 38).

[Hum+22] Peter C. Humphreys, David Raposo, Tobias Pohlen, Gregory Thornton, Rachita Chhaparia, Alistair Muldal, Josh Abramson, Petko Georgiev, Adam Santoro, and Timothy P. Lillicrap. "A data-driven approach for learning to control computers". In: *International Conference on Machine Learning, ICML*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 9466–9482. URL: https://proceedings.mlr.press/v162/humphreys22a.html (cit. on p. 26).

[HVD15] Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. "Distilling the Knowledge in a Neural Network". In: *CoRR* abs/1503.02531 (2015). arXiv: 1503.02531. URL: http://arxiv.org/abs/1503.02531 (cit. on p. 25).

[HvM09] Marijn Heule and Hans van Maaren. "Look-Ahead Based SAT Solvers". In: *Handbook of Satisfiability*. Vol. 185. Frontiers in Artificial Intelligence and Applications. IOS Press, 2009, pp. 155–184. URL: https://doi.org/10.3233/978-1-58603-929-5-155 (cit. on p. 122).

[HW09] Shai Haim and Toby Walsh. "Restart Strategy Selection Using Machine Learning Techniques". In: *Theory and Applications of Satisfiability Testing - SAT*. Vol. 5584. Lecture Notes in Computer Science. Springer, 2009, pp. 312–325. URL: https://doi.org/10.1007/978-3-642-02777-2%5C_30 (cit. on p. 121).

[HYK22] Sunghoon Hong, Deunsol Yoon, and Kee-Eung Kim. "Structure-Aware Transformer Policy for Inhomogeneous Multi-Task Reinforcement Learning". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=fy%5C_XRVHqly (cit. on p. 32).

[Iba+22] Borja Ibarz, Vitaly Kurin, George Papamakarios, Kyriacos Nikiforou, Mehdi Bennani, Róbert Csordás, Andrew Dudzik, Matko Bosnjak, Alex Vitvitskyi, Yulia Rubanova, Andreea Deac, Beatrice Bevilacqua, Yaroslav Ganin, Charles Blundell, and Petar Velickovic. "A Generalist Neural Algorithmic Learner". In: *CoRR* abs/2209.11142 (2022). arXiv: 2209.11142. URL: https://doi.org/10.48550/arXiv.2209.11142 (cit. on pp. 23, 24, 27, 129).

[Igl+19] Maximilian Igl, Kamil Ciosek, Yingzhen Li, Sebastian Tschiatschek, Cheng Zhang, Sam Devlin, and Katja Hofmann. "Generalization in Reinforcement Learning with Selective Noise Injection and Information Bottleneck". In: *Advances in Neural Information Processing Systems*. 2019, pp. 13956–13968. URL: https://proceedings.neurips.cc/paper/2019/hash/e2ccf95a7f2e1878fcafc8376649b6e8-Abstract.html (cit. on p. 28).

[Igl+21] Maximilian Igl, Gregory Farquhar, Jelena Luketina, Wendelin Boehmer, and Shimon Whiteson. "Transient Non-stationarity and Generalisation in Deep Reinforcement Learning". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=Qun8fv4qSby (cit. on p. 38).

[Iqb+20] Shariq Iqbal, Christian A. Schröder de Witt, Bei Peng, Wendelin Böhmer, Shimon Whiteson, and Fei Sha. "AI-QMIX: Attention and Imagination for Dynamic Multi-Agent Reinforcement Learning". In: *CoRR* abs/2006.04222 (2020). arXiv: 2006.04222. URL: https://arxiv.org/abs/2006.04222 (cit. on pp. 41, 44).

[IS19] Shariq Iqbal and Fei Sha. "Actor-Attention-Critic for Multi-Agent Reinforcement Learning". In: *International Conference on Machine Learning, ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 2961–2970. URL: http://proceedings.mlr.press/v97/iqbal19a.html (cit. on p. 41).

[Jad+17] Max Jaderberg, Volodymyr Mnih, Wojciech Marian Czarnecki, Tom Schaul, Joel Z. Leibo, David Silver, and Koray Kavukcuoglu. "Reinforcement Learning with Unsupervised Auxiliary Tasks". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJ6yPD5xg (cit. on p. 38).

[Jam+19] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. "Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 2019, pp. 12627–12637. URL: http://openaccess.thecvf.com/content%5C_CVPR%5C_2019/html/James%5C_Sim-To-Real%5C_via%5C_Sim-To-Sim%5C_Data-Efficient%5C_Robotic%5C_Grasping%5C_via%5C_Randomized-To-Canonical%5C_Adaptation%5C_Networks%5C_CVPR%5C_2019%5C_paper.html (cit. on p. 29).

[Jay03] Edwin T Jaynes. *Probability theory: The logic of science*. Cambridge university press, 2003 (cit. on p. 127).

[JDL18] Jiechuan Jiang, Chen Dun, and Zongqing Lu. "Graph Convolutional Reinforcement Learning for Multi-Agent Cooperation". In: *CoRR* abs/1810.09202 (2018). arXiv: 1810.09202. URL: http://arxiv.org/abs/1810.09202 (cit. on p. 44).

[JLM20] Sebastian Jaszczur, Michal Luszczyk, and Henryk Michalewski. "Neural heuristics for SAT solving". In: *CoRR* abs/2005.13406 (2020). arXiv: 2005.13406. URL: https://arxiv.org/abs/2005.13406 (cit. on p. 121).

[JLT20] Daniel D. Johnson, Hugo Larochelle, and Daniel Tarlow. "Learning Graph Structure With A Finite-State Automaton Layer". In: *Advances in Neural Information Processing Systems*. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/1fdc0ee9d95c71d73df82ac8f0721459-Abstract.html (cit. on p. 42).

[Jos22] Chaitanya K. Joshi. *Recent Advances in Efficient and Scalable Graph Neural Networks*. [Online; accessed 16-February-2023]. 2022. URL: https://www.chaitjo.com/post/efficient-gnns/ (cit. on pp. 68, 124).

[JS97] Roberto J. Bayardo Jr. and Robert Schrag. "Using CSP Look-Back Techniques to Solve Real-World SAT Instances". In: *Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*. AAAI Press / The MIT Press, 1997, pp. 203–208. URL:

http://www.aaai.org/Library/AAAI/1997/aaai97-032.php (cit. on p. 106).

[JT17] Vikas Jain and Theja Tulabandhula. "Faster Reinforcement Learning Using Active Simulators". In: *CoRR* abs/1703.07853 (2017). arXiv: 1703.07853. URL: http://arxiv.org/abs/1703.07853 (cit. on p. 36).

[Jul+18] Arthur Juliani, Vincent-Pierre Berges, Esh Vckay, Yuan Gao, Hunter Henry, Marwan Mattar, and Danny Lange. "Unity: A General Platform for Intelligent Agents". In: *CoRR* abs/1809.02627 (2018). arXiv: 1809.02627. URL: http://arxiv.org/abs/1809.02627 (cit. on p. 43).

[Jul+20] Ryan Julian, Benjamin Swanson, Gaurav S. Sukhatme, Sergey Levine, Chelsea Finn, and Karol Hausman. "Efficient Adaptation for End-to-End Vision-Based Robotic Manipulation". In: *CoRR* abs/2004.10190 (2020). arXiv: 2004.10190. URL: https://arxiv.org/abs/2004.10190 (cit. on p. 32).

[JV22] Adrián Javaloy and Isabel Valera. "RotoGrad: Gradient Homogenization in Multitask Learning". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2022. URL: https://openreview.net/forum?id=T8wHz4rnuGL (cit. on pp. 20, 133, 134).

[Kak01a] Sham M. Kakade. "A Natural Policy Gradient". In: *Advances in Neural Information Processing Systems*. MIT Press, 2001, pp. 1531–1538. URL: https://proceedings.neurips.cc/paper/2001/hash/4b86abe48d358ecf194c56c69108433e-Abstract.html (cit. on p. 26).

[Kak01b] Sham M. Kakade. "A Natural Policy Gradient". In: *Advances in Neural Information Processing Systems*. MIT Press, 2001, pp. 1531–1538. URL: https://proceedings.neurips.cc/paper/2001/hash/4b86abe48d358ecf194c56c69108433e-Abstract.html (cit. on p. 69).

[Kal+21] Dmitry Kalashnikov, Jacob Varley, Yevgen Chebotar, Benjamin Swanson, Rico Jonschkowski, Chelsea Finn, Sergey Levine, and Karol Hausman. "MT-Opt: Continuous Multi-Task Robotic Reinforcement Learning at Scale". In: *CoRR* abs/2104.08212 (2021). arXiv: 2104.08212. URL: https://arxiv.org/abs/2104.08212 (cit. on pp. 27, 46).

[Kar72] Richard M. Karp. "Reducibility Among Combinatorial Problems". In: *Symposium on the Complexity of Computer Computations*. The IBM Research Symposia Series. Plenum Press, New York, 1972, pp. 85–103. URL: https://doi.org/10.1007/978-1-4684-2001-2%5C_9 (cit. on pp. 6, 106).

