UNIVERSIDAD
DE MÁLAGA

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

Universidad de Málaga
Escuela de Ingenierías Industriales

TESIS DOCTORAL

Nuevos algoritmos de aprendizaje por refuerzo distributivo y aprendizaje ensamblador

En cooperación con HAW Hamburg, Alemania

Autor:	M.Sc. Vanya Aziz
Directores:	Dr. Eligius M.T. Hendrix & Dr. Ivo Nowak
Programa de Doctorado:	Programa de Doctorado en Ingeniería Mécatronica
Centro:	E.T.S. de Ingeniería Informática

Málaga, Febrero 2025

AUTORA: Vanya Aziz

http://orcid.org/0009-0003-1068-9729

EDITA: Publicaciones y Divulgación Científica. Universidad de Málaga

UNIVERSIDAD DE MÁLAGA

Hochschule für Angewandte Wissenschaften Hamburg
Hamburg University of Applied Sciences

University of Málaga
School of Industrial Engineering

Ph.D. Thesis

Novel Distributional Reinforcement and Ensemble Learning Algorithms

In cooperation with HAW Hamburg, Germany

Author: M.Sc. Vanya Aziz
Supervisors: Dr. Eligius M.T. Hendrix
Dr. Ivo Nowak

Málaga, January 2025

Abstract

The term “Industry 4.0” describes the fourth industrial revolution and is characterized by the integration of digital technology into manufacturing processes. The transformative concepts in Industry 4.0 enable economic production at radically small lot sizes, requiring unprecedented levels of automation and adaptability. These requirements on production facilities necessitate autonomously acting and self-optimizing systems. The field of machine learning offers promising solutions to achieve the objectives of Industry 4.0, particularly by enabling data-driven decision-making and adaptive control mechanisms. This dissertation focuses on Deep Reinforcement Learning (DRL), a neural network-based approach for solving Markov Decision Processes in high-dimensional spaces with unknown transition dynamics. The main contribution of this thesis is the development of a novel state-of-the-art distributional reinforcement learning algorithm within the maximum-entropy Actor-Critic framework. This algorithm, termed ”Cramér-based Soft Distributional Soft Actor-critic” (C-DSAC), demonstrates superior performance to other RL algorithms, especially in environments with high-dimensional spaces and complex dynamics. Its performance is shown to be partly rooted in a phenomenon arising in Cramér-metric-based Distributional Reinforcement Learning, referred to as confidence-driven model updates. This mechanism ensures that the value function approximator is updated more conservatively when confidence in its estimates is low. Theoretical justifications for the algorithm are provided, demonstrating its convergence in the policy evaluation setting and, under widely accepted mild assumptions, in the control setting as well. Beyond foundational algorithmic research, this thesis contributes to the practical application of RL in robotics. Given the crucial role of multi-joint robotic systems in modern production technology, a RL meta-algorithm called ”Reinforcement Learning - Inverse Kinematics” (RL-IK) is devised. This approach enhances the applicability of reinforcement learning to robotic control tasks by significantly accelerating convergence to near-optimal policies

compared to standard RL methods. An essential prerequisite for real-world RL applications in control systems is machine perception for state identification. To address challenges in this field, this thesis explores novel Supervised Learning (SL) approaches, validated on image classification tasks, with a focus on ensemble learning strategies. To provide a deeper theoretical understanding, the methods are analysed through the lens of classical optimization and compared to heuristic approaches. This investigation offers valuable insights into the difficulties of finding globally optimal classifier ensembles.

Acknowledgements

First and foremost, I thank the living God for His endless grace and blessings, which have illuminated my path through the vast state-space of life's journey. In His intricate environment, I often found myself acting as a curious agent – opting for exploration over mere exploitation, eagerly sampling every state transition, and collecting life's rewards one unexpected discovery at a time. My decisions leading to my current state have been guided by His rewards during pivotal episodes that I now wish to recount.

I am deeply grateful to my parents Avan and Aziz. By continuously feeding my desire for self-improvement, you ensured that I do not become trapped in local optima but embrace exploratory actions to obtain better policies. Through you I learned what dedication and love – be it for another soul or for higher principles – truly means.

To my aunts Nishtiman and Snawbar, your kindness, prayers, and inspiring words have shaped me since I was an inexperienced agent and you remind me every day to be humble and avoid overestimation bias.

I would like to extend my heartfelt gratitude to Ivo, whose guidance has acted as a masterful policy update in my iterative learning process. Your insightful gradients for navigating the high-dimensional space of mathematics significantly refined my approximation models in this field. Like a well-tuned learning rate, your patience and encouragement helped me discover rewarding state transitions, both in this thesis and in my personal growth as a researcher. The discussions we had at the intersection of Machine Learning and classical Optimization were indeed very rewarding actions.

I also want to deeply thank my supervisor, Eligius, whose uncompromising commitment to formalism fined-tuned aspects of this work. You established an environment in Spain where my learning rate could accelerate and exploration could thrive. You have been indeed a good sparring partner.

Thanks goes to Jan Kronqvist as well for his invaluable input on the ensemble learning part of this thesis. Our engaging discussions on novel algorithm

concepts consistently guided me to new and exciting states.

Last but certainly not least, I extend my deepest appreciation to my girlfriend Jessica. Thank you for adjusting your own reward function to accommodate my busier days and for tolerating my habit of sneaking machine learning theory into our discussions. Your support has provided me with joy and comfort during the most challenging episodes of this journey.

RL terminology aside, I owe my sincerest thanks to all of you. This achievement would not have been possible without you.

Contents

1 Introduction

This chapter establishes the framework for this thesis. It introduces the concept of machine learning, along with several essential terms and key concepts that are necessary for understanding the more advanced topics explored in subsequent chapters. The chapter includes a brief technical excursion into one of the primary drivers behind the success of contemporary machine learning tools: Powerful function approximators, particularly artificial neural networks, inspired by the operating principles of (biological) neurons. This chapter outlines the key motivations and objectives of the research presented in this thesis.

1.1 Introduction to Machine Learning

Machine learning is a branch of Artificial Intelligence (AI) that "refers to the capacity of computer systems to learn and adapt without following explicit instructions, by using algorithms and statistical models to analyse and draw inferences from patterns in data" [1]. Machine learning is used in leveraging historical data to predict outcomes, classify information, make data-driven decisions, cluster data points, reduce dimensionality, and generate "synthetic" data [2]. The field is broadly divided into four paradigms: supervised learning, unsupervised learning, semi-supervised learning, and reinforcement learning. Unsupervised learning employs algorithms to analyze and cluster unlabelled datasets, uncovering latent patterns or data groupings without human guidance. Supervised learning, in contrast, relies on labelled datasets where each data point is paired with a corresponding target output. The model predictions are evaluated against these target labels using a cost function, which quantifies the difference between predicted and actual values. The cost function serves as a critical optimization met-

ric, guiding the training process to minimize prediction errors [3, 4]. Semi-supervised learning integrates both labelled and unlabelled data into the learning process, bridging the methodologies of supervised and unsupervised learning [3]. Reinforcement learning addresses sequential decision-making problems typically formulated within the framework of Markov Decision Processes (MDPs) [5]. In reinforcement learning, the exact calculation of the cost function is hindered by the sparsity of the feedback mechanism. The cost must be approximated. Another distinguishing characteristic from other machine learning paradigms is the absence of a predefined dataset; instead, data must be generated through interactions with the environment.

An important concept in all machine learning branches is that of the generalization error. Generalization error, also referred to as out-of-sample error, quantifies model performance on unseen data. By assessing the model on a validation set, one obtains an estimation of this error, providing insight into the model's performance on unseen instances [3]. For example, a control model trained using a reinforcement learning algorithm is said to have a low generalization error if it consistently exhibits the desired control behaviour across a wide range of scenarios similar to those encountered during training.

Function approximators play a central role in machine learning methodologies, with artificial neural networks assuming a distinguished role. These networks are particularly effective for approximating complex functions in high-dimensional and large-scale datasets, making them indispensable tools for minimizing generalization error and enhancing model performance [3].

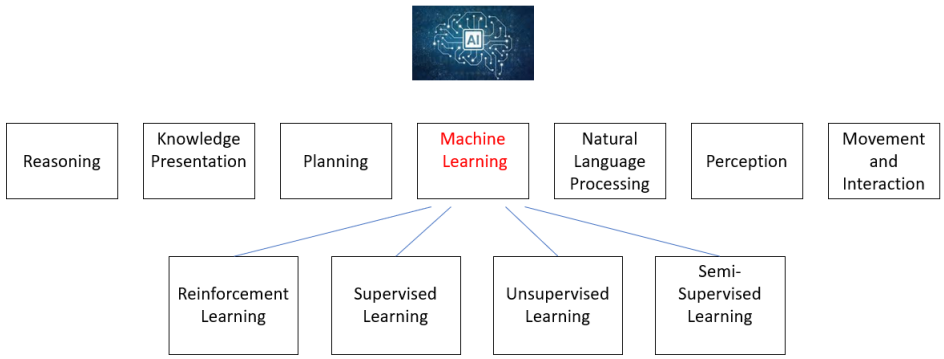


Figure 1.1: Hierarchical representation of the branches of artificial intelligence

1.2 Artificial Neural Networks

Artificial neural networks (ANNs) are inspired by biological neural networks (BNNs) and are conceptualized according to the Hebbian principle "neurons that fire together wire together" [6]. This succinct statement encapsulates the concept of plasticity in biological neural networks, where neurons responding to similar stimuli forge associations, leading to heightened responsiveness to these stimuli. In artificial neural networks, this concept is adopted and realized with weightings between the nodes. In addition to their basis in the Hebbian principle, ANNs also integrate mathematical models of learning and optimization, allowing them to simulate complex behaviours and solve intricate problems across various domains, ranging from image recognition to natural language processing. These networks have demonstrated remarkable success in mimicking cognitive functions, advancing fields such as supervised and reinforcement learning.

1.2.1 Different kinds of ANNs

The conceptualization of modern ANN elements traces back to 1943, when the perceptron was introduced [7]. In 1958, a hardware-software integrated approach was taken with the creation of the Mark I Perceptron, a machine designed for image classification [8], able to differentiate linearly separable images. The limitations of single-layer perceptrons were elucidated in 1969 with the proof of their inability to learn XOR logical functions [9]. However, the discovery that multi-layer perceptrons could approximate arbitrarily complex functions ignited a resurgence of interest in the field, leading to the proliferation of ANNs with diverse architectures and purposes. Notably, in reinforcement learning, multilayer perceptrons (MLPs) and convolutional neural networks (CNNs) have emerged as effective approximators for control and feature extraction functions, fostering a renewed enthusiasm for RL [10]. As of 2025, the effectiveness of large deep neural networks is evident in the realm of large language models (LLMs), exemplified by "ChatGPT," showcasing adept communication, problem-solving abilities, and broad domain knowledge [11]. Contemporary LLMs predominantly rely on the transformer architecture [12], typically possessing billions of parameters.

1.2.2 Description of MLP

The multilayer perceptron is the most commonly used ANN. It consists of nodes that are organised in layers $\mathbf{l}^{(i)}$ with a maximum number of L layers. Two special layers are the input layer (index $i = 0$) and the output layer (index $i = L$). The other layers are referred to as hidden layers. The layers have activation functions $\mathbf{a}^{(i)}(\cdot)$ that are

funneled by σ. The weights $\mathbf{w}^{(i)}$ and biases $\mathbf{b}^{(i)}$ constitute the parameters of layer i. The *activation of the layer* $\mathbf{l}^{(i)}$ is defined as

$$\mathbf{a}^{(i)}(\mathbf{x}) := \sigma(\mathbf{w}^{(i)}\mathbf{a}^{(i-1)}(\mathbf{x}) + \mathbf{b}^{(i)}),$$

where

$$\mathbf{a}^{(i-1)}(\mathbf{x}) = \sigma(\mathbf{w}^{(i-1)}\mathbf{a}^{(i-2)}(\mathbf{x}) + \mathbf{b}^{(i-1)}),$$

with input layer activation $\mathbf{a}^{(0)}(\mathbf{x}) := \mathbf{x}$. With this definition, the output of a MLP can be described by $f_\theta(\mathbf{x}) := \mathbf{a}^{(L)}(\mathbf{x})$. The goal is to find the parameters of f_θ to approximate a target function.

1.2.3 Training and Backpropagation

Fitting of a MLP refers to the process of tuning the parameters of a neural network to approximate a target y given x. With a certain amount of nodes per layer spanning the parameter space Θ, the goal is to find optimal parameters θ^* consisting of weights $\mathbf{w}$ and biases $\mathbf{b}$ that minimises the loss $\mathcal{L}$ by:

$$\theta^* = \underset{\theta\in\Theta}{\operatorname{argmin}}\{\mathbb{E}[\mathcal{L}(f_\theta(\mathbf{x}), y)]\}.$$

In some machine learning paradigms, the data is given by a finite set of pairs $T = \{(x_j, y_j)\}_{j\in J}$ that follow an underlying joint probability distribution $P(x, y)$ over $\mathcal{X}\times\mathcal{Y}$. Hence $\mathbb{E}$ denotes the expectation regarding that joint probability distribution.
The parameters are updated with gradient descent. Let $\mathbf{z}^{(i)} := \mathbf{w}^{(i)}\mathbf{a}^{(i-1)}(\mathbf{x}) + \mathbf{b}^{(i)}$, then the gradient for f_θ w.r.t. to the parameters of the last layer is obtained by

$$\nabla_{\mathbf{w}^{(L)}}\mathcal{L} = \nabla_{\mathbf{a}^{(L)}}\mathcal{L}\nabla_{\mathbf{z}^{(L)}}\mathbf{a}^{(L)}\nabla_{\mathbf{w}^{(L)}}\mathbf{z}^{(L)}$$

$$\nabla_{\mathbf{b}^{(L)}}\mathcal{L} = \nabla_{\mathbf{a}^{(L)}}\mathcal{L}\nabla_{\mathbf{z}^{(L)}}\mathbf{a}^{(L)}\nabla_{\mathbf{b}^{(L)}}\mathbf{z}^{(L)}$$

With $\nabla_{\mathbf{w}^{(L)}}\mathbf{z}^{(L)} := \mathbf{a}^{(L-1)}$,$\nabla_{b^{(L)}}\mathbf{z}^{(L)} := \mathbf{1}$ and $\sigma'(\mathbf{z}^{(L)}) := \nabla_{\mathbf{z}^{(L)}}\mathbf{a}^{(L)}$, the updates become

$$\mathbf{w}^{(L)} \leftarrow \mathbf{w}^{(L)} - \alpha\nabla_{\mathbf{a}^{(L)}}\mathcal{L}\sigma'(\mathbf{z}^{(L)})\mathbf{a}^{(L-1)}$$

and

$$\mathbf{b}^{(L)} \leftarrow \mathbf{b}^{(L)} - \alpha\nabla_{\mathbf{a}^{(L)}}\mathcal{L}\sigma'(\mathbf{z}^{(L)})$$

If $\mathbf{a}^{(L-1)}$ depends on parameters of deeper layers, the gradients must be further backtracked:

$$\nabla_{\mathbf{w}^{(L-1)}}\mathcal{L} = \nabla_{\mathbf{z}^{(L)}}\mathcal{L}\nabla_{\mathbf{a}^{(L-1)}}\mathbf{z}^{(L)}\nabla_{\mathbf{w}^{(L-1)}}\mathbf{a}^{(L-1)} \tag{1.1}$$

Expanding the terms results in

$$\nabla_{\mathbf{z}^{(L)}}\mathcal{L} = \nabla_{\mathbf{a}^{(L)}}\mathcal{L}\nabla_{\mathbf{z}^{(L)}}\mathbf{a}^{(L)},$$

$$\nabla_{\mathbf{a}^{L-1}}\mathbf{z}^{(L)} = \mathbf{w}^{(L)},$$

$$\nabla_{\mathbf{w}^{(L-1)}}\mathbf{a}^{(L-1)} = \nabla_{\mathbf{z}^{(L-1)}}\mathbf{a}^{(L-1)}\nabla_{\mathbf{w}^{(L-1)}}\mathbf{z}^{(l-1)}.$$

Simplifying the terms and plugging them in Equation (1.1) yields

$$\nabla_{\mathbf{w}^{(L-1)}}\mathcal{L} = \nabla_{\mathbf{a}^{(L)}}\mathcal{L}\nabla_{\mathbf{z}^{(L)}}\mathbf{a}^{(L)}\mathbf{w}^{(L)}\nabla_{\mathbf{z}^{(L-1)}}\mathbf{a}^{(L-1)}\mathbf{a}^{(L-2)}$$

The alterations are traced recursively until reaching the input layer. The process of propagating gradients back to the input layer is termed **backpropagation**. Backpropagation incurs significant computational expense. In practice, multiple pairs are aggregated into minibatches of size K. The loss is then computed as the average $\frac{1}{K}\sum_{j=1}^{K}\mathcal{L}(f_\theta(\mathbf{x}_j), y_j)$ over the minibatch, and a single gradient descent step is executed based on the mean loss. The concept of a multi-layer perceptron is summarised in Figure 1.2.