[Kaz+20] Anees Kazi, Luca Cosmo, Nassir Navab, and Michael M. Bronstein. "Differentiable Graph Module (DGM) Graph Convolutional Networks". In: *CoRR* abs/2002.04999 (2020). arXiv: 2002.04999. URL: https://arxiv.org/abs/2002.04999 (cit. on p. 41).

[KB06] George Dimitri Konidaris and Andrew G. Barto. "Autonomous shaping: knowledge transfer in reinforcement learning". In: *International Conference on Machine Learning ICML*. Vol. 148. ACM International Conference Proceeding Series. ACM, 2006, pp. 489–496. URL: https://doi.org/10.1145/1143844.1143906 (cit. on p. 31).

[KB15] Diederik P. Kingma and Jimmy Ba. "Adam: A Method for Stochastic Optimization". In: *International Conference on Learning Representations, ICLR*. 2015. URL: http://arxiv.org/abs/1412.6980 (cit. on p. 132).

[KB17] Tom Kocmi and Ondrej Bojar. "An Exploration of Word Embedding Initialization in Deep-Learning Tasks". In: *International Conference on Natural Language Processing, ICON*. NLP Association of India, 2017, pp. 56–64. URL: https://aclanthology.org/W17-7508/ (cit. on p. 83).

[Kes+17] Nitish Shirish Keskar, Dheevatsa Mudigere, Jorge Nocedal, Mikhail Smelyanskiy, and Ping Tak Peter Tang. "On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=H1oyRlYgg (cit. on pp. 62, 64, 65).

[KGC18] Alex Kendall, Yarin Gal, and Roberto Cipolla. "Multi-Task Learning Using Uncertainty to Weigh Losses for Scene Geometry and Semantics". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 2018, pp. 7482–7491. URL: http://openaccess.thecvf.com/content%5C_cvpr%5C_2018/html/Kendall%5C_Multi-Task%5C_Learning%5C_Using%5C_CVPR%5C_2018%5C_paper.html (cit. on pp. 20, 46, 56).

[Kha+17] Elias B. Khalil, Hanjun Dai, Yuyu Zhang, Bistra Dilkina, and Le Song. "Learning Combinatorial Optimization Algorithms over Graphs". In: *Advances in Neural Information Processing Systems*. 2017, pp. 6348–6358. URL: https://proceedings.neurips.cc/paper/2017/hash/d9896106ca98d3d05b8cbdf4fd8b13a1-Abstract.html (cit. on pp. 30, 40, 43, 123).

[Khe+20a] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. "Towards Continual Reinforcement Learning: A Review and Perspectives". In: *CoRR* abs/2012.13490 (2020). arXiv: 2012.13490. URL: https://arxiv.org/abs/2012.13490 (cit. on pp. 21, 66).

[Khe+20b] Khimya Khetarpal, Matthew Riemer, Irina Rish, and Doina Precup. "Towards Continual Reinforcement Learning: A Review and Perspectives". In: *CoRR* abs/2012.13490 (2020). arXiv: 2012.13490. URL: https://arxiv.org/abs/2012.13490 (cit. on p. 37).

[Kip+18] Thomas N. Kipf, Ethan Fetaya, Kuan-Chieh Wang, Max Welling, and Richard S. Zemel. "Neural Relational Inference for Interacting Systems". In: *International Conference on Machine Learning, ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 2693–2702. URL: http://proceedings.mlr.press/v80/kipf18a.html (cit. on p. 42).

[Kir+16] James Kirkpatrick, 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. "Overcoming catastrophic forgetting in neural networks". In: *CoRR* abs/1612.00796 (2016). arXiv: 1612.00796. URL: http://arxiv.org/abs/1612.00796 (cit. on pp. 26, 37).

[Kir+21] Robert Kirk, Amy Zhang, Edward Grefenstette, and Tim Rocktäschel. "A Survey of Generalisation in Deep Reinforcement Learning". In: *CoRR* abs/2111.09794 (2021). arXiv: 2111.09794. URL: https://arxiv.org/abs/2111.09794 (cit. on p. 30).

[KJV83] Scott Kirkpatrick, D. Gelatt Jr., and Mario P. Vecchi. "Optimization by Simmulated Annealing". In: *Sci.* 220.4598 (1983), pp. 671–680. URL: https://doi.org/10.1126/science.220.4598.671 (cit. on p. 123).

[Kli+20] Pascal Klink, Carlo D'Eramo, Jan Peters, and Joni Pajarinen. "Self-Paced Deep Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/68a9750337a418a86fe06c1991a1d64c-Abstract.html (cit. on p. 35).

[KLY18] Robert Kleinberg, Yuanzhi Li, and Yang Yuan. "An Alternative View: When Does SGD Escape Local Minima?" In: *International Conference on Machine Learning, ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 2703–2712. URL: http://proceedings.mlr.press/v80/kleinberg18a.html (cit. on pp. 62, 64, 65).

[KNS22] Raiyan Khan, Thanh V Nguyen, and Srinivasan H Sengamedu. "Hyperbolic representations of source code". In: (2022) (cit. on p. 127).

[Kok17] Iasonas Kokkinos. "UberNet: Training a Universal Convolutional Neural Network for Low-, Mid-, and High-Level Vision Using Diverse Datasets and Limited Memory". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE Computer Society, 2017, pp. 5454–5463. URL: https://doi.org/10.1109/CVPR.2017.579 (cit. on p. 46).

[KSS11] Hadi Katebi, Karem A. Sakallah, and João P. Marques Silva. "Empirical Study of the Anatomy of Modern Sat Solvers". In: *Theory and Applications of Satisfiability Testing - SAT*. Vol. 6695. Lecture Notes in Computer Science. Springer, 2011, pp. 343–356. URL: https://doi.org/10.1007/978-3-642-21581-0%5C_27 (cit. on p. 121).

[Kur+20] Vitaly Kurin, Saad Godil, Shimon Whiteson, and Bryan Catanzaro. "Can Q-Learning with Graph Networks Learn a Generalizable Branching Heuristic for a SAT Solver?" In: *Advances in Neural Information Processing Systems*. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/6d70cb65d15211726dcce4c0e971e21c-Abstract.html (cit. on pp. 30, 40, 43, 66, 107).

[Kur+21] Vitaly Kurin, Maximilian Igl, Tim Rocktäschel, Wendelin Boehmer, and Shimon Whiteson. "My Body is a Cage: the Role of Morphology in Graph-Based Incompatible Control". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=N3zUDGN5lO (cit. on pp. 66, 89).

[Kur+22] Vitaly Kurin, Alessandro De Palma, Ilya Kostrikov, Shimon Whiteson, and M. Pawan Kumar. "In Defense of the Unitary Scalarization for Deep Multi-Task Learning". In: *Advances in Neural Information Processing Systems*. 2022. URL: https://arxiv.org/abs/2201.04122 (cit. on p. 47).

[Küt+20] Heinrich Küttler, Nantas Nardelli, Alexander H. Miller, Roberta Raileanu, Marco Selvatici, Edward Grefenstette, and Tim Rocktäschel. "The NetHack Learning Environment". In: *Advances in Neural Information Processing Systems*. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/569ff987c643b4bedf504efda8f786c2-Abstract.html (cit. on pp. 30, 43).

[KW14] Diederik P. Kingma and Max Welling. "Auto-Encoding Variational Bayes". In: *International Conference on Learning Representations, ICLR*. 2014. URL: http://arxiv.org/abs/1312.6114 (cit. on pp. 14, 42).

[KW17] Thomas N. Kipf and Max Welling. "Semi-Supervised Classification with Graph Convolutional Networks". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJU4ayYgl (cit. on pp. 39, 41).

[LeC+89] Yann LeCun, Bernhard E. Boser, John S. Denker, Donnie Henderson, Richard E. Howard, Wayne E. Hubbard, and Lawrence D. Jackel. "Backpropagation Applied to Handwritten Zip Code Recognition". In: *Neural Comput.* 1.4 (1989), pp. 541–551. URL: https://doi.org/10.1162/neco.1989.1.4.541 (cit. on p. 28).

[LeC+98] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. "Gradient-based learning applied to document recognition". In: *Proc. IEEE* 86.11 (1998), pp. 2278–2324. URL: https://doi.org/10.1109/5.726791 (cit. on p. 132).

[Led+20] Gil Lederman, Markus N. Rabe, Sanjit Seshia, and Edward A. Lee. "Learning Heuristics for Quantified Boolean Formulas through Reinforcement Learning". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=BJluxREKDB (cit. on p. 122).

[Lee+20] Kimin Lee, Kibok Lee, Jinwoo Shin, and Honglak Lee. "Network Randomization: A Simple Technique for Generalization in Deep Reinforcement Learning". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=HJgcvJBFvB (cit. on p. 28).

[LG10] Alessandro Lazaric and Mohammad Ghavamzadeh. "Bayesian Multi-Task Reinforcement Learning". In: *International Conference on Machine Learning ICML*. Omnipress, 2010, pp. 599–606. URL: https://icml.cc/Conferences/2010/papers/269.pdf (cit. on p. 22).