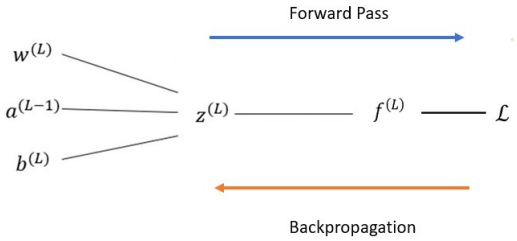


Figure 1.2: Visualizing elementary dynamics in multilayer perceptron: Forward pass (for inference) and backpropagation (for learning).

1.2.4 Application

Various types of ANNs serve as function approximators across machine learning paradigms. In supervised learning, ANNs are employed for tasks such as classification and regression. For instance, classification tasks encompass discerning between spam and non-spam emails [3] or identifying product irregularities in manufacturing using CNNs [13]. Regression problems may involve estimating the energy consumption in heavy industry [14]. In addition to traditional feedforward neural networks, large visual models (LVMs), notably implemented with transformers similar to the Vision Transformer

(ViT) [15], are pivotal in generative artificial intelligence. Generative models have demonstrated notable success in text-to-image generation [16] and voice cloning [17]. Moreover, ANNs are utilised as MLPs in reinforcement learning to approximate the control function.

1.3 Motivation, Goals and Contributions

First introduced at the Hannover Fair in 2011 [18], the concept of "Industry 4.0" represents a paradigm shift in production technology. This shift is characterized by the integration of connectivity, machine cognitivity, innovative human-machine interfaces, and versatile engineering technologies into the production chain. This transformative approach would enable mass production of highly customized products [19]. The proposed concepts of Industry 4.0 aim to revolutionize traditional manufacturing processes by creating intelligent systems capable of operating autonomously and effectively in complex production environments (see also the abstract).

The primary driver of the research presented in this thesis is to contribute to the machine cognitivity aspect of Industry 4.0, addressing the algorithmic challenges associated with generating sophisticated models for decision-making. Reinforcement learning (RL) algorithms, particularly their well-performing distributional variants, are promising avenues for establishing robust and versatile control models. Flexible control models are essential for navigating the complexities of modern industrial systems and are discussed in greater detail in Section 2.3. Distributional reinforcement learning (DistRL) algorithms, in particular, have demonstrated remarkable performance in solving decision-making problems within black-box contexts and under conditions of uncertainty. Their ability to capture the full distribution of potential rewards makes them particularly effective for tasks requiring nuanced decision-making. It is this capability that motivates the following research efforts.

A prerequisite for learning optimal control models is accurate machine perception. Effective decision-making depends on the availability of a clearly defined state. That in turn is possible only if all relevant information is accessible to the acting instance. Falling into the category of machine perception are sensory modalities such as vision, acoustics, and tactility. Machine vision stands out as particularly critical, since it often allows for the inference of additional information from visual data alone. Spatial and contextual cues obtained through vision can provide valuable information that would otherwise require supplementary sensory data, e.g. capacitance measurement for collision detection. By addressing challenges in machine cognitivity, this thesis

seeks to advance the state of the art in reinforcement learning and explores strategies to improve state identification through supervised learning methodologies (see Chapter 5). The ultimate goal is to develop a novel RL algorithm that not only excels in controlled experimental environments but also demonstrates potential for real-world applications in autonomous production facilities. The work's objectives are described in more detail in the next section.

1.3.1 Goals

The objective of this thesis is to contribute to the understanding of DistRL and develop a novel DistRL algorithm that ideally outperforms current state-of-the-art DistRL/RL algorithms. Ensemble learning methods are investigated as a complementary approach for Supervised Learning (SL) tasks. This section outlines the specific objectives that guide the research presented in the following chapters. It is well-established that DistRL algorithms often outperform their base variants. While various hypotheses have been proposed to explain this superior performance, a consensus among researchers has yet to be reached [20]. This work seeks to identify one or more factors responsible for the enhanced capabilities of DistRL algorithms. By addressing this gap in understanding, the operating principles of DistRL are further explored and results stemming from these efforts might form the foundation for further research in this domain. A key metric for improvement in the field of reinforcement learning, as defined in this thesis, is the ability of an algorithm to converge to a desired level of performance in less iterations compared to the current state-of-the-art approaches. To this end, the thesis begins by exploring the characteristics of existing DistRL methods and identifying promising research directions. Based on these insights, a working hypothesis for an improved DistRL algorithm is formulated. The proposed algorithm, called C-DSAC, is rigorously analyzed for its convergence properties, and its performance is thoroughly evaluated through comprehensive experiments in benchmark environments. Ideally, the novel algorithm will constitute an advancement over existing methods. While reinforcement learning is the central focus of this thesis, ensemble learning, a subset of supervised learning, is also explored as part of a tangential research direction. Specifically, ensemble methods are investigated for image classification tasks, with the aim of enhancing robotic vision using low-complexity models. This exploration serves as a precursor and potential auxiliary to state perception in robotics, complementing the primary research on DRL. Readers who wish to bypass the detailed discussions and analyses can find a concise summary of the findings in Chapter 3.6.

The structure of this thesis is organized as follows. Chapter 2 introduces RL,

entropy-augmented RL, and DistRL, while also exploring methods for DistRL implementation. Building on this foundation, Chapter 3 presents the primary contribution of this thesis: a novel entropy-augmented DistRL algorithm based on the Cramér metric, which demonstrates superior performance compared to current RL algorithms. Chapter 4 shifts the focus to model-based RL in robotics, proposing an innovative approach for efficiently generating control using inverse kinematics. The SL counterpart to Chapter 2 is discussed in Chapter 5, which establishes foundational concepts and methods. Chapter 6 explores a novel ensemble learning approach that bridges classical optimization and machine learning through linear programming. In contrast, Chapter 7 details another ensemble learning method, relying on neural networks and non-linear techniques. Finally, Chapter 8 concludes the thesis, summarizing the key findings and providing an outlook on future research directions in reinforcement learning.

1.3.2 Contributions

This thesis delivers four principal advances in distributional reinforcement learning (DistRL) and related supervised learning methods, culminating in both theoretical insights and practical algorithms for robotic control and perception. First, a rigorous theoretical framework for entropy-augmented, actor-critic DistRL under the Cramér metric is established. Building on an extensive literature review, it is proved that, under standard assumptions, algorithms developed in this framework converge to the optimal distributional state-action function and policy.

Second, guided by this theory, Cramér-based Distributional Soft Actor-Critic (C-DSAC) is introduced, a novel neural-network implementation that achieves **state-of-the-art performance** in benchmark robotic environments. Analysis reveals that the algorithm possesses a property termed confidence-driven model updates, which contributes to its superior performance. The results of this analysis deepen the understanding of why DistRL algorithms frequently outperform their expectation-based counterparts.

The contributions of this work are extended to specialized robotic applications. "Reinforcement Learning - Inverse Kinematics" (RL-IK) is presented, a model-based approach that decouples task planning from low-level control by integrating a robot's kinematic model into the learning loop. Empirical results show that RL-IK not only accelerates task learning but also produces smoother, more natural robotic movements.

Finally, as a complementary investigation in supervised learning, we propose a novel boosting algorithm that unites column-generation techniques from classical optimization with neural-network–based classifiers. While this method's immediate practical

impact is modest, it bridges methodological gaps between optimization theory and modern learning frameworks, paving the way for more research at the intersection of these fields.

2 Introduction to Reinforcement Learning Algorithms

2.1 Reinforcement Learning

In science and technology, deep reinforcement learning algorithms have been successfully applied to various problems. A notable example is AlphaTensor [21]; a framework/agent exhibiting superhuman performance for automating algorithmic discoveries in matrix multiplication. Reinforcement learning [5] is used in contemporary large language models for fine-tuning and serves as the primary method to eliminate undesired responses from the model [22]. In robotic tasks, this approach has been explored for data-driven generation of optimal control [23]. Nevertheless, applying reinforcement learning presents significant challenges, primarily due to the high sample complexity and the limited amount of data available in real-world scenarios [24]. What has begun as a promising approach to learn discrete control policies [10], and has since evolved to address general problems and mitigate various limitations [25, 26, 27, 28, 29, 30], appears to have reached a developmental standstill. A more recent approach by [20, 31], which expands and formalizes the distributional perspective on reinforcement learning, has revitalized interest in the field, including among researchers from other disciplines [32].

2.1.1 Basic Concepts

2.1.1.1 Introduction to MDP

A Markov Decision Process (MDP) models decision-making in dynamic environments [33]. It consists of a set of states $\mathcal{S}$, actions $\mathcal{A}$, a reward function R, and a transition kernel P, which governs how actions affect the environment [34]. In this work, MDPs are assumed to have the Markov Property, namely that "the probability distribution of

future states of the process conditioned on both the past and present states depends only on the present state" [35]. A policy π selects actions to maximize an objective in this dynamic setting, with the assumption of full observability. This means the resulting next state is always known. The objective is tied to the reward function and can involve maximizing the total reward over a finite horizon, maximizing the cumulative discounted reward, or maximizing the average reward over time. The discounted reward is the most widely studied, as it adjusts the importance of immediate versus future rewards, thus influencing an agents planning behaviour. Note that as the future reward is discounted, variance is reduced at the cost of introducing bias.

2.1.1.2 Problem Formulation

In reinforcement learning, the interaction of an agent with its environment is modelled as a MDP: $(\mathcal{S}, \mathcal{A}, r, P, \gamma)$. In this context, $\mathcal{S}$ is the state space, $\mathcal{A}$ denotes the action space, $r : \mathcal{S} \times \mathcal{A} \mapsto [r_{min}, r_{max}]$ is the bounded reward function, $P : \mathcal{S} \times \mathcal{A} \mapsto Pr(\mathcal{S})$[1] is the transition probability density function and $\gamma \in [0, 1]$ denotes the discount factor. At each time step t, an agent executes an action $a_t \in \mathcal{A}$ from state $s_t \in \mathcal{S}$ according to a policy π and receives a reward $r(s_t, a_t)$, transitioning into the next state $s_{t+1} \sim P(\cdot \,|\, s_t, a_t)$. An episodic MDP considers to reach a terminal state at T within finite time. In this work, the policy is treated as a stochastic quantity, modelled as a probability distribution over the action space $\pi : \mathcal{S} \mapsto Pr(\mathcal{A})$. The action is sampled from the policy, i.e. $a_t \sim \pi(\cdot \,|\, s_t)$. When a random variable is explicitly sampled from a distribution, the random variable is written in small letters. It is assumed that the policy π is ergodic and thus there exists a well-defined discounted state-action marginal distribution of the trajectory induced by π which is expressed as $\rho^{\gamma}_{\pi}(s_t, a_t)$. The general goal is to find an optimal policy π^* that maximizes the expected discounted return along a trajectory of interactions:

$$J(\pi) := \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right], \tag{2.1}$$

i.e., $\pi^* = \text{argmax}_{\pi \in \Pi} J(\pi)$; an optimal policy π^* must not be unique. The objective can be expressed in more detail and in terms of a discounted state-action marginal stationary distribution ρ^{γ}_{π}:

$$J(\pi) := \sum_{t=0}^{T} \mathbb{E}_{(s_t, a_t) \sim \rho^{\gamma}_{\pi}}\left[\sum_{\ell=t}^{T} \gamma^{\ell - t} \mathbb{E}_{s_\ell \sim P, a_\ell \sim \pi}[r(s_\ell, a_\ell)]\right], \tag{2.2}$$

[1] The set of all probability measures on a measurable space (Ω, Σ) is denoted by $Pr(\Omega)$.

Note that $J(\pi)$ is the discounted expected reward along the trajectory originating from every state-action tuple (s_t, a_t).

The episode has a length of T and thus J optimizes a discounted return in a finite MDP. Unlike an infinite-horizon MDP, where an upper bound of J is the maximum value of the geometric series ad infinitum, $\max_\pi J(\pi) \leq \frac{1}{1-\gamma} r_{max}$, an upper bound of J on a finite MDP is $\max_\pi J(\pi) \leq \frac{1-\gamma^T}{1-\gamma} r_{max}$. This poses a potential problem as future rewards might not be accounted for. This problem can be avoided by adjusting the discount factor to align with the average episode length. To determine the discount factor, heuristics such as a hyperparameter search or adherence to established values can be employed [36]. In that way, accurate value estimations across different settings can be ensured.

Optimal policies can be approximated by policy iteration, a process that incorporates policy evaluation and improvement using Q-values, see Lemma 2.4.

2.1.2 Frameworks

2.1.2.1 Critic Only

A Q-value induced by a policy π is denoted by $Q^\pi(s_t, a_t)$. In critic-only methods, the (greedy) policy is derived from the value function by

$$\pi(s) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q(s, a). \tag{2.3}$$

As a result, the superscript for the policy will be omitted. The Q-value is the return for a state-action pair (a_t, s_t),

$$Q(s_t, a_t) := \mathbb{E}_t \left[\sum_{k=t}^{T} \gamma^{k-t} r(s_k, a_k) \right]. \tag{2.4}$$

The optimal Q-value Q^* is a solution of the Bellman optimality equation, i.e. the solution of

$$Q^*(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P}[\max_a Q^*(s_{t+1}, a)] \tag{2.5}$$

for the state-action pair (s_t, a_t). Based on (2.5), we define the Bellman optimality operator $\mathcal{T}$ by

$$\mathcal{T}Q(s_t, a_t) := r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P} \left[\max_a Q(s_{t+1}, a) \right]. \tag{2.6}$$

The one-step update rule thus becomes

$$Q(s_t, a_t) \leftarrow (1 - \alpha) Q(s_t, a_t) + \alpha (r(s_t, a_t) + \gamma [\max_a Q(s_{t+1}, a)] - Q(s_t, a_t)), \tag{2.7}$$

where α is the learning rate. The one-step temporal difference error $\delta := r(s_t, a_t) + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t)$ is relevant for the main algorithm of this work presented in Section 3.2. For optimal policy evaluation, it can be shown that the sequence of value functions $Q_{k+1} := \mathcal{T}Q_k$, starting from some arbitrary Q_0, will exponentially quickly converge to Q^* as k increases.

Lemma 2.1. Let $Q_1(s, a)$ and $Q_2(s, a)$ be two state-action value functions. Then $\mathcal{T}$ according to (2.6) is a γ-contraction, i.e.

$$\|\mathcal{T}Q_1 - \mathcal{T}Q_2\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty. \tag{2.8}$$

Proof. To show that (2.8) holds, we expand the left-hand side. Be reminded that $s_{t+1} \sim P(s_t, a_t)$. Then

$$\begin{aligned}
&\|\mathcal{T}Q_1 - \mathcal{T}Q_2\|_\infty \\
&= \| \left(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P} \left[\max_a Q_1(s_{t+1}, a) \right] \right) - \left(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P} \left[\max_a Q_2(s_{t+1}, a) \right] \right) \|_\infty \\
&= \sup_{s_t \in \mathcal{S}, a_t \in \mathcal{A}} |\gamma \mathbb{E}_{s_{t+1} \sim P} \left[\max_a Q_1(s_{t+1}, a) \right] - \gamma \mathbb{E}_{s_{t+1} \sim P} \left[\max_a Q_2(s_{t+1}, a) \right] | \\
&= \gamma \sup_{s_t \in \mathcal{S}, a_t \in \mathcal{A}} |\mathbb{E}_{s_{t+1} \sim P} \left[\max_a Q_1(s_{t+1}, a) \right] - \mathbb{E}_{s_{t+1} \sim P} \left[\max_a Q_2(s_{t+1}, a) \right] | \\
&\leq \gamma \sup_{s_t \in \mathcal{S}, a_t \in \mathcal{A}} \max_a \mathbb{E}_{s_{t+1} \sim P} | \left[Q_1(s_{t+1}, a) \right] - \left[Q_2(s_{t+1}, a) \right] | \\
&\leq \gamma \sup_{s \in \mathcal{S}, a \in \mathcal{A}} |Q_1(s, a) - Q_2(s, a)| = \gamma \|Q_1 - Q_2\|_\infty.
\end{aligned}$$

Since $\mathcal{T}^*$ is a γ-contraction, it follows from the Banach theorem that the series $\{Q_k\}$ with $Q_{k+1} := \mathcal{T}Q_k$ converges to a unique fixed point Q^*, i.e. $Q = \mathcal{T}Q$ and $||Q_k - Q^*||_\infty \to 0$ as $k \to \infty$. □

In practice, Q^* (and therefore π^*) is unknown and must be iteratively improved through exploration, such as with an ϵ-greedy approach. With probability ϵ, a random action is chosen, allowing updates to $\max_a Q(s_t, a)$. In Q-learning (see Section 2.1.6.1), these maximum values are stored in a tabular fashion. Convergence is guaranteed if the state-action values are updated and exploration of the environment occurs under GLIE assumptions, see Section 2.1.2.3.

The state value V^π is similar to the aforementioned state-action value induced by π and can be expressed in terms of the latter in the critic-only context as

$$V^\pi(s_t) = \mathbb{E}_\pi[Q^\pi(s_t, a_t)]. \tag{2.9}$$

Analogous to the state-action approach, the Bellman equation and operator can be defined for the state value function. In this context, properties and convergence of policy evaluation, improvement and iteration can be proven similarly. The value function is more relevant to on-policy algorithms, see Section 2.1.2.6.

2.1.2.2 Temporal Difference Learning and Advantage Estimator

Temporal Difference (TD) learning is a method to learn the value of a state or state-action pair from observations without requiring a model of the environment. TD learning generalizes concepts from Monte Carlo learning and dynamic programming. In $TD(1)$, returns are estimated based on the trajectory of an entire episode. Unlike Monte Carlo methods, $TD(1)$ can be applied to discounted continuing tasks, i.e, learning can occur prior to episode termination. Using the off-policy state-action formulation and $G_t := \sum_{t=0}^{T} \gamma^t r(s_t, a_t)$, the $TD(1)$ update rule can be expressed as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(G_t - Q(s_t, a_t)),$$

where α denotes the step size. The counterpart to $TD(1)$ is $TD(0)$, where the Q-value is updated considering only the immediate future. An example for $TD(0)$ is given in the update rule (2.7). $TD(0)$ is widely used in reinforcement learning and is a foundational method for dynamic programming. Sutton introduced $TD(\lambda)$ in [5], allowing to strike a balance between $TD(0)$ and $TD(1)$. Defining the off-policy return for n-steps in the setting of state-action values as

$$G_{t:t+n} := r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \ldots + \gamma^n \max_a Q(s_{t+n}, a), \quad 0 \leq t \leq T - n, \tag{2.10}$$

and the λ-return as

$$G_t^\lambda := (1 - \lambda) \sum_{n=1}^{T-t} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t} G_t,$$

the $TD(\lambda)$ update can be written as

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha(G_t^\lambda - Q(s_t, a_t)).$$

Here, λ is the trace-decay parameter, $0 \leq \lambda \leq 1$. Equation (2.10) indicates that the update rule (effectively) reduces to $TD(0)$ and $TD(1)$ with $\lambda = 0$ and $\lambda = 1$, respectively. Note also that a high value for λ reduces bias, but increases variance whereas a low value reduces variance but increases bias. Most notably, $TD(\lambda)$ was used in 1995 to learn to play Backgammon [37]. Analoguous methods for managing the bias-variance trade-off in policy updates exist and are discussed in Section 2.1.2.5.

2.1.2.3 Direct Policy Search

In contrast to critic-only methods described in Section 2.1.2.1, the class of actor-only reinforcement learning algorithms aims to directly optimize the parameters of the policy for maximizing the return. Thus, a value function is not needed. In reinforcement learning, where environment dynamics are unknown, direct policy search simplifies the problem to a purely stochastic one. Assuming some kernel P underlying the transitions and a stochastic policy parameterized by ϕ, the objective to maximize is a function of the policy parameters ϕ only, i.e.

$$J(\phi) = \mathbb{E}_{\pi_\phi}\left[\sum_{t=k}^{T} \gamma^{t-k} r(s_t, a_t)\right]. \tag{2.11}$$

This stochastic problem can be solved by various methods (evolutionary approach, cross-entropy, ..., etc.) of which only the gradient-based method is relevant for this work. The parameters of the policy are updated with gradient update rule [38]

$$\phi \leftarrow \phi + \alpha \nabla_\phi J. \tag{2.12}$$

According to the simplified policy gradient theorem, assuming discrete spaces and writing the choice of actions explicitly, the gradient becomes

$$\begin{aligned}
&\nabla_\phi J(\phi) \\
&= \nabla_\phi \sum_{a\in\mathcal{A}} \sum_{t=k}^{T} \pi_\phi(a \mid s_t) \gamma^{t-k} r(s_t, a) \\
&= \sum_{a\in\mathcal{A}} \sum_{t=k}^{T} \nabla_\phi \pi_\phi(a \mid s_t) \gamma^{t-k} r(s_t, a) \\
&= \sum_{a\in\mathcal{A}} \sum_{t=k}^{T} \nabla_\phi \pi_\phi(a \mid s_t) \gamma^{t-k} r(s_t, a) \\
&= \sum_{a\in\mathcal{A}} \sum_{t=k}^{T} \pi_\phi(a \mid s_t) \frac{\nabla_\phi \pi_\phi(a \mid s_t)}{\pi_\phi(a \mid s_t)} \gamma^{t-k} r(s_t, a) \\
&\text{Since } \frac{d}{dx} \log f(x) = \frac{f'(x)}{f(x)} \\
&= \sum_{a\in\mathcal{A}} \sum_{t=k}^{T} \pi_\phi(a \mid s_t) \nabla_\phi \log \pi(a \mid s_t) \gamma^{t-k} r(s_t, a) \\
&= \mathbb{E}_{\pi_\phi}\left[\sum_{t=k}^{T} \nabla_\phi \log \pi_\phi(a \mid s_t) \gamma^{t-k} r(s_t, a)\right].
\end{aligned} \tag{2.13}$$

The simplified policy gradient theorem ignores the fact that state distributions depend on the policy parameters. In fact, [39] showed that the simplified policy gradient theorem does not optimize Equation (2.11). To illustrate this problem, consider again Equation (2.11) and take into account the state distribution discount as shown by [38]:

$$J(\phi) = \sum_{s \in \mathcal{S}} d^{\gamma}_{\pi_\phi}(s) \sum_{a \in \mathcal{A}} \sum_{t=k}^{T} \pi_\phi(a \,|\, s_t) \gamma^{t-k} r(s_t, a) \,|\, s_k = s, \tag{2.14}$$

where $d^{\gamma}_{\pi_\phi}(s) = \sum_{t=0}^{\infty} \gamma^t \mathbb{P}(s_t = s \,|\, s_0, \pi_\phi)$ and $\mathbb{P} : \mathbb{R} \mapsto [0, \infty)$ is a generic probability density function. This formulation shows that discounted state distributions depend on the policy parameters. Despite the technical issues posed by the simplified policy gradient theorem, most if not all reinforcement learning algorithms, including those mentioned in this work, use it for policy optimization due to its empirical success.

Gradient-based direct policy search has strong convergence properties due to the ones it inherits from gradient descent methods [40]. To guarantee convergence to (a local) maximum, J must be continuously differentiable, ∇J must satisfy Lipschitz conditions and the "Greedy in the Limit with Infinite Exploration (GLIE)" requirements must be met. The GLIE conditions are expressed as

$$\begin{aligned} \sum_{t=0}^{\infty} \alpha_t &= \infty \\ \sum_{t=0}^{\infty} \alpha_t^2 &= \infty. \end{aligned} \tag{2.15}$$

However, the gradient $\nabla_\phi J$ can exhibit high variance depending on the length of the trajectory due to the stochastic nature of J. A direct policy algorithm will be introduced in Section 2.1.6.3.

2.1.2.4 Actor-Critic

Actor-critic (AC) methods incorporate a policy and a value function, called actor and critic, respectively. The policy and value functions are dependent but separate structures. AC algorithms can handle continuous action spaces and reduce variance (potentially at the cost of higher bias) due to the critic. The policy is updated according to the estimate of the critic and the policy gradient in (2.13) becomes

$$\nabla_\phi J(\phi) = \mathbb{E}_{\pi_\phi}[\nabla_\phi \log \pi_\phi(a_t|s_t) Q^{\pi_\phi}(s_t, a_t)]. \tag{2.16}$$

An analysis reveals however that the actor-critic policy gradient in Equation (2.16) is an approximation of the true AC policy gradient:

$$\begin{aligned}
&\nabla_\phi J(\phi) \\
&= \nabla_\phi \sum_{a\in\mathcal{A}} \sum_{t=k}^{T} \pi_\phi(a \mid s_t) Q^{\pi_\phi}(s_t, a) \\
&= \sum_{a\in\mathcal{A}} \sum_{t=k}^{T} \nabla_\phi [\pi_\phi(a \mid s_t) Q^{\pi_\phi}(s_t, a)] \\
&= \sum_{a\in\mathcal{A}} \sum_{t=k}^{T} \nabla_\phi \pi_\phi(a \mid s_t) Q^{\pi_\phi}(s_t, a) + \pi_\phi(a \mid s_t) \nabla_\phi Q^{\pi_\phi}(s_t, a) \\
&\approx \sum_{a\in\mathcal{A}} \sum_{t=k}^{T} \nabla_\phi \pi_\phi(a \mid s_t) Q^{\pi_\phi}(s_t, a)
\end{aligned} \tag{2.17}$$

The last step is justified by [41] and it can be shown that the approximated gradient still converges to a true local minimum. The rest of the derivation follows the scheme described in (2.13) and yields the formulation in (2.16). The critic is updated by evaluating the current policy. The state-action value function induced by a policy π is defined as

$$Q^\pi(s_t, a_t) := \mathbb{E}_\pi \left[\sum_{k=t}^{T} \gamma^{k-t} r(s_k, a_k) \right] \tag{2.18}$$

and satisfies the Bellman equation

$$Q^\pi(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim P, a_{t+1}\sim\pi}[Q^\pi(s_{t+1}, a_{t+1})]. \tag{2.19}$$

Based on (2.19), we define the Bellman operator $\mathcal{T}^\pi$,

$$\mathcal{T}^\pi Q(s_t, a_t) := r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim P, a_{t+1}\sim\pi} \left[Q(s_{t+1}, a_{t+1}) \right]. \tag{2.20}$$

Analogous to the proof in Section 2.1.2.1, we can show evaluation contraction of policy π.

Lemma 2.2. Let $Q_1(s, a)$ and $Q_2(s, a)$ be two state-action value functions.Then $\mathcal{T}^\pi$ according to (2.20) is a γ-contraction, i.e.

$$\|\mathcal{T}^\pi Q_1 - \mathcal{T}^\pi Q_2\|_\infty \leq \gamma \|Q_1 - Q_2\|_\infty. \tag{2.21}$$

Proof. To show that (2.21) holds, we expand the left-hand side. Be reminded that $s_{t+1} \sim P(s_t, a_t)$. Then

$$
\begin{aligned}
&\|\mathcal{T}^\pi Q_1 - \mathcal{T}^\pi Q_2\|_\infty \\
&= \| \left(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim P}[Q_1(s_{t+1}, \pi(s_{t+1}))]\right) - \left(r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}\sim P}[Q_2(s_{t+1}, \pi(s_{t+1}))]\right) \|_\infty \\
&= \sup_{s_t\in\mathcal{S}, a_t\in\mathcal{A}} |\gamma \mathbb{E}_{s_{t+1}\sim P}[Q_1(s_{t+1}, \pi(s_{t+1}))] - \gamma \mathbb{E}_{s_{t+1}\sim P}[Q_2(s_{t+1}, \pi(s_{t+1}))]| \\
&= \gamma \sup_{s_t\in\mathcal{S}, a_t\in\mathcal{A}} |\mathbb{E}_{s_{t+1}\sim P}[(Q_1(s_{t+1}, \pi(s_{t+1})) - Q_2(s_{t+1}, \pi(s_{t+1})))]| \\
&\leq \gamma \sup_{s_t\in\mathcal{S}, a_t\in\mathcal{A}} \mathbb{E}_{s_{t+1}\sim P}[|Q_1(s_{t+1}, \pi(s_{t+1})) - Q_2(s_{t+1}, \pi(s_{t+1}))|] \\
&\leq \gamma \sup_{s\in\mathcal{S}} |Q_1(s, \pi(s)) - Q_2(s, \pi(s))| \\
&\leq \gamma \sup_{s\in\mathcal{S}, a\in\mathcal{A}} |Q_1(s, a) - Q_2(s, a)| = \gamma \|Q_1 - Q_2\|_\infty
\end{aligned}
$$

We conclude that, since $\mathcal{T}^\pi$ is a γ-contraction, it follows from the Banach theorem that the series $\{Q_k\}$ with $Q_{k+1} := \mathcal{T}^\pi Q_k$ converges to a unique fixed point Q^π, i.e. $Q^\pi = \mathcal{T}^\pi Q^\pi$ and $\|Q_k - Q^\pi\|_\infty \to 0$ as $k \to \infty$. □

Policy improvement involves exploitation of knowledge about the environment and updating towards higher values, i.e.

$$\pi_{k+1}(s_t) = \underset{\pi\in\Pi}{\operatorname{argmax}}\, \mathbb{E}_{a_t\sim\pi} Q^{\pi_k}(s_t, a_t). \tag{2.22}$$

Lemma 2.3. Given the policy update rule in (2.22), then

$Q^{\pi_k}(s_t, \pi_k(s_t)) \leq Q^{\pi_{k+1}}(s_t, \pi_{k+1}(s_t)) \forall s_t \in \mathcal{S}$.