[Li+16] Yujia Li, Daniel Tarlow, Marc Brockschmidt, and Richard S. Zemel. "Gated Graph Sequence Neural Networks". In: *International Conference on Learning Representations, ICLR*. 2016. URL: http://arxiv.org/abs/1511.05493 (cit. on p. 70).

[Li+20] Sheng Li, Jayesh K. Gupta, Peter Morales, Ross E. Allen, and Mykel J. Kochenderfer. "Deep Implicit Coordination Graphs for Multi-agent Reinforcement Learning". In: *CoRR* abs/2006.11438 (2020). arXiv: 2006.11438. URL: https://arxiv.org/abs/2006.11438 (cit. on p. 41).

[Lia+15] Jia Hui Liang, Vijay Ganesh, Ed Zulkoski, Atulan Zaman, and Krzysztof Czarnecki. "Understanding VSIDS Branching Heuristics in Conflict-Driven Clause-Learning SAT Solvers". In: *International Haifa Verification Conference, HVC*. Vol. 9434. Lecture Notes in Computer Science. Springer, 2015, pp. 225–241. URL: https://doi.org/10.1007/978-3-319-26287-1%5C_14 (cit. on p. 108).

[Lia+16] Jia Hui Liang, Vijay Ganesh, Pascal Poupart, and Krzysztof Czarnecki. "Learning Rate Based Branching Heuristic for SAT Solvers". In: *Theory and Applications of Satisfiability Testing - SAT*. Vol. 9710. Lecture Notes in Computer Science. Springer, 2016, pp. 123–140. URL: https://doi.org/10.1007/978-3-319-40970-2%5C_9 (cit. on pp. 121, 122).

[Lil+16] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. "Continuous control with deep reinforcement learning". In: *International Conference on Learning Representations, ICLR*. 2016. URL: http://arxiv.org/abs/1509.02971 (cit. on p. 13).

[Lin+19] Xingyu Lin, Harjatin Singh Baweja, George Kantor, and David Held. "Adaptive Auxiliary Task Weighting for Reinforcement Learning". In: *Advances in Neural Information Processing Systems*. 2019, pp. 4773–4784. URL: https://proceedings.neurips.cc/paper/2019/hash/0e900ad84f63618452210ab8baae0218-Abstract.html (cit. on p. 38).

[Liu+15] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Attributes in the Wild". In: *IEEE International Conference on Computer Vision, ICCV*. IEEE Computer Society, 2015, pp. 3730–3738. URL: https://doi.org/10.1109/ICCV.2015.425 (cit. on pp. 53, 65, 133, 139, 143, 144).

[Liu+18] Evan Zheran Liu, Kelvin Guu, Panupong Pasupat, Tianlin Shi, and Percy Liang. "Reinforcement Learning on Web Interfaces using Workflow-Guided Exploration". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=ryTp3f-0- (cit. on p. 27).

[Liu+21a] Bo Liu, Xingchao Liu, Xiaojie Jin, Peter Stone, and Qiang Liu. "Conflict-Averse Gradient Descent for Multi-task learning". In: *Advances in Neural Information Processing Systems*. 2021, pp. 18878–18890. URL: https://proceedings.neurips.cc/paper/2021/hash/9d27fdf2477ffbff837d73ef7ae23db9-Abstract.html (cit. on pp. 20, 133, 134).

[Liu+21b] Liyang Liu, Yi Li, Zhanghui Kuang, Jing-Hao Xue, Yimin Chen, Wenming Yang, Qingmin Liao, and Wayne Zhang. "Towards Impartial Multi-task Learning". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=IMPnRXEWpvr (cit. on pp. 20, 46, 47, 49, 50, 52–54, 56, 63, 133, 146, 160–162).

[Liu+21c] Zhuang Liu, Xuanlin Li, Bingyi Kang, and Trevor Darrell. "Regularization Matters in Policy Optimization - An Empirical Study on Continuous Control". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=yr1mzrH3IC (cit. on p. 57).

[LJD19] Shikun Liu, Edward Johns, and Andrew J. Davison. "End-To-End Multi-Task Learning With Attention". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. 2019, pp. 1871–1880. URL: http://openaccess.thecvf.com/content%5C_CVPR%5C_2019/html/Liu%5C_End-To-End%5C_Multi-Task%5C_Learning%5C_With%5C_Attention%5C_CVPR%5C_2019%5C_paper.html (cit. on pp. 20, 133, 137).

[LLC09] Hui Li, Xuejun Liao, and Lawrence Carin. "Multi-task Reinforcement Learning in Partially Observable Stochastic Environments". In: *J. Mach. Learn. Res.* 10 (2009), pp. 1131–1186. URL: https://dl.acm.org/doi/10.5555/1577069.1577109 (cit. on p. 22).

[LO12] Manuel Lopes and Pierre-Yves Oudeyer. "The strategic student approach for life-long exploration and learning". In: *IEEE International Conference on Development and Learning and Epigenetic Robotics*. IEEE, 2012, pp. 1–8. URL: https://doi.org/10.1109/DevLrn.2012.6400807 (cit. on p. 36).

[Loy+20] Ricky Loynd, Roland Fernandez, Asli Çelikyilmaz, Adith Swaminathan, and Matthew J. Hausknecht. "Working Memory Graphs". In: *International Conference on Machine Learning, ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 6404–6414. URL: http://proceedings.mlr.press/v119/loynd20a.html (cit. on p. 41).

[LPK22] Xingyu Liu, Deepak Pathak, and Kris Kitani. "REvolveR: Continuous Evolutionary Models for Robot-to-robot Policy Transfer". In: *International Conference on Machine Learning, ICML*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 13995–14007. URL: https://proceedings.mlr.press/v162/liu22p.html (cit. on p. 35).

[LSO20] Mingchen Li, Mahdi Soltanolkotabi, and Samet Oymak. "Gradient Descent with Early Stopping is Provably Robust to Label Noise for Overparameterized Neural Networks". In: *International Conference on Artificial Intelligence and Statistics, AISTATS*. Vol. 108. Proceedings of Machine Learning Research. PMLR, 2020, pp. 4313–4324. URL: http://proceedings.mlr.press/v108/li20j.html (cit. on p. 62).

[Lu+22] Kevin Lu, Aditya Grover, Pieter Abbeel, and Igor Mordatch. "Frozen Pretrained Transformers as Universal Computation Engines". In: *AAAI Conference on Artificial Intelligence, AAAI*. AAAI Press, 2022,

pp. 7628–7636. URL: https://ojs.aaai.org/index.php/AAAI/article/view/20729 (cit. on p. 84).

[Lyl+21] Clare Lyle, Mark Rowland, Georg Ostrovski, and Will Dabney. "On the Effect of Auxiliary Tasks on Representation Dynamics". In: *International Conference on Artificial Intelligence and Statistics, AISTATS*. Vol. 130. Proceedings of Machine Learning Research. PMLR, 2021, pp. 1–9. URL: http://proceedings.mlr.press/v130/lyle21a.html (cit. on p. 39).

[LYZ21] Baijiong Lin, Feiyang Ye, and Yu Zhang. "A Closer Look at Loss Weighting in Multi-Task Learning". In: *CoRR* abs/2111.10603 (2021). arXiv: 2111.10603. URL: https://arxiv.org/abs/2111.10603 (cit. on pp. 20, 51–53, 56, 62–64, 133, 137).

[Mal+18] Aleksandra Malysheva, Tegg Tae Kyong Sung, Chae-Bong Sohn, Daniel Kudenko, and Aleksei Shpilman. "Deep Multi-Agent Reinforcement Learning with Relevance Graphs". In: *CoRR* abs/1811.12557 (2018). arXiv: 1811.12557. URL: http://arxiv.org/abs/1811.12557 (cit. on p. 44).

[Mat+20] Tambet Matiisen, Avital Oliver, Taco Cohen, and John Schulman. "Teacher-Student Curriculum Learning". In: *IEEE Trans. Neural Networks Learn. Syst.* 31.9 (2020), pp. 3732–3740. URL: https://doi.org/10.1109/TNNLS.2019.2934906 (cit. on pp. 36, 37).

[MBB18] Siyuan Ma, Raef Bassily, and Mikhail Belkin. "The Power of Interpolation: Understanding the Effectiveness of SGD in Modern Over-parametrized Learning". In: *International Conference on Machine Learning, ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 3331–3340. URL: http://proceedings.mlr.press/v80/ma18a.html (cit. on p. 165).

[MC89] Michael McCloskey and Neal J Cohen. "Catastrophic interference in connectionist networks: The sequential learning problem". In: *Psychology of learning and motivation*. Vol. 24. Elsevier, 1989, pp. 109–165. URL: https://www.sciencedirect.com/science/article/abs/pii/S0079742108605368 (cit. on pp. 22, 37).

[Meh+19] Bhairav Mehta, Manfred Diaz, Florian Golemo, Christopher J. Pal, and Liam Paull. "Active Domain Randomization". In: *3rd Annual Conference on Robot Learning, CoRL*. Vol. 100. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1162–1176. URL: http://proceedings.mlr.press/v100/mehta20a.html (cit. on p. 29).

[Mik+13] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. "Efficient Estimation of Word Representations in Vector Space". In: *International Conference on Learning Representations, ICLR, Workshop Track Proceedings*. 2013. URL: http://arxiv.org/abs/1301.3781 (cit. on p. 83).

[Mir+17] Piotr Mirowski, Razvan Pascanu, Fabio Viola, Hubert Soyer, Andy Ballard, Andrea Banino, Misha Denil, Ross Goroshin, Laurent Sifre, Koray Kavukcuoglu, Dharshan Kumaran, and Raia Hadsell. "Learning to Navigate in Complex Environments". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SJMGPrcle (cit. on p. 38).

[Mis+16] Ishan Misra, Abhinav Shrivastava, Abhinav Gupta, and Martial Hebert. "Cross-Stitch Networks for Multi-task Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE Computer Society, 2016, pp. 3994–4003. URL: https://doi.org/10.1109/CVPR.2016.433 (cit. on p. 20).

[MLT15] Igor Mordatch, Kendall Lowrey, and Emanuel Todorov. "Ensemble-CIO: Full-body dynamic motion planning that transfers to physical humanoids". In: *International Conference on Intelligent Robots and Systems, IROS*. IEEE, 2015, pp. 5307–5314. URL: https://doi.org/10.1109/IROS.2015.7354126 (cit. on p. 29).

[Mni+15] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. "Human-level control through deep reinforcement learning". In: *Nat.* 518.7540 (2015), pp. 529–533. URL: https://doi.org/10.1038/nature14236 (cit. on pp. 12, 106).

[Mni+16] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. "Asynchronous Methods for Deep Reinforcement Learning". In: *International Conference on Machine Learning, ICML*. Vol. 48. JMLR Workshop and Conference Proceedings. JMLR.org, 2016, pp. 1928–1937. URL: http://proceedings.mlr.press/v48/mniha16.html (cit. on p. 26).

[Mol+18] Artem Molchanov, Karol Hausman, Stan Birchfield, and Gaurav S. Sukhatme. "Region Growing Curriculum Generation for Reinforcement Learning". In: *CoRR* abs/1807.01425 (2018). arXiv: 1807.01425. URL: http://arxiv.org/abs/1807.01425 (cit. on p. 34).

[Mos+01] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. "Chaff: Engineering an Efficient SAT Solver". In: *Design Automation Conference, DAC*. ACM, 2001, pp. 530–535. URL: https://doi.org/10.1145/378239.379017 (cit. on pp. 106–108).

[MPS] Siddharth Mysore, Robert Platt, and Kate Saenko. "Reward-guided Curriculum for Robust Reinforcement Learning". In: (). URL: https://sidmysore.github.io/publication/2019-01-23-RewardGuidedCurriculum-RL-1 (cit. on p. 36).

[Nar+20a] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. "Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey". In: *J. Mach. Learn. Res.* 21 (2020), 181:1–181:50. URL: http://jmlr.org/papers/v21/20-212.html (cit. on pp. 21, 66).

[Nar+20b] Sanmit Narvekar, Bei Peng, Matteo Leonetti, Jivko Sinapov, Matthew E. Taylor, and Peter Stone. "Curriculum Learning for Reinforcement Learning Domains: A Framework and Survey". In: *J. Mach. Learn. Res.* 21 (2020), 181:1–181:50. URL: http://jmlr.org/papers/v21/20-212.html (cit. on p. 34).

[Nas+20] Roula Nassif, Stefan Vlaski, Cédric Richard, Jie Chen, and Ali H. Sayed. "Multitask Learning Over Graphs: An Approach for Distributed, Streaming Machine Learning". In: *IEEE Signal Process. Mag.* 37.3 (2020), pp. 14–25. URL: https://doi.org/10.1109/MSP.2020.2966273 (cit. on p. 128).

[Nav+22] Aviv Navon, Aviv Shamsian, Idan Achituve, Haggai Maron, Kenji Kawaguchi, Gal Chechik, and Ethan Fetaya. "Multi-Task Learning as a Bargaining Game". In: *International Conference on Machine Learning, ICML*. Vol. 162. Proceedings of Machine Learning Research. PMLR, 2022, pp. 16428–16446. URL: https://proceedings.mlr.press/v162/navon22a.html (cit. on pp. 20, 62, 133, 134).

[New+14] Zack Newsham, Vijay Ganesh, Sebastian Fischmeister, Gilles Audemard, and Laurent Simon. "Impact of Community Structure on SAT Solver Performance". In: *Theory and Applications of Satisfiability Testing - SAT, Held as Part of the Vienna Summer of Logic, VSL*. Vol. 8561. Lecture Notes in Computer Science. Springer, 2014, pp. 252–268. URL: https://doi.org/10.1007/978-3-319-09284-3%5C_20 (cit. on p. 124).

[NSS17] Sanmit Narvekar, Jivko Sinapov, and Peter Stone. "Autonomous Task Sequencing for Customized Curriculum Design in Reinforcement Learning". In: *International Joint Conference on Artificial Intelligence, IJCAI*.

ijcai.org, 2017, pp. 2536–2542. URL: https://doi.org/10.24963/ijcai.2017/353 (cit. on p. 36).

[NT20] Chris Nota and Philip S. Thomas. "Is the Policy Gradient a Gradient?" In: *International Conference on Autonomous Agents and Multiagent Systems, AAMAS*. International Foundation for Autonomous Agents and Multiagent Systems, 2020, pp. 939–947. URL: https://dl.acm.org/doi/10.5555/3398761.3398871 (cit. on p. 12).

[Oh+20] Junhyuk Oh, Matteo Hessel, Wojciech M. Czarnecki, Zhongwen Xu, Hado van Hasselt, Satinder Singh, and David Silver. "Discovering Reinforcement Learning Algorithms". In: *Advances in Neural Information Processing Systems*. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/0b96d81f0494fde5428c7aea243c9157-Abstract.html (cit. on p. 33).

[Ope+19] OpenAI, Ilge Akkaya, Marcin Andrychowicz, Maciek Chociej, Mateusz Litwin, Bob McGrew, Arthur Petron, Alex Paino, Matthias Plappert, Glenn Powell, Raphael Ribas, Jonas Schneider, Nikolas Tezak, Jerry Tworek, Peter Welinder, Lilian Weng, Qiming Yuan, Wojciech Zaremba, and Lei Zhang. "Solving Rubik's Cube with a Robot Hand". In: *CoRR* abs/1910.07113 (2019). arXiv: 1910.07113. URL: http://arxiv.org/abs/1910.07113 (cit. on p. 29).

[OSC09] Olga Ohrimenko, Peter J. Stuckey, and Michael Codish. "Propagation via lazy clause generation". In: *Constraints An Int. J.* 14.3 (2009), pp. 357–391. URL: https://doi.org/10.1007/s10601-008-9064-x (cit. on p. 106).

[Pal+20] Aditya Paliwal, Sarah M. Loos, Markus N. Rabe, Kshitij Bansal, and Christian Szegedy. "Graph Representations for Higher-Order Logic and Theorem Proving". In: *AAAI Conference on Artificial Intelligence*. AAAI Press, 2020, pp. 2967–2974. URL: https://ojs.aaai.org/index.php/AAAI/article/view/5689 (cit. on pp. 43, 123).

[Par+20] Emilio Parisotto, H. Francis Song, Jack W. Rae, Razvan Pascanu, Çaglar Gülçehre, Siddhant M. Jayakumar, Max Jaderberg, Raphaël Lopez Kaufman, Aidan Clark, Seb Noury, Matthew Botvinick, Nicolas Heess, and Raia Hadsell. "Stabilizing Transformers for Reinforcement Learning". In: *International Conference on Machine Learning, ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 7487–7498. URL: http://proceedings.mlr.press/v119/parisotto20a.html (cit. on p. 41).

[Pas+19] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems*. 2019, pp. 8024–8035. URL: https://proceedings.neurips.cc/paper/2019/hash/bdbca288fee7f92f2bfa9f7012727740-Abstract.html (cit. on pp. 47, 48, 152, 156).

[PBS16] Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. "Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning". In: *International Conference on Learning Representations*. 2016. URL: http://arxiv.org/abs/1511.06342 (cit. on pp. 22, 25, 32, 39).

[Pea89] Judea Pearl. *Probabilistic reasoning in intelligent systems - networks of plausible inference*. Morgan Kaufmann series in representation and reasoning. Morgan Kaufmann, 1989. URL: https://www.sciencedirect.com/book/9780080514895/probabilistic-reasoning-in-intelligent-systems (cit. on p. 40).

[Pen+18] Xue Bin Peng, Marcin Andrychowicz, Wojciech Zaremba, and Pieter Abbeel. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *IEEE International Conference on Robotics and Automation, ICRA*. IEEE, 2018, pp. 1–8. URL: https://doi.org/10.1109/ICRA.2018.8460528 (cit. on p. 29).