Proof. The proof relies on expanding the Q-function in the Bellman equation repeatedly with actions of the improved policy. For convenience, we define $\pi' := \pi_{k+1}$ and $\pi := \pi_k$ and shorten the subscripts of $\mathbb{E}$ for long trajectories. We use the notation $\mathbb{E}_{P,\pi'} = \mathbb{E}_{s_{t+1}\sim P, a_{t+1}\sim\pi'}$.

$$
\begin{aligned}
&Q^{\pi}(s_t, \pi(s_t)) \leq Q^{\pi}(s_t, \pi'(s_t)) \\
&= \mathbb{E}_{a_t \sim \pi', s_{t+1} \sim P}[r(s_t, a_t) + \gamma Q^{\pi}(s_{t+1}, \pi(s_{t+1}))] \\
&\leq \mathbb{E}_{a_t \sim \pi', s_{t+1} \sim P}[r(s_t, a_t) + \gamma Q^{\pi}(s_{t+1}, \pi'(s_{t+1})] \\
&= \mathbb{E}_{a_t \sim \pi', s_{t+1} \sim P}[r(s_t, a_t) + \gamma \mathbb{E}_{a_{t+1} \sim \pi', s_{t+2} \sim P}[r(s_{t+1}, a_{t+1}) + \gamma Q^{\pi}(s_{t+2}, \pi(s_{t+2}))]] \\
&= \mathbb{E}_{\pi', P}[r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 Q^{\pi}(s_{t+2}, \pi(s_{t+2}))] \\
&\leq \mathbb{E}_{\pi', P}[r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \gamma^3 Q^{\pi}(s_{t+3}, \pi(s_{t+3}))] \\
&\vdots \\
&\leq \mathbb{E}_{\pi', P}[r(s_t, a_t) + \gamma r(s_{t+1}, a_{t+1}) + \gamma^2 r(s_{t+2}, a_{t+2}) + \gamma^3 r(s_{t+3}, a_{t+1}) + \dots] \\
&= Q^{\pi'}(s_t, a_t).
\end{aligned}
$$

Thus $Q^{\pi}(s_t, a_t) \leq Q^{\pi'}(s_t, a_t)$. □

In that manner, the policy will strictly be non-decreasing. The estimation of the state-action value can be improved by better exploration of the state-space.

The Q-function approximator is parameterized by θ and is denoted as Q^{π}_{θ}. Policy iteration is the method of alternating between evaluation and improvement. It can be shown that policy iteration converges towards the optimal value function and an optimal policy. Although its relevance is not exclusive to AC methods, in AC algorithms it can be shown explicitly due to the separation of the acting and evaluating instances.

Lemma 2.4. Under the assumptions that the reward is bounded, $|\mathcal{A}| < \infty$ and $|\mathcal{S}| < \infty$, repeatedly alternating between policy evaluation and policy improvement from any arbitrary policy $\pi \in \Pi$, converges to an optimal policy π^* according to (2.1) and an optimal state-action value function Q^*.

Proof. The assumptions dictate that there are a finite number of policies. From policy improvement we know that $Q^{\pi_{k+1}} \geq Q^{\pi_k}$ for the series $\{\pi_k\}$, which is bounded above by $Q^* = Q^{\pi^*}$. At convergence, π^* must be an optimal policy according to (2.1), since it can not be further improved. □

2.1.2.5 Advantage and Generalized Advantage Estimator

The advantage is defined as the difference between the value obtained by executing a particular action from a state and an (approximated) baseline value for that particular state under policy π [42]. Formally it can be expressed as

$$A^{\pi}(s_t, a_t) := Q^{\pi}(s_t, a_t) - V^{\pi}(s_t). \tag{2.23}$$

The term "advantage" reflects how beneficial a particular action is compared to default behaviour. The selection of the advantage-based policy gradients is simply justified: Policy gradients calculated with Equation 2.24 yield low variance [43].

Replacing the critic-based policy gradient by a advantage-based one yields

$$\nabla_\phi J(\phi) = \mathbb{E}_{\pi_\phi}[\nabla_\phi \log \pi_\phi(a_t \mid s_t) A^\pi(s_t, a_t)]. \tag{2.24}$$

In practice, $A^\pi(s_t, a_t)$ is not known and when approximated with $\hat{Q}^\pi$ and $\hat{V}^\pi$ in a $TD(0)$-context, introduces high bias. Let $\hat{A}(s_t, a_t) = \hat{Q}^\pi(s_t, a_t) - \hat{V}^\pi(s_t)$ denote the approximated advantage and $\delta^{\hat{V}}(s_t, a_t) = r(s_t, a_t) + \gamma \hat{V}^\pi(s_{t+1}) - \hat{V}^\pi(s_t)$ the TD-error in terms of the approximated value function for the next state. Then analogous to TD-Learning for value functions, the multi-step approximated advantage can be described by

$$\begin{aligned} \hat{A}^{(\ell)}(s_t, a_t) &:= \sum_{k=0}^{\ell-1} \gamma^k \delta^{\hat{V}^\pi}(s_{t+k}, a_{t+k}) \mid \pi \\ &= -\hat{V}^\pi(s_t) + r(s_t, a_t) + \ldots + \gamma^{\ell-1} r(s_{t+\ell-1}, a_{t+\ell-1}) + \\ &\quad \gamma^\ell \hat{V}^\pi(s_{t+\ell}) \mid \pi \end{aligned} \tag{2.25}$$

Note that if $\ell = T$, the bias becomes 0. Summing over the advantages for $(1) \ldots (T)$ yields the generalized advantage estimator (GAE)

$$\begin{aligned} \hat{A}^{GAE}(s_t, a_t) &:= (1-\lambda)\left(\hat{A}^{(1)}(s_t, a_t) + \lambda \hat{A}^{(2)}(s_t, a_t) \ldots\right) \mid \pi \\ &= \sum_{k=0}^{T} (\gamma\lambda)^k \delta^{\hat{V}^\pi}(s_{t+k} a_{t+k}) \mid \pi \end{aligned} \tag{2.26}$$

Then, replacing the theoretical advantage function in Equation (2.24) with the GAE, we obtain the GAE policy update. By introducing the parameter λ, the bias-variance trade-off can be controlled. A high value of λ reduces bias and increases variance and vice-versa. Notably, if $\lambda = 1$ and the ℓ-step advantage estimator in (2.25) sums either to termination or infinity, it is equal to the GAE. The advantage function is most commonly used in on-policy algorithms, see Section 2.1.2.6. One such algorithm will be discussed in Section 2.1.6.4.

2.1.2.6 On- and Off-Policy Learning

Let μ denote the **behaviour policy**, which is **any** policy used to generate experiences, and let π denote the policy being updated during the learning process. In on-policy

algorithms, the behaviour policy and the policy coincide, i.e., $\mu = \pi$. This is the case because experiences generated by π are disregarded after the policy is updated with these experiences.

In off-policy algorithms (distinct from offline RL, which relies on pre-generated experiences), π is optimized but only generates the most recent experiences. Off-policy methods store experiences in a replay buffer, a mutable set of μ-experiences iteratively expanded by π, and learn from them. In other words, the policy is updated with experiences generated by μ.

A related concept is the target policy, which can refer to a (greedy) policy derived from π by moving average updates. Unlike the policy, the target policy does not generate new experiences and is used solely for learning the value function. Similarly, the value function itself may have a target counterpart.

2.1.2.7 Multi-Agent Reinforcement Learning

Due to its increasing popularity, the reader shall be informed about the existence of multi-agent reinforcement learning (MARL). Multi-agent methods are applied when an acting system can be divided into smaller, independently or autonomously operating agents. Examples include UAV swarm control for mapping and flexible energy grid management, see [44, 45]. Agents interact within a shared environment. In cooperative settings, agents collaborate towards a shared goal. For instance, the agents control the members of a multi-joint robot, acting interdependently to reach the same goal as in coordinating a robot to pick up an object. In a competitive setting, it is required that only one agent reaches the goal and the agents try to outperform each other. Mixed settings can involve agents with varying objectives, such as controlling chess pieces where each agent suggests moves, with an overarching policy selecting the final action. MARL solutions are generally robust due to the optimized behaviour across multiple agents. They support parallel computation, and experience sharing can lead to more thorough state-space exploration. However, MARL also introduces unique challenges, such as coordinating agent communication, assigning credit in cooperative tasks, and scalability issues.

2.1.3 Challenges

2.1.3.1 Exploration-Exploitation Dilemma

Reward associated with transitions must be learned by thoroughly searching the state space and requires testing new actions to gain more knowledge about the environment.

Exploration involves actively seeking out less-known areas of the state space, enabling the learning process to refine the association between actions and resulting rewards. By contrast, *exploitation* uses the present knowledge of the environment to choose actions believed to yield the highest expected reward, based on current, potentially incomplete information. Reinforcement learning faces the challenge of balancing exploitation and exploration [46]. Theoretically, convergence to an optimal policy requires adherence to GLIE conditions (see Equation 2.15), ensuring that all actions in all states are sufficiently sampled over time. However, this approach is often impractical in real-world applications, especially when dealing with high-dimensional, continuous state/action spaces and sparse reward functions (see Section 2.1.4). Brute-force exploration becomes infeasible in such environments, even with policies gradually increasing in greediness.

In practice, reinforcement learning algorithms employ heuristics such as the ϵ-greedy method, widely used in algorithms like DQN (see Section 2.1.6.1), and stochastic policies as used in SAC or PPO (see Sections 2.2.2 and 2.1.6.4). Heuristics add instability to reinforcement learning algorithms as they rely heavily on parameter initialization and offer no convergence guarantee.

2.1.3.2 Overestimation Bias in Q-learning

Overestimation of state-action values is a systemic issue in Q-based reinforcement learning algorithms when function approximators are used. First identified by [47] in 1993, overestimation bias remains challenging to mitigate effectively due to the computational expense of current solutions. To illustrate the overestimation bias, consider the RHS of an aggressive update rule (TD-updates are omitted for clarity):

$$\hat{Q}(s_t, a_t) \leftarrow r(s_t, a_t) + \gamma \max_a \hat{Q}(s_{t+1}, a),$$

where $\hat{Q}(s_{t+1}, a_{t+1}) := Q^*(s_{t+1}, a_{t+1}) + \epsilon$, Q^* is the true Q value and ϵ is a random noise term. Then the bias is caused by the difference

$$\begin{aligned}
\Delta(s_t, a_t) &= r(s_t, a_t) + \gamma \max_a \hat{Q}(s_{t+1}, a) - (r(s_t, a_t) + \gamma \max_a Q^*(s_{t+1}, a)) \,|\, s_{t+1} \sim P \\
&= \gamma(\max_a \hat{Q}(s_{t+1}, a) - \max_a Q^*(s_{t+1}, a)) \\
&= \gamma(\max_a [Q^*(s_{t+1}, a) + \epsilon] - \max_a Q^*(s_{t+1}, a)). \qquad (2.27)
\end{aligned}$$

With $\mathbb{E}[\epsilon] = 0$, it holds that the bias $\mathbb{E}_{P,\epsilon}[\Delta(s_t, a_t)] \geq 0$. W.l.o.g., if ϵ is uniformly distributed in $[-v, v]$, the max operator tends to select actions with positive noise values, causing overestimation. This effect can lead to premature convergence to a local

maximum with inflated value estimates. A common heuristic to counteract overestimation bias is presented in Section 2.1.5.1. This work presents an algorithm that further mitigates the effect of overestimation at low computational cost.

2.1.3.3 Large Spaces and Complexity

Sample complexity refers to experience instances required for an algorithm to converge to an optimal policy. This view is typically under worst-case assumptions to define an upper bound on computational requirements. For instance, analysis may be based on assuming discrete spaces, as continuous spaces can be approximated as large discrete ones. The algorithms presented here achieve optimality via (Howardian) policy iteration (PI), which combines policy evaluation and improvement with discounting, see [48]. Scherrer (2013) showed that the upper bound on the number of policy iterations is $\mathcal{O}(\frac{|\mathcal{A}|}{1-\gamma}\log\frac{1}{1-\gamma})$, see [49]. The per-iteration cost is comprised of the cost of evaluation and improvement. For every state, the Bellman condition (2.19) has to be satisfied, yielding a linear system solvable in polynomial time at $\mathcal{O}(|\mathcal{S}|^3)$. Consider a constructive form of the policy improvement step (2.22):

$$\pi(s_t) = \underset{a}{\operatorname{argmax}}\, r(s_t, a) + \gamma \mathbb{E}_\pi[Q^\pi(s_{t+1}, a_{t+1})]$$

A naive approach might iterate over all state-action pairs, making the complexity $\mathcal{O}(|\mathcal{S}|^2|\mathcal{A}|)$. Thus, the upper bound of the per-iteration complexity for PI becomes

$$\mathcal{O}(|\mathcal{S}|^3 + |\mathcal{S}|^2|\mathcal{A}|).$$

An improved version of PI evaluates the policy partially in every k steps, reducing the complexity to

$$\mathcal{O}(k|\mathcal{S}|^2 + |\mathcal{S}|^2|\mathcal{A}|).$$

In high-dimensional environments, such as in the humanoid environment with $\mathcal{S}^d; d = 374$ and $\mathcal{A}^d, d = 17$ (Section 3.5), a naive approach to PI is computationally prohibitive. Instead, (deep) neural networks with high-generalization capabilities and sophisticated techniques are used in training for managing complexities. A novel neural-based reinforcement learning algorithm is presented in this work that can be applied to large and continuous observation and action spaces.

2.1.3.4 Sim-to-Real Gap

Training reinforcement learning models in simulations is often preferred, as it allows faster-than-real-time processing and parallelization. A task in a simulated environment

can be solved analytically provided that a model is at hand (and reasonably costly to engineer) and the dynamics are implemented accurately. The former is usually not obtainable. In fact, simulating contact-rich interactions such as encountered in robotic assembly is not a trivial task and requires highly specialized software. These software systems can be used as a reliable surrogate only within rigid boundaries [50]. The sim-to-real gap refers to the mismatch between simulated and real-world transition dynamics, where the simulation fails to fully replicate the complexities of the physical environment. Model-based reinforcement learning addresses this by iteratively learning both the control (or policy) and a model of the environment. This approach is gaining attention, notably as part of the "Alberta Plan", an AI initiative led by Richard Sutton aiming for Artificial General Intelligence (AGI) with reinforcement learning principles as core components [51]. It is important to remember that results in this work, i.e. control models for various simulated tasks, can not be directly used in a real setting due to the sim-to-real gap.

2.1.4 Reward Engineering and Value Alignment

Defining a reward function that elicits desired behaviour in real-life settings is a significant challenge and warrants separate discussion due to the critical issues it can present. This section will go on a tangent about social and moral problems and introduces the concept of value alignment.