[Pen+19] Xue Bin Peng, Michael Chang, Grace Zhang, Pieter Abbeel, and Sergey Levine. "MCP: Learning Composable Hierarchical Control with Multiplicative Compositional Policies". In: *Advances in Neural Information Processing Systems*. 2019, pp. 3681–3692. URL: https://proceedings.neurips.cc/paper/2019/hash/95192c98732387165bf8e396c0f2dad2-Abstract.html (cit. on p. 24).

[Pet+18] Matthew Peters, Mark Neumann, Luke Zettlemoyer, and Wen-tau Yih. "Dissecting Contextual Word Embeddings: Architecture and Representation". In: *Conference on Empirical Methods in Natural Language Processing*. Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 1499–1509. URL: https://www.aclweb.org/anthology/D18-1179 (cit. on p. 88).

[Pol+22] Stanislas Polu, Jesse Michael Han, Kunhao Zheng, Mantas Baksys, Igor Babuschkin, and Ilya Sutskever. "Formal Mathematics Statement Curriculum Learning". In: *CoRR* abs/2202.01344 (2022). arXiv: 2202.01344. URL: https://arxiv.org/abs/2202.01344 (cit. on p. 37).

[Por+19] Rémy Portelas, Cédric Colas, Katja Hofmann, and Pierre-Yves Oudeyer. "Teacher algorithms for curriculum learning of Deep RL in continuously parameterized environments". In: *3rd Annual Conference on Robot*

Learning, CoRL 2019. Vol. 100. Proceedings of Machine Learning Research. PMLR, 2019, pp. 835–853. URL: http://proceedings.mlr.press/v100/portelas20a.html (cit. on p. 34).

[Por+20] Rémy Portelas, Cédric Colas, Lilian Weng, Katja Hofmann, and Pierre-Yves Oudeyer. "Automatic Curriculum Learning For Deep RL: A Short Survey". In: *International Joint Conference on Artificial Intelligence, IJCAI*. ijcai.org, 2020, pp. 4819–4825. URL: https://doi.org/10.24963/ijcai.2020/671 (cit. on p. 34).

[PS06] Jan Peters and Stefan Schaal. "Policy Gradient Methods for Robotics". In: *International Conference on Intelligent Robots and Systems, IROS*. IEEE, 2006, pp. 2219–2225. URL: https://doi.org/10.1109/IROS.2006.282564 (cit. on p. 12).

[PSM14] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. "Glove: Global Vectors for Word Representation". In: *Conference on Empirical Methods in Natural Language Processing, EMNLP*. ACL, 2014, pp. 1532–1543. URL: https://doi.org/10.3115/v1/d14-1162 (cit. on p. 83).

[Put14] Martin L Puterman. *Markov decision processes: discrete stochastic dynamic programming*. John Wiley & Sons, 2014. URL: https://www.wiley.com/en-us/Markov+Decision+Processes%3A+Discrete+Stochastic+Dynamic+Programming-p-9781118625873 (cit. on p. 10).

[Raj+17] Aravind Rajeswaran, Sarvjeet Ghotra, Balaraman Ravindran, and Sergey Levine. "EPOpt: Learning Robust Neural Network Policies Using Model Ensembles". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=SyWvgP5el (cit. on p. 28).

[Ras+20] Tabish Rashid, Mikayel Samvelyan, Christian Schröder de Witt, Gregory Farquhar, Jakob N. Foerster, and Shimon Whiteson. "Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning". In: *J. Mach. Learn. Res.* 21 (2020), 178:1–178:51. URL: http://jmlr.org/papers/v21/20-081.html (cit. on p. 41).

[Ree+22] Scott E. Reed, 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. "A Generalist Agent". In: *CoRR*

abs/2205.06175 (2022). arXiv: 2205.06175. URL: https://doi.org/10.48550/arXiv.2205.06175 (cit. on p. 27).

[RHW86] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning representations by back-propagating errors". In: *nature* 323.6088 (1986), pp. 533–536. URL: https://www.nature.com/articles/323533a0 (cit. on p. 16).

[Rie+18] Martin A. Riedmiller, Roland Hafner, Thomas Lampe, Michael Neunert, Jonas Degrave, Tom Van de Wiele, Vlad Mnih, Nicolas Heess, and Jost Tobias Springenberg. "Learning by Playing Solving Sparse Reward Tasks from Scratch". In: *International Conference on Machine Learning, ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 4341–4350. URL: http://proceedings.mlr.press/v80/riedmiller18a.html (cit. on p. 39).

[Rin95] Mark B. Ring. "Continual learning in reinforcement environments". PhD thesis. University of Texas at Austin, TX, USA, 1995. URL: https://d-nb.info/945690320 (cit. on p. 37).

[RST20] Alessandro Raganato, Yves Scherrer, and Jörg Tiedemann. "Fixed Encoder Self-Attention Patterns in Transformer-Based Machine Translation". In: *Conference on Empirical Methods in Natural Language Processing: Findings, EMNLP*. ACL, 2020, pp. 556–568 (cit. on p. 84).

[Rud17] Sebastian Ruder. "An Overview of Multi-Task Learning in Deep Neural Networks". In: *CoRR* abs/1706.05098 (2017). arXiv: 1706.05098. URL: http://arxiv.org/abs/1706.05098 (cit. on p. 19).

[Rus+16a] Andrei A. Rusu, Sergio Gomez Colmenarejo, Çaglar Gülçehre, Guillaume Desjardins, James Kirkpatrick, Razvan Pascanu, Volodymyr Mnih, Koray Kavukcuoglu, and Raia Hadsell. "Policy Distillation". In: *International Conference on Learning Representations, ICLR*. 2016 (cit. on pp. 25, 26, 39).

[Rus+16b] Andrei A. Rusu, Neil C. Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell. "Progressive Neural Networks". In: *CoRR* abs/1606.04671 (2016). arXiv: 1606.04671. URL: http://arxiv.org/abs/1606.04671 (cit. on p. 37).

[Sam+21] Mikayel Samvelyan, Robert Kirk, Vitaly Kurin, Jack Parker-Holder, Minqi Jiang, Eric Hambro, Fabio Petroni, Heinrich Kuttler, Edward Grefenstette, and Tim Rocktäschel. "MiniHack the Planet: A Sandbox for Open-Ended Reinforcement Learning Research". In: *Neural Information Processing Systems, Datasets and Benchmarks Track*. 2021.

URL: https://openreview.net/forum?id=skFwlyefkWJ (cit. on pp. 30, 43).

[San+18] Alvaro Sanchez-Gonzalez, Nicolas Heess, Jost Tobias Springenberg, Josh Merel, Martin A. Riedmiller, Raia Hadsell, and Peter W. Battaglia. "Graph Networks as Learnable Physics Engines for Inference and Control". In: *International Conference on Machine Learning, ICML*. Vol. 80. Proceedings of Machine Learning Research. PMLR, 2018, pp. 4467–4476. URL: http://proceedings.mlr.press/v80/sanchez-gonzalez18a.html (cit. on p. 42).

[SB19] Daniel Selsam and Nikolaj S. Bjørner. "Guiding High-Performance SAT Solvers with Unsat-Core Predictions". In: *Theory and Applications of Satisfiability Testing - SAT*. Vol. 11628. Lecture Notes in Computer Science. Springer, 2019, pp. 336–353. URL: https://doi.org/10.1007/978-3-030-24258-9%5C_24 (cit. on pp. 121, 124).

[Sch+15] John Schulman, Sergey Levine, Pieter Abbeel, Michael I. Jordan, and Philipp Moritz. "Trust Region Policy Optimization". In: *International Conference on Machine Learning, ICML*. Vol. 37. JMLR Workshop and Conference Proceedings. JMLR.org, 2015, pp. 1889–1897. URL: http://proceedings.mlr.press/v37/schulman15.html (cit. on pp. 12, 26, 69).

[Sch+16] John Schulman, Philipp Moritz, Sergey Levine, Michael I. Jordan, and Pieter Abbeel. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *International Conference on Learning Representations, ICLR*. 2016. URL: http://arxiv.org/abs/1506.02438 (cit. on p. 12).

[Sch+17] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. "Proximal Policy Optimization Algorithms". In: *CoRR* abs/1707.06347 (2017). arXiv: 1707.06347. URL: http://arxiv.org/abs/1707.06347 (cit. on pp. 13, 26, 69, 149).

[SD13] Michael L. Seltzer and Jasha Droppo. "Multi-task learning in deep neural networks for improved phoneme recognition". In: *IEEE International Conference on Acoustics, Speech and Signal Processing, ICASSP 2013*. IEEE, 2013, pp. 6965–6969. URL: https://doi.org/10.1109/ICASSP.2013.6639012 (cit. on p. 20).

[SE05] Niklas Sorensson and Niklas Een. "Minisat v1. 13-a sat solver with conflict-clause minimization". In: *SAT* 2005.53 (2005), pp. 1–2. URL: http://minisat.se/downloads/MiniSat_v1.13_short.pdf (cit. on pp. 43, 108).

[Sel+19] Daniel Selsam, Matthew Lamm, Benedikt Bünz, Percy Liang, Leonardo de Moura, and David L. Dill. "Learning a SAT Solver from Single-Bit Supervision". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=HJMC%5C_iA5tm (cit. on pp. 6, 106, 110, 116, 121).

[Seq] Sequence-to-Sequence Modeling, Pytorch Tutorial. *Sequence-to-Sequence Modeling with nn.Transformer and TorchText*. [Online; accessed 8-August-2020]. URL: https://pytorch.org/tutorials/beginner/transformer_tutorial.html (cit. on p. 152).

[SFH17] Sara Sabour, Nicholas Frosst, and Geoffrey E. Hinton. "Dynamic Routing Between Capsules". In: *Advances in Neural Information Processing Systems*. 2017, pp. 3856–3866. URL: https://proceedings.neurips.cc/paper/2017/hash/2cad8fa47bbef282badbb8de5374b894-Abstract.html (cit. on p. 132).

[Sha+18] Sahil Sharma, Ashutosh Kumar Jha, Parikshit Hegde, and Balaraman Ravindran. "Learning to Multi-Task by Active Sampling". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=B1nZ1weCZ (cit. on p. 36).

[She+17] Evan Shelhamer, Parsa Mahmoudieh, Max Argus, and Trevor Darrell. "Loss is its own Reward: Self-Supervision for Reinforcement Learning". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2017. URL: https://openreview.net/forum?id=S15PPJStl (cit. on p. 38).

[Sil+16] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. "Mastering the game of Go with deep neural networks and tree search". In: *Nat.* 529.7587 (2016), pp. 484–489. URL: https://doi.org/10.1038/nature16961 (cit. on p. 168).

[Sil+17] David Silver, 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, Laurent Sifre, George van den Driessche, Thore Graepel, and Demis Hassabis. "Mastering the game of Go without human knowledge". In: *Nat.* 550.7676 (2017),

pp. 354–359. URL: https://doi.org/10.1038/nature24270 (cit. on p. 121).

[Sin+09] Rishabh Singh, Joseph P Near, Vijay Ganesh, and Martin Rinard. "AvatarSAT: An Auto-tuning Boolean SAT Solver". In: (2009). URL: http://people.csail.mit.edu/rishabh/papers/avatarsat.pdf (cit. on p. 121).

[Sin+15] Jivko Sinapov, Sanmit Narvekar, Matteo Leonetti, and Peter Stone. "Learning Inter-Task Transferability in the Absence of Target Task Samples". In: *International Conference on Autonomous Agents and Multiagent Systems, AAMAS*. ACM, 2015, pp. 725–733. URL: http://dl.acm.org/citation.cfm?id=2773247 (cit. on p. 36).

[SK18] Ozan Sener and Vladlen Koltun. "Multi-Task Learning as Multi-Objective Optimization". In: *Advances in Neural Information Processing Systems*. 2018, pp. 525–536. URL: https://proceedings.neurips.cc/paper/2018/hash/432aca3a1e345e339f35a30c8f65edce-Abstract.html (cit. on pp. 9, 20, 46, 48, 49, 53, 54, 62, 132, 133, 137, 138, 158, 159).

[SKC93] Bart Selman, Henry A. Kautz, and Bram Cohen. "Local search strategies for satisfiability testing". In: *Cliques, Coloring, and Satisfiability, DIMACS Workshop*. Vol. 26. DIMACS Series in Discrete Mathematics and Theoretical Computer Science. DIMACS/AMS, 1993, pp. 521–531. URL: https://doi.org/10.1090/dimacs/026/25 (cit. on pp. 43, 122).

[SL17] Fereshteh Sadeghi and Sergey Levine. "CAD2RL: Real Single-Image Flight Without a Single Real Image". In: *Robotics: Science and Systems XIII, Massachusetts Institute of Technology*. 2017. URL: http://www.roboticsproceedings.org/rss13/p34.html (cit. on p. 29).

[SMG14] Andrew M. Saxe, James L. McClelland, and Surya Ganguli. "Exact solutions to the nonlinear dynamics of learning in deep linear neural networks". In: *International Conference on Learning Representations, ICLR*. 2014. URL: http://arxiv.org/abs/1312.6120 (cit. on p. 78).

[Son+20] H. Francis Song, Abbas Abdolmaleki, Jost Tobias Springenberg, Aidan Clark, Hubert Soyer, Jack W. Rae, Seb Noury, Arun Ahuja, Siqi Liu, Dhruva Tirumala, Nicolas Heess, Dan Belov, Martin A. Riedmiller, and Matthew M. Botvinick. "V-MPO: On-Policy Maximum a Posteriori Policy Optimization for Discrete and Continuous Control". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2020. URL: https://openreview.net/forum?id=SylOlp4FvH (cit. on pp. 24, 39).

[SPH20] Gregor N. C. Simm, Robert Pinsler, and José Miguel Hernández-Lobato. "Reinforcement Learning for Molecular Design Guided by Quantum Mechanics". In: *International Conference on Machine Learning, ICML*. Vol. 119. Proceedings of Machine Learning Research. PMLR, 2020, pp. 8959–8969. URL: http://proceedings.mlr.press/v119/simm20b.html (cit. on p. 43).

[SPS99] Richard S. Sutton, Doina Precup, and Satinder Singh. "Between MDPs and Semi-MDPs: A Framework for Temporal Abstraction in Reinforcement Learning". In: *Artif. Intell.* 112.1-2 (1999), pp. 181–211. URL: https://doi.org/10.1016/S0004-3702(99)00052-1 (cit. on p. 24).

[Sri+14] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a simple way to prevent neural networks from overfitting". In: *J. Mach. Learn. Res.* 15.1 (2014), pp. 1929–1958. URL: https://dl.acm.org/doi/10.5555/2627435.2670313 (cit. on pp. 48, 53).

[SS99] João P. Marques Silva and Karem A. Sakallah. "GRASP: A Search Algorithm for Propositional Satisfiability". In: *IEEE Trans. Computers* 48.5 (1999), pp. 506–521. URL: https://doi.org/10.1109/12.769433 (cit. on p. 106).

[Sut+11] Richard S. Sutton, Joseph Modayil, Michael Delp, Thomas Degris, Patrick M. Pilarski, Adam White, and Doina Precup. "Horde: a scalable real-time architecture for learning knowledge from unsupervised sensorimotor interaction". In: *International Conference on Autonomous Agents and Multiagent Systems AAMAS*. IFAAMAS, 2011, pp. 761–768. URL: http://portal.acm.org/citation.cfm?id=2031726%5C&CFID=54178199%5C&CFTOKEN=61392764 (cit. on p. 38).

[Sut92] Richard S. Sutton. "Adapting Bias by Gradient Descent: An Incremental Version of Delta-Bar-Delta". In: *10th National Conference on Artificial Intelligence*. AAAI Press / The MIT Press, 1992, pp. 171–176. URL: http://www.aaai.org/Library/AAAI/1992/aaai92-027.php (cit. on p. 21).

[Sve+17] Maxwell Svetlik, Matteo Leonetti, Jivko Sinapov, Rishi Shah, Nick Walker, and Peter Stone. "Automatic Curriculum Graph Generation for Reinforcement Learning Agents". In: *AAAI Conference on Artificial Intelligence*. AAAI Press, 2017, pp. 2590–2596. URL: http://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14961 (cit. on p. 34).

[SZP21] Shagun Sodhani, Amy Zhang, and Joelle Pineau. "Multi-Task Reinforcement Learning with Context-based Representations". In: *International Conference on Machine Learning,ICML*. Vol. 139. Proceedings of Machine Learning Research. PMLR, 2021, pp. 9767–9779. URL: http://proceedings.mlr.press/v139/sodhani21a.html (cit. on pp. 24, 57, 58, 134–137).

[Tay+20] Yi Tay, Mostafa Dehghani, Dara Bahri, and Donald Metzler. "Efficient Transformers: A Survey". In: *CoRR* abs/2009.06732 (2020). arXiv: 2009.06732. URL: https://arxiv.org/abs/2009.06732 (cit. on p. 104).

[Teh+17] Yee Whye Teh, Victor Bapst, Wojciech M. Czarnecki, John Quan, James Kirkpatrick, Raia Hadsell, Nicolas Heess, and Razvan Pascanu. "Distral: Robust multitask reinforcement learning". In: *Advances in Neural Information Processing Systems*. 2017, pp. 4496–4506. URL: https://proceedings.neurips.cc/paper/2017/hash/0abdc563a06105aee3c6136871c9f4d1-Abstract.html (cit. on pp. 26, 39, 46).

[Ten+19] Ian Tenney, Patrick Xia, Berlin Chen, Alex Wang, Adam Poliak, R. Thomas McCoy, Najoung Kim, Benjamin Van Durme, Samuel R. Bowman, Dipanjan Das, and Ellie Pavlick. "What do you learn from context? Probing for sentence structure in contextualized word representations". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=SJzSgnRcKX (cit. on p. 88).