2.1.4.1 Paradigm Challenge

In reinforcement learning, the objective is to identify a policy that maximizes return, Equation (2.11), making the reward function a key determinant of an agent's post-training **behaviour**. In that context, the concept of reward density is introduced. A reward function that provides feedback at each transition is considered a maximal dense reward function, whereas an environment that emits one reward only after the desired behaviour is exhibited has the lowest possible reward density. The density of the reward function significantly impacts learning efficacy. Ideally, an agent learns desired behaviour from sparse feedback, but in practice, sparse reward makes training difficult, often hindering convergence to a local maximum. Crafting a dense reward function requires extensive domain knowledge and human input, which can resemble supervised learning and conflict with autonomous learning goals. Challenges in defining effective guiding signals are ongoing. Solutions include reward-free learning with human guidance through example states and intrinsically motivated reinforcement learning,

where agents explore for the sake of discovery rather than for solving a problem [52, 53]. However, these alternatives come with their own limitations. A promising approach to address reward-based reinforcement learning involves using foundational models, such as LLMs, to automate guidance by either writing reward functions for a specific task or labelling states directly, offering potential for fully automatic reinforcement learning. The next sections discuss issues arising from the paradigm challenge.

2.1.4.2 Technical Problem: An example from a Pick and Place Task

Assuming that training with a sparse reward function is computationally tractable, the problem of avoiding unwanted behaviour from a low-resolution guidance signal remains: How to avoid learning unwanted behaviour without over-specifying the feedback? The following example illustrates that the answer to this question is not trivial and simple intuitive solutions can be misleading. Consider a robotic task as illustrated in Figure 2.1.

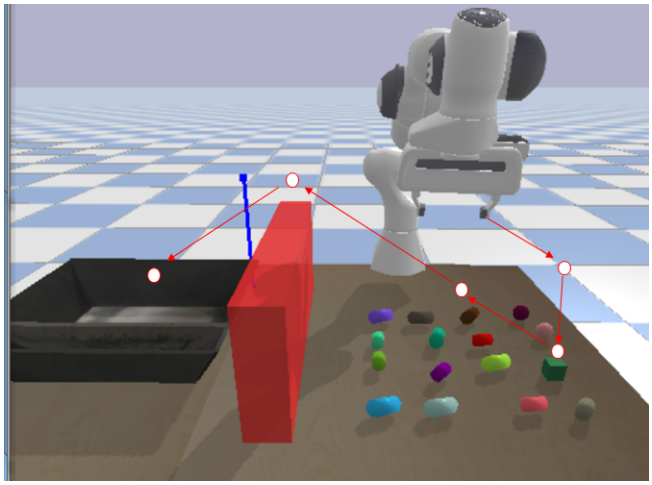


Figure 2.1: PyBullet simulation of a pick and place task with the Franka Emika Panda robot. The robot must pickup a target object and place it inside the tray while avoiding collision with the environment. The robot is allowed to execute 8 collision-free movements before the episode terminates.

The environment includes a target object to be placed in a tray, a robotic agent, and static objects forming the surroundings. The action space consists of spatial points and Eulerian orientation coordinates, see Chapter 4. The agent has up to 8 moves to complete the task, with n as the current number of moves taken. The episode terminates prematurely if the agent collides with the environment.

Let the reward function $r(s,a)$ incorporate penalties and rewards for collision, distance to the object, goal-reaching, and execution time. Define $\upsilon_a(s,a)$ as the robots body position vertices and $\upsilon_e(s,a)$ as static object vertices. Then $col(s,a) := -c\mathbb{I}(\upsilon_a(s,a) \cap \upsilon_e(s,a) \neq \emptyset)$, where c is a collision penalty and $\mathbb{I}$ is the indicator function. The full reward function is given by

$$r(s,a) := col(s,a) - d(\mathbf{p}_a, \mathbf{p}_o|s,a) + \mathbb{I}_{s_g}(s,a) + t(s,a),$$

where s_g is the goal state and $t(s,a) := -n * (1 - \mathbb{I}s_g(s,a))$ can be interpreted as a control cost; it encourages task completion with minimal movements. The functions appears intuitive, punishing collisions and excessive moves while rewarding reaching the goal and proximity to the target object. However, it unexpectedly leads the agent, after initial attempts to pick up the object, to collide with the closest edge of the red wall. In attempting to pick up the object, the agent, lacking initial dexterity, frequently collides with the table and incurs a penalty of $-c$. Additionally, repeated failed attempts increase the control cost, leading the agent to learn that it can avoid cumulative penalties by terminating the episode quickly. The quickest termination occurs through collision with the nearest object, the wall. Since the agent can not reach the goal state s_g if the target object is not picked up, the reward for s_g does not affect the agents "suicidal" behaviour. While this issue is easily diagnosed in a simple environment, identifying reward function flaws in more complex settings with numerous objects or agents becomes intractably difficult. Training environments must incorporate guiding signals that align the agent's behaviour with human intentions.

2.1.4.3 Moral Problem and Value Alignment

Beyond the technical issues in reinforcement learning, the value alignment problem is critical for ensuring safe human-machine coexistence [54]. Consider a scenario in which a humanoid robot is ordered to fetch a cup of coffee in the kitchen. Let "Fiona" be a pet cat that blocks the entrance to the kitchen. If the robot is solely rewarded for prompt delivery, it might take harmful actions against Fiona, a behaviour that is impermissible for (most) humans. Then it is said that the machine "ethics" is not aligned with human values [55]. Especially in reinforcement learning, the value alignment problem might be severe, as unwanted behaviour can emerge from seemingly reasonable rewards.

One proposed solution is to enhance guiding signals with a comprehensive, empirically tested "ethics library" to ensure ethical behaviour when agents interact with humans, animals, or property. Value alignment remains a critical challenge in machine learning, compounded by concerns about the unpredictability of "superintelli-

gence"—hypothetical entities with intellectual abilities far exceeding human intelligence. Although speculative at the time of writing, prominent researchers are actively exploring methods to control such entities to ensure their goals align with the users intend [56].

2.1.5 RL Algorithm Augmentations

In reinforcement learning, different techniques are used to address the challenges described in Section 2.1.3. The following sections detail methods applicable across algorithm classes or all algorithms.

2.1.5.1 Double Q-Learning

The double-Q method addresses the problem discussed in Section 2.1.3.2. The main idea is to decouple the selection of actions from evaluation of the state-action pair [25]. Let θ and θ' be the parameters of the critic and its target variant (Section 2.1.2.6). Consider the update rule in (2.7) with parameterized approximators and let the estimated label be

$$Y(s_t, a_t) := r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P}[\max_a Q_\theta(s_{t+1}, a)].$$

Then the double-Q method utilizing online and target critics can be expressed as

$$Y^D(s_t, a_t) := r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P}[Q_{\theta'}(s_{t+1}, \operatorname{argmax}_a Q_\theta(s_{t+1}, a))]. \tag{2.28}$$

In the AC framework (Section 2.1.2.4), the double-Q method modifies the critic update slightly: The critic is updated with the minimum of two Q-value estimators and uses their mean for updating the actor. Despite its simplicity, double-Q effectively reduces overestimation bias, providing a significant performance boost. This method is integrated into C-DSAC, see Section 3.2.

2.1.5.2 Normalization Techniques

Neural networks perform better with inputs in a consistent range, meaning enhancement in generalization and convergence, see [57, 58]. In reinforcement learning, reward and observation normalization adjust actor and critic inputs to improve stability. One common method is min-max scaling, which rescales feature values x_i in the vector $\mathbf{x} \in \mathcal{X}$ (either observation or concatenated observation and action) to $[0, 1]$. Let

$x_{min} := \min_{\mathbf{x}\in\mathcal{X}} x_i$ and $x_{max} := \max_{\mathbf{x}\in\mathcal{X}} x_i$, then the rescaled feature value is defined by

$$x_i' = \frac{x_i - x_{min}}{x_{max} - x_{min}}.$$

Min-max normalization does not handle outliers well. An alternative popular method, Z-Score Standardization, manages outliers by centering features around 0 and adjusting for variability. Let μ_i be the mean and σ_i the standard deviation of feature i over $\mathcal{X}$, then the Z-score of a feature value is

$$x_i' = \frac{x_i - \mu_i}{\sigma_i}.$$

When normalization is applied for the activation of neural layers, it is known as Batch Normalization [59]. In reinforcement learning, the available observations in $\mathcal{X}$ increase with the current iteration number N. In the case of max-min scaling, obtaining the new extrema is trivial. However, updating the metrics for the Z-score involves softly updating the mean by $\mu_{i,N} = (1 - \frac{1}{N})\mu_{i,N-1} + \frac{1}{N}x_i$ or using Welfords methods for the standard deviation. The algorithm presented in Section 3.2 suffers from training instability and application of normalization techniques might mitigate this issue.

2.1.5.3 Prioritized Experience Replay

As discussed earlier, off-policy algorithms store the experience tuples in a replay buffer $\mathcal{B}$. In the standard case, these experiences are sampled uniformly for learning. Prioritized Experience Replay (PER), however, increases the likelihood of sampling transitions with a higher TD-error, as these are expected to offer greater learning progress, see [60].

PER defines priority for a transition i either as $p_i := |\delta_i| + \epsilon$ or as $p_i := \frac{1}{rank(i)}$, where the function $rank$ sorts the experiences according to magnitude of their TD-error. Then the sampling probability becomes

$$P(i) = \frac{p_i^\eta}{\sum_k p_k^\eta},$$

where $\eta \in [0, 1]$ modules the effect of priority; $\eta = 0$ reverts to uniform sampling. The algorithms works by sampling a mini-batch according to P and updating the priorities of the samples in the batch. To make accurate estimates of expected values, estimate updates should match the approximated distribution. This is not the case in PER and thus additional bias is introduced which is adjusted by weighting the TD-error with

$$w_i = \left(\frac{1}{|\mathcal{B}|P(i)}\right)^\beta,$$

where $\beta \in [0, 1]$ determines the degree of bias compensation. It is recommended to anneal β to 1 [60]. PER has shown substantial performance gains in discrete critic-only algorithms and improved performance for the deterministic AC algorithms described in Section 2.1.6.5. The novel reinforcement learning algorithm C-DSAC proposed in this work might benefit from PER as it is relatively unstable in the training phase.

2.1.5.4 Initialization and Change of Neural Network Parameters

Neural networks learn by adjusting parameters through backpropagation, see Section 1.2.3. In machine learning, the way these parameters are handled before and during training is crucial. As the name implies, mini-batch stochastic descent is inherently random and might therefore lead to unstable training or prevent convergence altogether. Therefore, neural network parameter initialization matters; the accuracy for the classification tasks described in Section 6.4 can not be achieved if the Glorot uniform initialization scheme [61] is exchanged by a naive Gaussian scheme. Gradient explosion and vanishing are issues where gradients become excessively large or small. The Glorot initialization scheme aims to prevent these phenomena by maintaining the same activation variances across all layers of the neural network, resulting in balanced weights. The forward pass and backpropagation necessitates consideration to the number of both input and output weights. Let $n^{(\ell)}$ and $n^{(\ell+1)}$ be the number of inputs and outputs of a neural network layer respectively; the mean provides a balanced compromise for both directions. The variance should therefore be $Var(\mathbf{w}) = \frac{2}{n^{(l)}, n^{(l+1)}}$ to ensure that the accumulated effect of the weights do not lead to excessive activations. The variance of a uniform distribution $\mathcal{U}(a, b)$ is given by $Var = \frac{1}{12}(b-a)^2$. Setting one standard deviation as a target and substituting the target in the uniform distribution, the k-th node of layer ℓ is sampled according to

$$w_{\ell,k} \sim \mathcal{U}\left(-\sqrt{\frac{6}{n^{(\ell)} + n^{(\ell+1)}}}, \sqrt{\frac{6}{n^{(\ell)} + n^{(\ell+1)}}}\right),$$

with $\mathbb{E}[\mathcal{U}] = 0$. As discussed in Section 2.1.5.2, having activations closer to 0 is favourable and therefore the Normal Glorot initialization scheme

$$w_{\ell,k} \sim \mathcal{N}\left(0, \sqrt{\frac{2}{n^{(\ell)} + n^{(\ell+1)}}}\right)$$

leads to more stable training and higher generalizations. In this work, all neural network-based methods utilize the Normal Glorot scheme for initialization. Besides initialization, the changes of the parameters are also of interest. Especially in RL,

where the cost function is approximated, monitoring the gradients is advised. Some algorithms like PPO (Section 2.1.6.4) have gradient control as their main feature: the advantage function and the ratio of the pre- and post-update action probabilites are used to clip the gradient. Other algorithms like TRPO [27] subject the objective to a Kullback-Leibler constraint, making sure that the gradients do not steer the model outside of a "trusted region".

2.1.5.5 Learning Rate Annealing

The learning rate, or step size for model updates, is an essential yet often underestimated factor in reinforcement learning, with significant implications for convergence [62]. As a matter of fact, the novel confidence-driven concept presented in Section 3.4.1 is related to learning rate control. Learning rate annealing, the systematic reduction of this rate over time, plays a critical role in balancing exploration and exploitation throughout training. At the start of training, a high learning rate promotes exploratory actions and helps the model investigate the state space more broadly, capturing diverse potential strategies. As training progresses and promising regions are identified, a gradual reduction in the learning rate allows for a more careful, refined exploration of these areas, improving convergence to optimal policies. Several annealing strategies facilitate this progression. Linear annealing decreases the learning rate incrementally from an initial value to a minimum threshold, promoting a controlled shift from exploration to exploitation. Exponential decay of the learning rate is particularly effective when the discount factor approaches one, ensuring stability in long-horizon tasks where future rewards are highly valued [62]. Additionally, periodic annealing methods, such as Cosine Annealing, reset the learning rate to a higher value after it reaches its minimum, allowing the model to periodically reintroduce exploratory behaviour, which can prevent premature convergence to suboptimal policies.

2.1.6 Relevant Algorithms

To further illustrate the concepts discussed earlier, this section introduces various reinforcement learning algorithms, each emphasizing different aspects of reinforcement learning concepts. Some of these algorithms also serve as the foundation for more advanced methods.

2.1.6.1 Tabular Q-learning

The classical Q-learning algorithm of Watkins is a tabular critic-only (Section 2.1.2.1) reinforcement learning algorithm [63]. This approach involves maintaining a Q-value entry for each possible state-action pair (s_t, a_t), representing the expected return for that pair (2.4). The Q-value entries are updated iteratively utilizing the Bellman optimality operator (2.6) that performs an $TD(0)$-update. The goal is to find a function that satisfies equation (2.5) for all state-action pair entries. During training, the action is selected by an ϵ-greedy policy, i.e. at each time step, a random action is selected with probability ϵ, and the greedy option (2.3) according to the best current Q-estimate with probability $(1-\epsilon)$. This ensures that each action is eventually taken, allowing the Q-values to converge to optimal values under the GLIE condition described by (2.15). Typically, ϵ is initialized at 1, prioritizing exploration in early episodes, and linearly reduced as learning progresses. Tabular Q-learning is computationally feasible only for small or discrete state-action spaces, as it requires storage for each pair individually. For larger or continuous spaces, function approximation or other methods must be employed.