[TET12] Emanuel Todorov, Tom Erez, and Yuval Tassa. "MuJoCo: A physics engine for model-based control". In: *International Conference on Intelligent Robots and Systems, IROS*. IEEE, 2012, pp. 5026–5033. URL: https://doi.org/10.1109/IROS.2012.6386109 (cit. on pp. 42, 70, 72, 151).

[TFK21] Arash Tavakoli, Mehdi Fatemi, and Petar Kormushev. "Learning to Represent Action Values as a Hypergraph on the Action Vertices". In: *International Conference on Learning Representations*. 2021. URL: https://openreview.net/forum?id=Xv_s64FiXTv (cit. on p. 40).

[Thr95] Sebastian Thrun. "Is Learning The n-th Thing Any Easier Than Learning The First?" In: *Advances in Neural Information Processing Systems*. MIT Press, 1995, pp. 640–646. URL: http://papers.nips.cc/paper/1034-is-learning-the-n-th-thing-any-easier-than-learning-the-first (cit. on p. 37).

[TJS08] Matthew E. Taylor, Nicholas K. Jong, and Peter Stone. "Transferring Instances for Model-Based Reinforcement Learning". In: *Machine Learning and Knowledge Discovery in Databases, European Conference, ECML/PKDD*. Vol. 5212. Lecture Notes in Computer Science. Springer, 2008, pp. 488–505. URL: https://doi.org/10.1007/978-3-540-87481-2%5C_32 (cit. on p. 31).

[TKS08] Matthew E. Taylor, Gregory Kuhlmann, and Peter Stone. "Autonomous transfer for reinforcement learning". In: *International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS*. IFAAMAS, 2008, pp. 283–290. URL: https://dl.acm.org/citation.cfm?id=1402427 (cit. on p. 31).

[Tob+17] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. "Domain randomization for transferring deep neural networks from simulation to the real world". In: *International Conference on Intelligent Robots and Systems, IROS 2017*. IEEE, 2017, pp. 23–30. URL: https://doi.org/10.1109/IROS.2017.8202133 (cit. on p. 29).

[TS09] Matthew E. Taylor and Peter Stone. "Transfer Learning for Reinforcement Learning Domains: A Survey". In: *J. Mach. Learn. Res.* 10 (2009), pp. 1633–1685. URL: https://dl.acm.org/doi/10.5555/1577069.1755839 (cit. on p. 31).

[TS11] Matthew E. Taylor and Peter Stone. "An Introduction to Intertask Transfer for Reinforcement Learning". In: *AI Mag.* 32.1 (2011), pp. 15–34. URL: https://doi.org/10.1609/aimag.v32i1.2329 (cit. on p. 31).

[Tse20] Wei-Cheng Tseng. *WeiChengTseng/Pytorch-PCGrad*. 2020. URL: https://github.com/WeiChengTseng/Pytorch-PCGrad.git (cit. on p. 137).

[TSG21] Momchil S Tomov, Eric Schulz, and Samuel J Gershman. "Multi-task reinforcement learning in humans". In: *Nature Human Behaviour* 5.6 (2021), pp. 764–773. URL: https://www.nature.com/articles/s41562-020-01035-y (cit. on p. 1).

[TWS07a] Matthew E. Taylor, Shimon Whiteson, and Peter Stone. "Transfer via inter-task mappings in policy search reinforcement learning". In: *International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS*. IFAAMAS, 2007, p. 37. URL: https://doi.org/10.1145/1329125.1329170 (cit. on p. 31).

[TWS07b] Matthew E. Taylor, Shimon Whiteson, and Peter Stone. "Transfer via inter-task mappings in policy search reinforcement learning". In: *International Joint Conference on Autonomous Agents and Multiagent Systems AAMAS*. IFAAMAS, 2007, p. 37. URL: https://doi.org/10.1145/1329125.1329170 (cit. on p. 31).

[Tze+15] Eric Tzeng, Coline Devin, Judy Hoffman, Chelsea Finn, Xingchao Peng, Sergey Levine, Kate Saenko, and Trevor Darrell. "Towards Adapting Deep Visuomotor Representations from Simulated to Real Environments". In: *CoRR* abs/1511.07111 (2015). arXiv: 1511.07111. URL: http://arxiv.org/abs/1511.07111 (cit. on p. 28).

[Van+22] Simon Vandenhende, Stamatios Georgoulis, Wouter Van Gansbeke, Marc Proesmans, Dengxin Dai, and Luc Van Gool. "Multi-Task Learning for Dense Prediction Tasks: A Survey". In: *IEEE Trans. Pattern Anal. Mach. Intell.* 44.7 (2022), pp. 3614–3633. URL: https://doi.org/10.1109/TPAMI.2021.3054719 (cit. on p. 20).

[Vas+17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. "Attention is All you Need". In: *Advances in Neural Information Processing Systems*. 2017, pp. 5998–6008. URL: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html (cit. on pp. 6, 16, 17, 88).

[VB19] Jesse Vig and Yonatan Belinkov. "Analyzing the Structure of Attention in a Transformer Language Model". In: *ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*. Florence, Italy: Association for Computational Linguistics, 2019, pp. 63–76. URL: https://www.aclweb.org/anthology/W19-4808 (cit. on p. 88).

[VB21] Petar Velickovic and Charles Blundell. "Neural algorithmic reasoning". In: *Patterns* 2.7 (2021), p. 100273. URL: https://doi.org/10.1016/j.patter.2021.100273 (cit. on pp. 23, 129).

[Vel+17] Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. "Graph Attention Networks". In: *CoRR* abs/1710.10903 (2017). arXiv: 1710.10903. URL: http://arxiv.org/abs/1710.10903 (cit. on p. 16).

[VFJ15] Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. "Pointer Networks". In: *Advances in Neural Information Processing Systems*. 2015, pp. 2692–2700. URL: http://papers.nips.cc/paper/5866-pointer-networks (cit. on p. 122).

[vHas+16] Hado van Hasselt, Arthur Guez, Matteo Hessel, Volodymyr Mnih, and David Silver. "Learning values across many orders of magnitude". In: *Advances in Neural Information Processing Systems*. 2016, pp. 4287–4295. URL: https://proceedings.neurips.cc/paper/2016/hash/5227b6aaf294f5f027273aebf16015f2-Abstract.html (cit. on pp. 23, 24, 58).

[Vin+19] Oriol Vinyals et al. "Grandmaster level in StarCraft II using multi-agent reinforcement learning". In: *Nat.* 575.7782 (2019), pp. 350–354. URL: https://doi.org/10.1038/s41586-019-1724-z (cit. on p. 168).

[VM20] Nelson Vithayathil Varghese and Qusay H. Mahmoud. "A Survey of Multi-Task Deep Reinforcement Learning". In: *Electronics* 9.9 (2020), pp. 1363–1384. URL: http://dx.doi.org/10.3390/electronics9091363 (cit. on p. 39).

[Wan+16] Ziyu Wang, Frank Hutter, Masrour Zoghi, David Matheson, and Nando de Freitas. "Bayesian Optimization in a Billion Dimensions via Random Embeddings". In: *J. Artif. Intell. Res.* 55 (2016), pp. 361–387. URL: https://doi.org/10.1613/jair.4806 (cit. on p. 83).

[Wan+18] Tingwu Wang, Renjie Liao, Jimmy Ba, and Sanja Fidler. "NerveNet: Learning Structured Policy with Graph Neural Networks". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2018. URL: https://openreview.net/forum?id=S1sqHMZCb (cit. on pp. 5, 23, 25, 32, 40–42, 68, 70, 72, 73, 79, 80, 87, 88, 91, 92, 95, 98, 149, 150, 152, 153).

[Wan+19] Rui Wang, Joel Lehman, Jeff Clune, and Kenneth O. Stanley. "POET: open-ended coevolution of environments and their optimized solutions". In: *Genetic and Evolutionary Computation Conference GECCO*. ACM, 2019, pp. 142–151. URL: https://doi.org/10.1145/3321707.3321799 (cit. on p. 35).

[Wan+21] Zirui Wang, Yulia Tsvetkov, Orhan Firat, and Yuan Cao. "Gradient Vaccine: Investigating and Improving Multi-task Optimization in Massively Multilingual Models". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2021. URL: https://openreview.net/forum?id=F1vEjWK-lH%5C_ (cit. on pp. 20, 46, 64).

[WD92] Christopher J. C. H. Watkins and Peter Dayan. "Technical Note Q-Learning". In: *Mach. Learn.* 8 (1992), pp. 279–292. URL: https://doi.org/10.1007/BF00992698 (cit. on p. 12).

[Whi+09] Shimon Whiteson, Brian Tanner, Matthew E. Taylor, and Peter Stone. "Generalized Domains for Empirical Evaluations in Reinforcement Learning". In: *ICML Workshop on Evaluation Methods for Machine Learning*. June 2009. URL: https://www.cs.utexas.edu/users/pstone/Papers/bib2html/b2hd-ICML09ws-shimon.html (cit. on p. 27).