2.1.6.2 Deep Q-Learning

Following a period of stagnation in reinforcement learning research, DeepMind revitalized interest in this field by presenting an AI system that exceeded human performance in Atari 2600 games [10, 64]. This breakthrough was achieved using a deep neural network within the Q-learning framework, with MLPs approximating the Q-function; CNNs process the raw game pixels for extraction of abstract features from the high-dimensional state-space. In Algorithm 2.1, the CNN is represented by ϕ. The objective is to minimize the mean squared error between approximated targets and predicted Q-values for the current state-action pairs. Assuming a neural network approximator for the Q-function parameterized by θ, the loss function becomes

$$J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\rho_\mu}\left[\frac{1}{2}(\mathcal{T}Q(s_t,a_t) - Q_\theta(s_t,a_t))^2\right], \tag{2.29}$$

where ρ_μ is the marginal distribution overstate-action pairs induced by the behaviour policy. In practice, this expectation is approximated by sampling mini-batches of experience tuples (s_t, a_t, r_t, s_{t+1}) from a replay buffer $\mathcal{B}$ and computing the sample mean. The Q-network parameters, θ, are updated based on the gradient of the loss:

$$\nabla_\theta J_Q(\theta) = \mathbb{E}_{(s_t,a_t)\sim\rho_\mu}[(\mathcal{T}Q(s_t,a_t) - Q_\theta(s_t,a_t))\nabla_\theta Q_\theta(s_t,a_t)] \tag{2.30}$$

DQN stores past experiences as tuples in an experience replay buffer Section 2.1.2.6. During training, these experiences are randomly sampled from the buffer. Tuples containing single transitions are sampled uniformly from the replay buffer, reducing temporal correlations between the experiences. Given that DQN serves as the foundation for many Q-based algorithm in reinforcement learning, the complete DQN procedure is outlined in Algorithm 2.1. Due to the *max*-operator for a greedy action selection, only small action spaces can be managed by DQN, thus rendering it suitable only for discrete (and small) spaces. In fact, the game *Venture* might have possibly biggest action space consisting of only 18 discrete actions. Section 2.1.6.5 introduces a Q-based AC algorithm handling continuous actions.

Algorithm 2.1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{B}$ to capacity $|\mathcal{B}|$
Initialize action-value function Q with weights θ
Initialize learning parameter β_Q,
$\mathcal{B} \leftarrow \emptyset$
repeat
    Receive initial observation state $\phi_1 = \phi(s_t)$ from features s_1
    for each environment step **do**
        With probability ϵ, select a random action a_t
        Otherwise, select $a_t = \max_a Q_\theta(\phi(s_t), a)$
        Observe transition $s_{t+1} \sim P(s_t, a_t)$
        Observe reward r_t
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
    for each gradient step do **do**
        Sample a random minibatch of N transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{B}$
        Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ \hat{\mathcal{T}} Q_\theta(\phi_j, a_j) & \text{for non-terminal } \phi_{j+1} \end{cases} \quad j = 0, \dots N$
        Calculate approximated loss $\hat{J}_Q$ from y
        $\theta \leftarrow (1 - \beta_Q)\theta + \beta_Q \hat{\nabla}_\theta J_Q$ from approximated loss
until Stopping criterion
return Q_θ

2.1.6.3 REINFORCE

REINFORCE, short for "Reward Increment = Nonnegative Factor × Offset Reinforcement × Characteristic Egilibility" is a foundational on-policy algorithm that performs a direct policy search by optimizing the policy objective (2.11) for the parameters of a stochastic policy [65]. As one of the earliest policy gradient methods, REINFORCE has played a critical role in establishing techniques used in more advanced policy optimization algorithms. The REINFORCE algorithm operates based on Monte Carlo sampling, where updates occur only after an episode has terminated - mirroring $TD(1)$-updates. The update gradient is expressed as

$$\nabla_\phi J(\phi) = \mathbb{E}_{\pi_\phi}\left[\sum_{t=0}^{T} \nabla_\phi \log \pi_\phi(a_t \mid s_t) \left[\sum_{k=t}^{T} \gamma^k r(s_k, a_k)\right]\right].$$

Note that the log-trick is applied as in Section 2.1.2.3. The post-update parameters are computed using the gradient:

$$\phi_{t+1} = \phi_i + \alpha \mathbb{E}_{\pi_\phi}\left[\sum_{t=0}^{T} \nabla_\phi \log \pi_\phi(a_t \mid s_t) \left[\sum_{k=t}^{T} \gamma^k r(s_k, a_k)\right]\right].$$

The update rule for REINFORCE is obtained by substituting the term inside the expectation into (2.12) for $\nabla_\phi J(\phi)$. REINFORCE, being a Monte Carlo method, often exhibits high variance in gradient estimates, which can hinder stable learning. A common modification involves introducing a baseline value — typically a state value function V^π — to reduce variance, leading to a gradient formulation

$$\nabla_\phi J(\phi) = \mathbb{E}_{\pi_\phi}\left[\sum_{t=0}^{T} \nabla_\phi \log \pi_\phi(a_t \mid s_t) \left(\left[\sum_{k=t}^{T} \gamma^k r(s_k, a_k)\right] - V^\pi(s_t)\right)\right].$$

With hindsight to (2.25), the above equation can be re-written more compactly in terms of the estimated advantage function

$$\nabla_\phi J(\phi) = \mathbb{E}_{\pi_\phi}\left[\sum_{t=0}^{T} \nabla_\phi \log \pi_\phi(a_t \mid s_t) \hat{A}^T(s_t, a_t)\right]. \tag{2.31}$$

The derivation of the update rule is trivial and follows the same scheme as discussed above. "Baselines" are important concepts that will be used in more sophisticated methods. The full procedure is described in Algorithm 2.2 [66].

Algorithm 2.2 REINFORCE with Value Function

1: Initialize ϕ
2: **for** episode $\{s_0, a_0, r(s_0, a_0) \dots r(s_T, a_T)\} \sim \pi_\phi, M$ **do**
3: **for** $t = 0, T$ **do**
4: $\phi \leftarrow \phi + \alpha \nabla_\phi \log \pi_\phi(a_t \mid s_t) A^\pi(s_t, a_t)$

2.1.6.4 PPO

Proximal Policy Optimization (PPO), an Actor-Critic (AC) algorithm, is a popular on-policy approach that leverages stochastic policies and can incorporate the Generalized Advantage Estimator (GAE) [28]. Noted for its robustness and relative insensitivity to hyperparameter tuning, PPO is straightforward to implement and can solve all environments examined in this work. However, while effective, it is generally less sample-efficient compared to off-policy methods and does not achieve the same level of performance as the novel C-DSAC algorithm presented in Section 3.2. PPO employs a Conservative Policy Iteration (CPI) objective, designed to stabilize learning by bounding policy updates. Using GAE as discussed in Section 2.1.2.5, the CPI actor objective to be maximized is defined as follows:

$$J^{CPI}(\phi) := \mathbb{E}_{(s_t,a_t)\sim\rho_{\pi_{\phi_{old}}}} \left[\frac{\pi_\phi(a_t \mid s_t)}{\pi_{\phi_{old}}(a_t \mid s_t)} \hat{A}^{GAE}(s_t, a_t) \right], \tag{2.32}$$

where $\kappa_{t,\phi} := \kappa_\phi(s_t, a_t) := \frac{\pi_\phi(a_t \mid s_t)}{\pi_{\phi_{old}}(a_t \mid s_t)}$ measures the post-update parameter significance and $\rho_{\pi_{\phi_{old}}}$ is the state-action marginal distribution under $\pi_{\phi_{old}}$ which is not calculated on basis of the state distribution discounting. The central feature of PPO is its clipped objective that balances the incentive to improve the policy with a safeguard against excessive updates towards positive advantages given below.

$$J^{(CLIP)}(\phi) := \mathbb{E}_{(s_t,a_t)\sim\rho_{\pi_{\phi_{old}}}} \left[\min \left(\kappa_{t,\phi} \hat{A}^{GAE}(s_t, a_t), clip(\kappa_{t,\phi}, 1-\epsilon, 1+\epsilon) \hat{A}^{GAE}(s_t, a_t) \right) \right]. \tag{2.33}$$

If the advantage is positive with a high probability ratio, the right term is selected for the update; in case of a low probability ratio, the left term is chosen. The mechanism works in the opposite direction for negative advantages: If the probability is low, the clipped term is used and otherwise the left is selected. By taking the minimum of the clipped and unclipped terms, PPO ensures less adaptation to positive experiences and more adaptation to negative ones, making it a "pessimistic" algorithm. In general, PPO-agents learn from trajectory segments. This necessitates the following:

1. An ℓ-step GAE $\hat{A}^{GAE(\ell)}(s_t, a_t)$ which aggregates the $\gamma-$ and λ-discounted temporal-difference errors up to a chosen horizon within each episode.

2. State-value estimates at each time step, approximated by a parameterized critic as introduced in (2.9).
3. The critic itself can be updated with the methods described in Section 2.1.2.2. However, since the ℓ-step GAE is employed, the target value can be expressed in terms of

$$V_{tar}(s_t) := V^{\pi}(s_t) + \hat{A}^{GAE(l)}(s_t, a_t) \,|\, a_t \sim \pi_{\phi_{old}}.$$

Given the non-discounted state distribution $\varsigma_{\pi_{\phi_{old}}}$ induced by the old policy, the objective of the critic to be minimized becomes

$$J_V(\theta) := \mathbb{E}_{s_t \sim \varsigma_{\pi_{\phi_{old}}}}[(V_\theta(s_t) - V_{tar}(s_t))^2]. \tag{2.34}$$

In PPO, exploration is further encouraged by an entropy term, yielding a modified policy objective

$$J^{(CLIP+H)} = J^{(CLIP)} + \mathbb{E}_{s_t \sim \varsigma_{\pi_{\phi_{old}}}}[\mathcal{H}(\pi_\phi(\cdot \,|\, s_t))], \tag{2.35}$$

akin to the entropy-augmented approach in Soft Actor-Critic (SAC) and the newly developed C-DSAC algorithm. The complete PPO framework for multiple actors is outlined in Algorithm 2.3, where N actors generate trajectories, calculate GAE, and optimize both actor and critic with the clipping and entropy-augmented losses.

Algorithm 2.3 PPO, Actor-Critic Style

Initialize ϕ, θ and learning rate β
for episode$= 1, M$ **do**
  for actor $= 1, N$ **do**
    Run policy $\pi_{\theta_{\text{old}}}$ in environment for T timesteps
    Compute advantage estimates $\hat{A}(s_t, a_t)), \ldots, \hat{A}(s_{t+l}, a_{t+1})$
  $\phi \leftarrow (1-\beta)\phi + \beta \nabla_\phi J^{CLIP+H}$ # Gradient based on (2.35)
  $\theta \leftarrow (1-\beta)\theta + \beta \nabla_\theta J_V$ # Gradient based on (2.34)
return V_θ, π_ϕ

2.1.6.5 DDPG

Deep Deterministic Policy Gradient (DDPG) [26] is an off-policy, actor-critic reinforcement learning algorithm that extends Deep-Q Learning (Section 2.1.6.2) for continuous action spaces by leveraging ideas from Deterministic Policy Gradient (DPG) [67]. In DDPG, the policy is deterministic, mapping states directly to actions via a function

$\eta : \mathcal{S} \mapsto \mathcal{A}$. This deterministic nature separates the exploration problem from the learning process. In order to encourage exploration, noise is added to the action of the deterministic policy:

$$\tilde{\eta}(s_t) := \eta_\phi(s_t) + \epsilon \,|\, \epsilon \sim \mathcal{N}.$$

The authors in [26] use the Ornstein-Uhlenbeck process (OU process) to sample temporally correlated noise for exploration. Later implementations have shown that the rather complex noise function can be replaced by a Gaussian distribution without compromising the exploration quality [68]. As an off-policy algorithm, DDPG learns from experiences generated by a behaviour policy, which is stochastic due to the added exploration noise. Analogous to (2.19), the state-action value function $Q^\eta(s_t, a_t)$ for a deterministic policy satisfies the Bellman equation:

$$Q^\eta(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P}[Q^\eta(s_{t+1}, \eta(s_{t+1}))]. \tag{2.36}$$

The critic is updated by minimizing a loss function based on the Bellman operator for the deterministic policy

$$J_Q(\theta) := \mathbb{E}_{s_t \sim \varsigma_\mu, a_t \sim \mu}\left[\frac{1}{2}(\mathcal{T}^\eta Q(s_t, a_t) - Q_\theta(s_t, a_t))^2\right], \tag{2.37}$$

where

$$\mathcal{T}^\eta Q(s_t, a_t) := r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P}\left[Q(s_{t+1}, \eta(s_{t+1}))\right]$$

and ς is the state distribution induced by the stochastic behaviour policy. The gradient of the critic loss is thus computed as

$$\nabla_\theta J_Q(\theta) = \mathbb{E}_{s_t \sim \varsigma_\mu, a_t \sim \mu}\left[(\mathcal{T}^\eta Q(s_t, a_t) - Q_\theta(s_t, a_t))\nabla_\theta Q_\theta(s_t, a_t)\right]. \tag{2.38}$$

The actor is optimized using deterministic policy gradients derived by [67]. As usual, the state distribution discounting is not accounted for,

$$\nabla_\phi J_\pi(\phi) = \mathbb{E}_{s_t \sim \rho_\mu}[\nabla_a Q_\theta(s_t, \eta_\phi(s_t))\nabla_\phi \eta_\phi(s_t)]. \tag{2.39}$$

This formulation can be intuitively understood by considering the deterministic policy as a special case of a stochastic policy (2.17) with zero variance. In practice, DDPG uses deep neural networks to approximate both the deterministic policy and the Q-function, with parameters ϕ and θ, respectively. Due to the limitations of sample-based estimation and the fixed input size of neural networks, gradients $\hat{\nabla}_\phi J_\pi(\phi)$ and $\hat{\nabla}_\theta J_Q(\theta)$ are approximated from mini-batches of data sampled from the replay buffer. Target networks parameterized by $\bar{\phi}$ and $\bar{\theta}$ are employed to stabilize training, as described in Section 2.1.2.6.

The original DDPG paper proposed an Ornstein-Uhlenbeck process for temporally correlated noise during exploration. However, later implementations demonstrated that simpler Gaussian noise often suffices without degrading exploration performance. The complete DDPG scheme is summarized in Algorithm 2.4

Algorithm 2.4 DDPG Algorithm

Initialize θ, $\bar{\theta}$, ϕ, $\bar{\phi}$
Initialize learning parameters β_Q, β_π
$\mathcal{B} \leftarrow \emptyset$
repeat
    Initialize a random UO-process $\mathcal{N}$
    Receive initial observation state s_1
    for each environment step **do**
        $a_t := \eta_\phi(s_t) + \epsilon \,|\, \epsilon \sim \mathcal{N}$
        Observe transition $s_{t+1} \sim P(s_t, a_t)$
        Observe reward r_t
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
    for each gradient step do **do**
        Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from $\mathcal{B}$
        $\theta \leftarrow (1 - \beta_Q)\theta + \beta_Q \hat{\nabla}_\theta J_Q$ approximated from (2.38)
        $\phi \leftarrow (1 - \beta_\pi)\phi + \beta_\pi \hat{\nabla}_\phi J_\pi$ approximated from (2.39)
        Update target networks: $\bar{\theta} \leftarrow \tau\theta + (1 - \tau)\bar{\theta}$; $\bar{\phi} \leftarrow \tau\phi + (1 - \tau)\bar{\phi}$
until Stopping criterion
return $Q_{\bar{\theta}}, \pi_{\bar{\phi}}$

2.2 Maximum-Entropy Reinforcement Learning

Novel concepts in this work are rooted in maximum-entropy reinforcement learning, a framework known for its robustness and superior performance in complex, high-dimensional tasks. The notation in this section reflects the theoretical basis in the actor-critic scheme.