[Whi+11] Shimon Whiteson, Brian Tanner, Matthew E. Taylor, and Peter Stone. "Protecting against evaluation overfitting in empirical reinforcement learning". In: *IEEE Symposium on Adaptive Dynamic Programming And Reinforcement Learning, ADPRL*. IEEE, 2011, pp. 120–127. URL: https://doi.org/10.1109/ADPRL.2011.5967363 (cit. on p. 27).

[Wil+07] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. "Multi-task reinforcement learning: a hierarchical Bayesian approach". In: *International Conference on Machine Learning ICML*. Vol. 227. ACM International Conference Proceeding Series. ACM, 2007, pp. 1015–1022. URL: https://doi.org/10.1145/1273496.1273624 (cit. on p. 22).

[Wil92] Ronald J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Mach. Learn.* 8 (1992), pp. 229–256. URL: https://doi.org/10.1007/BF00992696 (cit. on pp. 43, 122).

[WR18] Fei Wang and Tiark Rompf. "From Gameplay to Symbolic Reasoning: Learning SAT Solver Heuristics in the Style of Alpha(Go) Zero". In: *CoRR* abs/1802.05340 (2018). arXiv: 1802.05340. URL: http://arxiv.org/abs/1802.05340 (cit. on pp. 43, 109, 121).

[Wu+21] Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S. Yu. "A Comprehensive Survey on Graph Neural Networks". In: *IEEE Trans. Neural Networks Learn. Syst.* 32.1 (2021), pp. 4–24. URL: https://doi.org/10.1109/TNNLS.2020.2978386 (cit. on pp. 39, 40).

[Xu+08] Lin Xu, Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. "SATzilla: Portfolio-based Algorithm Selection for SAT". In: *J. Artif. Intell. Res.* 32 (2008), pp. 565–606. URL: https://doi.org/10.1613/jair.2490 (cit. on p. 121).

[Yan+20] Ruihan Yang, Huazhe Xu, Yi Wu, and Xiaolong Wang. "Multi-Task Reinforcement Learning with Soft Modularization". In: *Advances in Neural Information Processing Systems*. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/32cfdce9631d8c7906e8e9d6e68b514b-Abstract.html (cit. on p. 24).

[Yan+22] Menglin Yang, Min Zhou, Zhihao Li, Jiahong Liu, Lujia Pan, Hui Xiong, and Irwin King. "Hyperbolic graph neural networks: A review of methods and applications". In: *arXiv preprint arXiv:2202.13852* (2022) (cit. on p. 127).

[YKF17] Fisher Yu, Vladlen Koltun, and Thomas A. Funkhouser. "Dilated Residual Networks". In: *IEEE Conference on Computer Vision and Pattern Recognition, CVPR*. IEEE Computer Society, 2017, pp. 636–644. URL: https://doi.org/10.1109/CVPR.2017.75 (cit. on p. 133).

[YLT19] Wenhao Yu, C. Karen Liu, and Greg Turk. "Policy Transfer with Strategy Optimization". In: *International Conference on Learning Representations, ICLR*. OpenReview.net, 2019. URL: https://openreview.net/forum?id=H1g6osRcFQ (cit. on pp. 32, 104).

[Yos+14] Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. "How transferable are features in deep neural networks?" In: *Advances in Neural Information Processing Systems*. 2014, pp. 3320–3328. URL: https://proceedings.neurips.cc/paper/2014/hash/375c71349b295fbe2dcdca9206f20a06-Abstract.html (cit. on p. 84).

[YP19] Emre Yolcu and Barnabás Póczos. "Learning Local Search Heuristics for Boolean Satisfiability". In: *Advances in Neural Information Processing Systems 32*. 2019, pp. 7990–8001. URL: https://proceedings.neurips.cc/paper/2019/hash/12e59a33dea1bf0630f46edfe13d6ea2-Abstract.html (cit. on pp. 43, 122).

[YSI20] Weiqiu You, Simeng Sun, and Mohit Iyyer. "Hard-Coded Gaussian Attention for Neural Machine Translation". In: *58th Annual Meeting of the Association for Computational Linguistics, ACL*. Association for Computational Linguistics, 2020, pp. 7689–7700. URL: https://doi.org/10.18653/v1/2020.acl-main.687 (cit. on p. 84).

[Yu+19] Tianhe Yu, Deirdre Quillen, Zhanpeng He, Ryan Julian, Karol Hausman, Chelsea Finn, and Sergey Levine. "Meta-World: A Benchmark and Evaluation for Multi-Task and Meta Reinforcement Learning". In: *3rd Annual Conference on Robot Learning, CoRL*. Vol. 100. Proceedings of Machine Learning Research. PMLR, 2019, pp. 1094–1100. URL: http://proceedings.mlr.press/v100/yu20a.html (cit. on pp. 28, 42, 57).

[Yu+20] Tianhe Yu, Saurabh Kumar, Abhishek Gupta, Sergey Levine, Karol Hausman, and Chelsea Finn. "Gradient Surgery for Multi-Task Learning". In: *Advances in Neural Information Processing Systems*. 2020.

URL: https://proceedings.neurips.cc/paper/2020/hash/3fe78a8acf5fda99de95303940a2420c-Abstract.html (cit. on pp. 9, 20, 23, 46, 50, 52, 53, 57, 58, 64, 133, 163, 164).

[Zam+18] Vinícius Flores Zambaldi, David Raposo, Adam Santoro, Victor Bapst, Yujia Li, Igor Babuschkin, Karl Tuyls, David P. Reichert, Timothy P. Lillicrap, Edward Lockhart, Murray Shanahan, Victoria Langston, Razvan Pascanu, Matthew Botvinick, Oriol Vinyals, and Peter W. Battaglia. "Relational Deep Reinforcement Learning". In: *CoRR* abs/1806.01830 (2018). arXiv: 1806.01830. URL: http://arxiv.org/abs/1806.01830 (cit. on p. 41).

[ZAP20] Yunzhi Zhang, Pieter Abbeel, and Lerrel Pinto. "Automatic Curriculum Learning through Value Disagreement". In: *Advances in Neural Information Processing Systems*. 2020. URL: https://proceedings.neurips.cc/paper/2020/hash/566f0ea4f6c2e947f36795c8f58ba901-Abstract.html (cit. on p. 35).

[ZBP18] Amy Zhang, Nicolas Ballas, and Joelle Pineau. "A Dissection of Overfitting and Generalization in Continuous Reinforcement Learning". In: *CoRR* abs/1806.07937 (2018). arXiv: 1806.07937. URL: http://arxiv.org/abs/1806.07937 (cit. on p. 28).

[ZCZ18] Ziwei Zhang, Peng Cui, and Wenwu Zhu. "Deep Learning on Graphs: A Survey". In: *CoRR* abs/1812.04202 (2018). arXiv: 1812.04202. URL: http://arxiv.org/abs/1812.04202 (cit. on p. 39).

[Zha+18] Chiyuan Zhang, Oriol Vinyals, Rémi Munos, and Samy Bengio. "A Study on Overfitting in Deep Reinforcement Learning". In: *CoRR* abs/1804.06893 (2018). arXiv: 1804.06893. URL: http://arxiv.org/abs/1804.06893 (cit. on p. 28).

[Zho+20] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. "Graph neural networks: A review of methods and applications". In: *AI Open* 1 (2020), pp. 57–81. URL: https://doi.org/10.1016/j.aiopen.2021.01.001 (cit. on p. 39).

[Zhu+21] Yanqiao Zhu, Weizhi Xu, Jinghao Zhang, Yuanqi Du, Jieyu Zhang, Qiang Liu, Carl Yang, and Shu Wu. "A survey on graph structure learning: Progress and opportunities". In: *arXiv e-prints* (2021), arXiv–2103 (cit. on p. 104).

[Zin+19] Luisa M. Zintgraf, Kyriacos Shiarlis, Vitaly Kurin, Katja Hofmann, and Shimon Whiteson. "Fast Context Adaptation via Meta-Learning". In: *International Conference on Machine Learning, ICML*. Vol. 97. Proceedings of Machine Learning Research. PMLR, 2019, pp. 7693–7702. URL: http://proceedings.mlr.press/v97/zintgraf19a.html (cit. on p. 33).

[ZLZ20] Zhuangdi Zhu, Kaixiang Lin, and Jiayu Zhou. "Transfer Learning in Deep Reinforcement Learning: A Survey". In: *CoRR* abs/2009.07888 (2020). arXiv: 2009.07888. URL: https://arxiv.org/abs/2009.07888 (cit. on p. 33).

[Zom+19] Zsolt Zombori, Adrián Csiszárik, Henryk Michalewski, Cezary Kaliszyk, and Josef Urban. "Curriculum learning and theorem proving". In: *Conference on Artificial Intelligence and Theorem Proving*. 2019. URL: http://aitp-conference.org/2019/abstract/paper%2013.pdf (cit. on p. 37).

[ZY17] Yu Zhang and Qiang Yang. "A Survey on Multi-Task Learning". In: *CoRR* abs/1707.08114 (2017). arXiv: 1707.08114. URL: http://arxiv.org/abs/1707.08114 (cit. on p. 27).