2.2.1 Problem Formulation

In the maximum-entropy setting, the objective in (2.2) is augmented with the policy entropy to improve exploration

$$J_h(\pi) := \sum_{t=0}^{T} \mathbb{E}_{(s_t,a_t)\sim\rho_\pi} \left[\sum_{\ell=t}^{T} \gamma^{\ell-t} \mathbb{E}_{s_\ell\sim P, a_\ell\sim\pi}[r(s_\ell, a_\ell) + \alpha\mathcal{H}(\pi(\cdot \mid s_\ell))] \right], \tag{2.40}$$

and the aim is to find an optimal policy $\pi^* = \operatorname{argmax}_{\pi\in\Pi} J_h(\pi)$ [30]. The parameter α represents the impact of policy stochasticity on the state-action value.

2.2.2 Soft Actor-Critic Framework

Analogous to the definition of the Bellman operator for standard reinforcement learning, a modified Bellman operator $\mathcal{T}_h^\pi$ is defined in the maximum entropy setting. Given the expected soft state-action value

$$V_h(s_t) = \mathbb{E}_{a_t\sim\pi}[Q(s_t, a_t) + \alpha\mathcal{H}(\pi(\cdot \mid s_\ell))],$$

$\mathcal{T}_h^\pi$ is expressed as

$$\mathcal{T}_h^\pi Q(s_t, a_t) := r(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim P}[V_h(s_{t+1})]. \tag{2.41}$$

Policy entropy can be calculated by $\mathcal{H}(\pi(\cdot|s)) := -\log\pi(a|s)$. Due to the negative sign, smaller probabilities increase the value. Similar to $\mathcal{T}^\pi$, the operator $\mathcal{T}_h^\pi$ is a γ-contraction, see Lemma 2.5.

Lemma 2.5. Let $Q_1(s, a)$ and $Q_2(s, a)$ be two state-action value functions. If $|\mathcal{A}| < \infty$, then the soft Bellman operator $\mathcal{T}_h^\pi$ according to (2.41) is a γ-contraction

$$\|\mathcal{T}_h^\pi Q_1 - \mathcal{T}_h^\pi Q_2\|_\infty \leq \gamma\|Q_1 - Q_2\|_\infty. \tag{2.42}$$

Proof. Defining $r_h(s_t, a_t) := r(s_t, a_t) + \mathbb{E}_{s_{t+1}\sim P}[\mathcal{H}(\pi(\cdot \mid s_{t+1}))]$ and noting that $\mathcal{T}_h^\pi Q(s_t, a_t) = r_h(s_t, a_t) + \gamma\mathbb{E}_{s_{s+t}\sim P, a_{t+1}\sim\pi}[Q^\pi(s_{t+1}, a_{t+1})]$, we can apply the proof in Lemma 2.2 such that

$$\|\mathcal{T}_h^\pi Q_1 - \mathcal{T}_h^\pi Q_2\|_\infty \leq \gamma\|Q_1 - Q_2\|_\infty. \tag{2.43}$$

Thus if $Q_{k+1} := \mathcal{T}_h^\pi Q_k$, the series $\{Q_k\}$ converges to Q_h^π, i.e. $\|Q_k - Q_h^\pi\|_\infty \to 0$ as $k \to \infty$. □

Soft policy improvement involves the information projection and update towards the exponential of the new Q-value,

$$\pi_{k+1} = \underset{\pi}{\operatorname{argmin}}\, d_{kl}\left(\pi(\cdot \mid s_t) \,\middle|\middle|\, \frac{\exp(\frac{1}{\alpha}Q^{\pi_k}(s_t, \pi(\cdot \mid s_t)))}{W^{\pi_k}(s_t)}\right), \tag{2.44}$$

where d_{kl} is the Kullback-Leibler divergence function and $W^{\pi_k}(s_t)$ denotes the partition function normalizing the distribution. $W^{\pi_k}(s_t)$ can be ignored since it does not contribute to the new policy.

2.2.3 Contraction and Improvement Properties

Convergence towards the maximum of (2.40) can be shown in soft policy iteration, which consists of soft policy evaluation under $\mathcal{T}_h^\pi$ and soft policy improvement.

2.2.3.1 Soft Policy Evaluation

Lemma 2.6. Let $Q_1(s,a)$ and $Q_2(s,a)$ be two state-action value functions. If $|\mathcal{A}| < \infty$, then the soft Bellman operator $\mathcal{T}_h^\pi$ according to (2.41) is a γ-contraction

$$||\mathcal{T}_h^\pi Q_1 - \mathcal{T}_h^\pi Q_2||_\infty \leq \gamma ||Q_1 - Q_2||_\infty. \tag{2.45}$$

Proof. Defining $r_h(s_t, a_t) := r(s_t, a_t) + \mathbb{E}_{s_{t+1}\sim P}[\mathcal{H}(\pi(\cdot \mid s_{t+1}))]$ and noting that $\mathcal{T}_h^\pi Q(s_t, a_t) = r_h(s_t, a_t) + \gamma \mathbb{E}_{s_{s+t}\sim P, a_{t+1}\sim\pi}[Q^\pi(s_{t+1}, a_{t+1})]$, we can apply the proof of Lemma 2.2 such that

$$||\mathcal{T}_h^\pi Q_1 - \mathcal{T}_h^\pi Q_2||_\infty \leq \gamma ||Q_1 - Q_2||_\infty. \tag{2.46}$$

Thus if $Q_{k+1} := \mathcal{T}_h^\pi Q_k$, the series $\{Q_k\}$ converges to Q_h^π, i.e. $||Q_k - Q_h^\pi||_\infty \to 0$ as $k \to \infty$. □

2.2.3.2 Soft Policy Improvement

It must be shown that (2.44) improves the policy.

Lemma 2.7. If the policy is updated according to (2.44), then

$$\mathbb{E}_{a_t\sim\pi_{k+1}}[Q^{\pi_k}(s_t, a_t) - \alpha \log \pi_{k+1}(a_t \mid s_t)] \geq \mathbb{E}_{a_t\sim\pi_k}[Q^{\pi_k}(s_t, a_t) - \alpha \log \pi_k(a_t \mid s_t)].$$

Proof.

$$\begin{aligned}
\pi_{k+1} &= \underset{\pi\in\Pi}{\operatorname{argmin}}\, D_{KL}\left(\pi(\cdot\,|\,s_t) \;\middle\|\; \frac{\exp\left(\frac{1}{\alpha}Q^{\pi_k}(s_t,\cdot)\right)}{W^{\pi_k}(s_t)}\right) \\
&= \underset{\pi\in\Pi}{\operatorname{argmin}}\, \mathbb{E}_{a_t\sim\pi}\left[\log\left(\frac{\pi(a_t\,|\,s_t)W^{\pi_k}(s_t)}{\exp\frac{1}{\alpha}Q^{\pi_k}(s_t,a_t)}\right)\right] \\
&= \underset{\pi\in\Pi}{\operatorname{argmin}}\, \mathbb{E}_{a_t\sim\pi}\left[\log\pi(a_t\,|\,s_t) - \frac{1}{\alpha}Q^{\pi_k}(s_t,a_t) + \log W^{\pi_k}(s_t)\right].
\end{aligned}$$

Due to minimization, we have $\mathbb{E}_{a_t\sim\pi_{k+1}}\left[\log\pi_{k+1}(a_t\,|\,s_t) - \frac{1}{\alpha}Q^{\pi_k}(s_t,a_t) + \log W^{\pi_k}(s_t)\right] \leq \mathbb{E}_{a_t\sim\pi_k}\left[\log\pi_k(a_t\,|\,s_t) - \frac{1}{\alpha}Q^{\pi_k}(s_t,a_t) + \log W^{\pi_k}(s_t)\right]$ and since the partition function does not depend on the action, we can write

$$\mathbb{E}_{a_t\sim\pi_{k+1}}[Q^{\pi_k}(s_t,a_t) - \alpha\log\pi_{k+1}(a_t\,|\,s_t)] \geq \mathbb{E}_{a_t\sim\pi_k}[Q^{\pi_k}(s_t,a_t) - \alpha\log\pi_k(a_t\,|\,s_t)].$$

□

The soft Q-value is expanded to show that π_{k+1} is indeed an improvement on π_k

Lemma 2.8. Let $Q_h^\pi := r(s_t,a_t) + \mathbb{E}_{s_{t+1}\sim P, a_{t+1}\sim\pi}[Q_h^\pi(s_{t+1},a_{t+1}) - \log\pi(a_{t+1}\,|\,s_{t+1})]$ and assume $|\mathcal{A}| < \infty$. If π_{k+1} is obtained according to the soft policy improvement in (2.44), then

$$Q_h^{\pi_k}(s_t,a_t) \leq Q_h^{\pi_{k+1}}(s_t,a_t) \quad \forall s\in\mathcal{S}.$$

Proof. Analogous to Lemma 2.3, the proof relies on expanding the soft Q-function in the Bellman equation repeatedly with actions of the improved policy. We have shown in the previous lemma an improvement for one step of the policy information equation. For convenience, we again define $\pi' := \pi_{k+1}$, $\pi := \pi_k$, let $\alpha = 1$ and abbreviate the subscripts of $\mathbb{E}$ for long trajectories. For better readability, we use the notation

$\mathbb{E}_{P,\pi'} = \mathbb{E}_{s_{t+1}\sim P, a_{t+1}\sim\pi'}$.

$$
\begin{aligned}
&Q_h^\pi(s_t, \pi(s_t)) \\
&\leq Q_h^\pi(s_t, \pi'(s_t)) \\
&= \mathbb{E}_{P,\pi'}[r(s_t, a_t) + \gamma(Q_h^\pi(s_{t+1}, \pi(s_{t+1})) - \log\pi(a_{t+1} \mid s_{t+1}))] \\
&\leq \mathbb{E}_{P,\pi'}[r(s_t, a_t) + \gamma(Q_h^\pi(s_{t+1}, \pi'(s_{t+1})) - \log\pi(a_{t+1} \mid s_{t+1})) \mid s_t = s] \\
&= \mathbb{E}_{P,\pi'}[r(s_t, a_t) + \gamma\mathbb{E}_P[r(s_{t+1}, a_{t+1}) - \log\pi(a_{t+1} \mid s_{t+1}) + \\
&\qquad\qquad \gamma^2(Q_h^\pi(s_{t+2}, \pi(s_{t+2})) - \log\pi(a_{t+2} \mid s_{t+1})) \mid s_{t+1}, a_{t+1} \sim \pi'(s_{t+1})] \mid s_t = s] \\
&= \mathbb{E}_{P,\pi'}[r(s_t, a_t) + \gamma(r(s_{t+1}, a_{t+1}) - \log\pi(a_{t+1} \mid s_{t+1})) + \\
&\qquad\qquad \gamma^2(Q^\pi(s_{t+2}, \pi(s_{t+2})) - \log\pi(a_{t+2} \mid s_{t+2})) \mid s_t = s] \\
&\vdots \\
&\leq \mathbb{E}_{P,\pi'}[r(s_t, a_t) + \gamma(r(s_{t+1}, a_{t+1}) - \log\pi(a_{t+1} \mid s_{t+1})) + \\
&\qquad\qquad \gamma^2(r(s_{t+2}, a_{t+2})\log\pi(a_{t+2} \mid s_{t+2})) + \ldots \mid s_t = s] \\
&= Q_h^{\pi'}(s_t, a_t).
\end{aligned}
$$

Thus $Q_h^\pi(s_t, a_t) \leq Q_h^{\pi'}(s_t, a_t)$. □

2.2.4 Implementation

As with other modern reinforcement learning approaches, the actor and critic are approximated by neural networks with parameters ϕ and θ, respectively. The update is based on

$$J_Q(\theta) := \mathbb{E}_{(s_t,a_t)\sim\rho_\mu}\left[\frac{1}{2}(\mathcal{T}_h^\pi Q(s_t, a_t) - Q_\theta(s_t, a_t))^2\right]. \tag{2.47}$$

The policy objective as a function of the actor parameters is

$$J_\pi(\phi) = \mathbb{E}_{s_t\sim\varsigma_\mu, \epsilon_t\sim\mathcal{N}(0,1)}[\log\pi_\phi(f_\phi(s_t; \epsilon_t) \mid s_t) - Q_\theta(s_t, f_\phi(s_t; \epsilon_t))], \tag{2.48}$$

where $a_t = f_{\phi(s_t;\epsilon_t)}$ is the action obtained by a reparameterized policy f_ϕ, an approach for calculating low variance gradients similar to [69]. As per usual notation, ς_μ denotes the state distribution induced by the behaviour policy μ and ϵ_t is a noise sampled from the standard Gaussian $\mathcal{N}(0, 1)$. In practice, the gradients derived from (2.47) and (2.48) are calculated from experience tuples sampled from the replay buffer. They are denoted by $\hat{\nabla}_\theta J_Q(\theta)$ and $\hat{\nabla}_\phi J_\pi(\phi)$.

2.2.4.1 Soft Policy Iteration

Theorem 2.9. *Under the assumptions that the reward is bounded, $|\mathcal{A}| < \infty$ and $|\mathcal{S}| < \infty$, repeatedly alternating between soft policy evaluation and soft policy improvement*

from any arbitrary policy $\pi \in \Pi$*, converges to an optimal policy* π^* *according to* (2.40) *and optimal state-action value function* Q_h^*.

Proof. A similar reasoning to policy iteration can be applied for soft policy iteration, see Lemma 2.4. □

2.2.5 Algorithm

Algorithm 2.5 presents a SAC variant with target networks. The original authors proposed a variant where the state value $V_h^\pi(s_t) := \mathbb{E}_{a_t \sim \pi_{k+1}}[Q^{\pi_k}(s_t, a_t) - \alpha \log \pi_{k+1}(a_t \mid s_t)]$ is approximated in addition to the state-action value. In the algorithm described below, the state value approximator is omitted.

Algorithm 2.5 SAC Algorithm

Initialize θ, $\bar{\theta}$, ϕ, $\bar{\phi}$
Initialize learning parameters β_Q, β_π
$\mathcal{B} \leftarrow \emptyset$
repeat
    Receive initial observation state s_1
    for each environment step **do**
        $a_t \sim \pi_\phi(a_t \mid s_t)$
        Observe transition $s_{t+1} \sim P(s_t, a_t)$
        Observe reward r_t
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
    for each gradient step do **do**
        Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from $\mathcal{B}$
        $\theta \leftarrow (1 - \beta_Q)\theta + \beta_Q \hat{\nabla}_\theta J_Q$ derived from (2.47)
        $\phi \leftarrow (1 - \beta_\pi)\phi + \beta_\pi \hat{\nabla}_\phi J_\pi$ derived from (2.48)
        Update target networks: $\bar{\theta} \leftarrow \tau\theta + (1 - \tau)\bar{\theta}$; $\bar{\phi} \leftarrow \tau\phi + (1 - \tau)\bar{\phi}$
until Stopping criterion
return $Q_{\bar{\theta}}, \pi_{\bar{\phi}}$

2.3 Distributional Reinforcement Learning

Distributional reinforcement learning (DistRL) shifts the focus from modelling expected returns, as in traditional value-based methods, to capturing the entire distribution of

returns. Therefore, the reward is treated as a random variable R as well as the expected state-action value function, e.g. in the actor-critic setting in (2.18), it becomes

$$Z^{\pi}(s,a) := \sum_{k=t}^{T} \gamma^{k-t} R(s_t, a_t) \,|\, s_t \sim P, a_t \sim \pi, a_0 = a, s_0 = s, \tag{2.49}$$

and

$$Q^{\pi}(s_t, a_t) = \mathbb{E}[Z^{\pi}(s_t, a_t)].$$

Empirical data indicate that DistRL algorithms often outperform their non-distributional counterparts. While the precise reasons for their accelerated learning remain unclear, several hypotheses and theories have been presented [20, 70, 71].

2.3.1 Modelling Distributions and Statistical Metrics

In practice, assumptions about the distribution underlying the random variable Z matters. The modelling choice might affect gradient properties and thus effect convergence.

In distributional reinforcement learning, the choice of statistical metric to quantify the distance between the approximated value distribution and its target is critical. An inappropriate choice can hinder convergence to a (local) minimum. Bellemare et al. (2017) [20] formalized and analysed the distributional perspective in reinforcement learning for the first time in context of the p-Wasserstein metric:

$$d_W(U,V) := \left(\int_0^1 |F_U^{-1}(x) - F_V^{-1}(x)|^p dx \right)^{\frac{1}{p}}, \tag{2.50}$$

where U, V are random variables and F_U^{-1} is the inverse cumulative distribution function (CDF) of the random variable U. The algorithm presented in [20] incorporates a projection step in order to utilise the Wasserstein metric. As highlighted in [72], some distributions, when used in conjunction with certain statistical metrics for $TD-$updates, inherently produce biased gradients. Quantile regression networks and implicit quantile regression networks were introduced in [73, 74] as a solution for the bias problem. They yield unbiased sample gradients calculated from a loss related to the Wasserstein metric.

The following sections provide the theoretical framework for DistRL. Analysis is performed with the Wasserstein metric.

2.3.2 Evaluation Setting

The distributional counterpart of the Bellman operator in (2.6) is used for distributional evaluation. It is defined as

$$\mathcal{T}_D^\pi Z(s_t, a_t) := R(s_t, a_t) + \gamma Z(s_{t+1}, a_{t+1}) \,|\, s_{t+1} \sim P, a_{t+1} \sim \pi. \tag{2.51}$$

To proof that $\mathcal{T}_D^\pi$ is a contraction mapping, the following properties of the p-Wasserstein metric are required,

$$d_W(aU, aV) \leq |a| d_w(U, V), \tag{W1}$$

$$d_W(A + U, A + V) \leq d_w(U, V), \tag{W2}$$

where a is an arbitrary constant and A is a random variable.

Lemma 2.10. Let $\bar{d}_W(Z_1, Z_2) := \sup_{s,a} d_W(Z_1(s,a), Z_2(s,a))$ for two state-action value distributions Z_1 and Z_2. Then $\mathcal{T}_D^\pi$ is a γ-contraction in the regularized Wasserstein metric, i.e.

$$\bar{d}_W(\mathcal{T}_D^\pi Z_1, \mathcal{T}_D^\pi Z_2) \leq \gamma \bar{d}_W(Z_1, Z_2). \tag{2.52}$$

Proof. To show that (2.52) holds, we expand the left-hand side. Be reminded that $s_{t+1} \sim P(s_t, a_t)$. Then $\mathcal{T}_D^\pi Z(s_t, a_t) = R(s_t, a_t) + \gamma Z^\pi(s_{t+1}, a_{t+1}) \,|\, s_{t+1} \sim P(s_t, a_t), a_{t+1} \sim \pi(\cdot \,|\, s_t)$. It follows,

$$\begin{aligned}
&\bar{d}_W(\mathcal{T}_D^\pi Z_1(s_t, a_t), \mathcal{T}_D^\pi Z_2(s_t, a_t)) \,|\, s_{t+1} \sim P, a_{t+1} \sim \pi \\
&\qquad = \sup_{s_t, a_t} d_W\left(R(s_t, a_t) + \gamma Z_1(s_{t+1}, a_{t+1}), R(s_t, a_t) + \gamma Z_2(s_{t+1}, a_{t+1})\right) \\
&\text{By property (W2)} \\
&\qquad \leq \sup_{s_t, a_t} d_W\left(\gamma Z_1(s_{t+1}, a_{t+1}), \gamma Z_2(s_{t+1}, a_{t+1})\right) \,|\, s_{t+1} \sim P, a_{t+1} \sim \pi \\
&\text{By property (W1)} \\
&\qquad \leq \gamma \sup_{s_t, a_t} d_W(Z_1(s_{t+1}, a_{t+1}), Z_2(s_{t+1}, a_{t+1})) \,|\, s_{t+1} \sim P, a_{t+1} \sim \pi \\
&\text{Taking the supremum over the state and action independent of } s_t, a_t \\
&\qquad \leq \gamma \sup_{s \in \mathcal{S}, a \in \mathcal{A}} d_W(Z_1(s,a), Z_2(s,a)) = \gamma \bar{d}_c(Z_1, Z_2).
\end{aligned}$$

Since $\mathcal{T}_D^\pi$ is a γ-contraction, it follows from the Banach theorem that the series $\{Z_k\}$ with $Z_{k+1} := \mathcal{T}_D^\pi Z_k$ converges to a unique fixed point Z^π, i.e. $Z = \mathcal{T}_D^\pi Z$ and $||Z_k - Z^\pi||_\infty \to 0$ as $k \to \infty$. □

Contraction mapping can also be shown in the expectation.

Lemma 2.11. Let Z_1 and Z_2 be random variables, then

$$\|\mathbb{E}\mathcal{T}_D^\pi Z_1 - \mathbb{E}\mathcal{T}_D^\pi Z_2\|_\infty = \gamma \|\mathbb{E}Z_1 - \mathbb{E}Z_2\|_\infty \tag{2.53}$$

Proof. The proof relies on the linearity of expectation and (2.51) leveraging the fact that $\mathbb{E}[Z] = Q$. It follows

$$\|\mathbb{E}\mathcal{T}_D^{\pi} Z_1 - \mathbb{E}\mathcal{T}_D^{\pi} Z_2\|_{\infty} = \|\mathcal{T}^{\pi}\mathbb{E}Z_1 - \mathcal{T}^{\pi}\mathbb{E}Z_2\|_{\infty} \tag{2.54}$$

From Lemma (2.2),

$$\|\mathcal{T}^{\pi}\mathbb{E}Z_1 - \mathcal{T}^{\pi}\mathbb{E}Z_2\|_{\infty} \leq \gamma\|\mathbb{E}Z_1 - \mathbb{E}Z_2\|_{\infty}$$

and thus

$$\|\mathbb{E}\mathcal{T}_D^{\pi} Z_1 - \mathbb{E}\mathcal{T}_D^{\pi} Z_2\|_{\infty} \leq \gamma\|\mathbb{E}Z_1 - \mathbb{E}Z_2\|_{\infty}$$

□

2.3.3 Control Setting

Recall that in the control setting, the goal is to find a policy that maximizes the value function. This principle is central to reinforcement learning and has been formalized in the classical context: for the critic-only framework, this was detailed in Equation (2.3), while for the actor-critic (AC) framework, it was introduced in Equation (2.22). In critic-only settings, a distributional variant of the Bellman optimality operator can be defined as

$$\mathcal{T}_D Z(s_t, a_t) := R(s_t, a_t) + \gamma\mathbb{E}_{s_{t+1}\sim P}\left[\max_a Z(s_{t+1}, a)\right]. \tag{2.55}$$

However, as shown in [20], this operator lacks a contraction property in the general sense. Consequently, if a unique fixed point Z^* exists, convergence to Z^* might only occur if there is a unique policy associated with Z^*. These challenges cast the distributional Bellman optimality operator in a less favorable light for practical applications. However, this limitation is circumvented in the present work. In the novel AC- algorithm, the policy is improved in a standard manner with the expectation, as detailed in Section 3.2.

2.3.4 Moment-Agnostic Distributions

Free-moment distributions refer to probability distributions that are not constrained in their moments, meaning that they can adopt arbitrary moments. The following sections introduce distributional free-moment algorithms in the critic-only framework (Section 2.1.2.1).

2.3.4.1 Fixed Supports - C51

The C51 algorithm, introduced in [20], models the return distribution using a discrete representation with N atoms. These fixed supports are defined by a discretization of the interval $[V_{min}, V_{max}]$ as $\{z_i = V_{min} + i\Delta z : 0 \leq i < N\}$, where $V_{min}, V_{max} \in \mathbb{R}$ are assumed value limits and $\Delta z := \frac{V_{max} - V_{min}}{N-1}$ represents the spacing between the atoms. The return distribution is characterized by the probabilities associated with these atoms:

$$p_i(s_t, a_t) := \frac{e^{Z_{\theta_i}(s_t, a_t)}}{\sum_j e^{Z_{\theta_j}(s_t, a_t)}},$$

where $Z_{\theta_i}(s_t, a_t)$ approximates the probability of z_i and is part of the parameterized network $Z_\theta(s_t, a_t)$. Although theoretically attractive, challenges arise in directly minimizing the Wasserstein distance between the estimated distribution Z and the target $\mathcal{T}_D$ in a mini-batch. Lemma 7 in [20] shows that stochastic gradient descent generally cannot achieve this. In addition, the support of $\mathcal{T}_D Z$ generally do not coincide with that of Z. To address this, C51 projects the distribution of $\mathcal{T}_D Z$ onto Z through a projection Φ. This effectively reduces the problem to a multi-class classification. The projection step is defined by

$$m_i = (\Phi \hat{\mathcal{T}}_D Z(s_t, a_t))_i := \sum_{j=0}^{N-1} \left[1 - \frac{|[\hat{\mathcal{T}} z_j]_{V_{min}}^{V_{max}} - z_i|}{\Delta z} \right] p_j(s_{t+1}, a_{t+1})$$

and the loss becomes

$$J_C(\theta) = \mathbb{E}_{(s_t, a_t) \sim \rho_\mu} \left[-\sum_i m_i \log p_i(s_t, a_t) \right]. \tag{2.56}$$

The expression $[\hat{\mathcal{T}} z_j]_{V_{min}}^{V_{max}}$ denotes the actual target bounded by the limits. The projected categorical loss is related to minimizing the Cramér metric [75].

Another limitation of C51 lies in the reliance on domain knowledge to set the parameters V_{min} and V_{max}, which define the limits of the supports. This practical challenge highlights a gap between theory and implementation, later addressed by quantile regression networks that can operate under the Wasserstein metric and eliminate the need for such predefined bounds. Algorithm 2.6 describes the C51 procedure.

2.3.4.2 Quantile Regression

Quantile Regression Networks (QR-DQN) is an extension of DistRL and offers a novel approach to represent the distribution over returns using a set of quantiles. This approach enables the direct computation of sample gradients over the Wasserstein metric,

Algorithm 2.6 C51 Algorithm

Initialize θ
Initialize learning parameter β_z
Initialize exploration $\epsilon = 1$ and decay parameter β_ϵ
$\mathcal{B} \leftarrow \emptyset$
repeat
    Receive initial observation state s_1
    for each environment step **do**
        With probability ϵ select random action $a_t \sim \mathcal{U}(\{1, \dots |\mathcal{A}|\})$
        With probability $1 - \epsilon$ select action $a_t = \text{argmax}_a \sum_i z_i p_i(s_t, a)$
        Observe transition $s_{t+1} \sim P(s_t, a_t)$
        Observe reward r_t
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
        $s_t \leftarrow s_{t+1}$
    for each gradient step do **do**
        Sample a random minibatch of N transitions (s_i, a_i, r_i, s_{i+1}) from $\mathcal{B}$
        $\theta \leftarrow (1 - \beta_Q)\theta + \beta_Q \hat{\nabla}_\theta J_C$ of (2.56) from mini-batch gradient $\hat{\nabla}_\theta$
        $\epsilon \leftarrow \max(\epsilon - \beta_\epsilon, 0)$
until Stopping criterion
return $Q_{\bar{\theta}}, \pi_{\bar{\phi}}$

bridging the gap between theory and practical application described in [20]. The key feature is to "transpose" the parameterization of C51: Fixed-support parameterizations are abondened in favor of "uniform probabilities to N adjustable locations "[73]. QR-DQN models return distributions using quantiles corresponding to specific probabilities within the cumulative distribution function (CDF):

$$F_Z(z_\tau) = P(Z \leq z_\tau)$$

with the quantile values given by

$$z_\tau = F_X^{-1}(\tau).$$

The discrete values of the CDF are partitioned into N quantiles, where each quantile fraction is defined as $\tau_i = \frac{1}{N}$ for $i = 1, \dots, N$. The learning process involves adjusting the predicted quantiles to minimize the projected Wasserstein-1 distance to the target

distribution.

$$W_1(Z_1, Z_\theta) := \sum_{i=1}^{N} \int_{\tau_{i-1}}^{\tau_i} |F_{Z_1}^{-1}(\omega) - F_{Z_{\theta_i}}^{-1}| d\omega, \tag{2.57}$$

with the objective being

$$\Pi_{W_1} Z := \operatorname*{argmin}_{Z_\theta} W_1(Z, Z_\theta). \tag{2.58}$$

Dabney et al. (2017) demonstrated that the optimal quantile values minimizing over the interval $[\tau, \tau']$ are those that yield $\hat{\tau} = \frac{\tau+\tau'}{2}$, i.e. $\{Z_{\theta_i} \in R \,|\, F(Z_{\theta_i}) = \frac{\tau+\tau'}{2}\}$. Adjusting the quantiles poses a challenge. If values associated with lower quantiles are significantly overestimated, the confidence in the error is higher than when these values are overestimated at upper quantiles. Additionally, since CDFs are non-decreasing, the error is accumulated in the upper quantiles. On the other hand, underestimation of values on upper quantiles means an underestimation error persistently occurring in lower quantiles also. To address this issue, QR-DQN employs the quantile regression loss, which penalizes discrepancies asymmetrically:

$$\begin{aligned} J_\tau^{QR}(\theta) := \sum_{i}^{N} \mathbb{E}[\rho_{\hat{\tau}_i}(\mathcal{T}_D Z(s_t, a_t) - Z_{\theta_i}(s_t, a_t))] \quad \text{where} \\ \rho_{\hat{\tau}_i}(u) = u(\hat{\tau}_i - \mathbb{I}(u < 0)), \forall u \in \mathbb{R} \end{aligned} \tag{2.59}$$

with $\mathbb{I}$ being the indicator function. This loss gives unbiased sample gradients, i.e. yields the minimizing parameters $\{Z_{\theta_1}, \ldots, Z_{\theta_N}\}$, and is related to the Wasserstein-1 metric through $\hat{\tau}_i$. Under the Wasserstein metric d_{w_1}, [73] proved that the projected distributional Bellman operator is a contraction,

$$\bar{d}_w(\Pi_{W1} \mathcal{T}_D Z_1, \Pi_{W1} \mathcal{T}_D Z_2) \leq \gamma \bar{d}_w(Z_1, Z_2). \tag{2.60}$$

While the quantile regression loss is effective, its non-smoothness at $u \mapsto 0^+$ can cause numerical instability. To address this issue, QR-DQN employs the Huber [76], enforcing a quadratic loss in an interval $[-\kappa, \kappa]$ around 0. It is defined as

$$H^\kappa(u) := \begin{cases} \frac{1}{2}u^2 & \text{if } |u| < \kappa \\ \kappa(|u| - \frac{1}{2}\kappa) & \text{otherwise} \end{cases} \tag{2.61}$$

Then the quantile Huber loss is

$$\begin{aligned} J_\tau^{QHR}(\theta) := \sum_{i}^{N} \mathbb{E}[\rho_{\hat{\tau}_i}^\kappa(\mathcal{T}_D Z(s_t, a_t) - Z_{\theta_i}(s_t, a_t))] \quad \text{where} \\ \rho_{\hat{\tau}}^\kappa(u) := |\hat{\tau}_i - \mathbb{I}(u < 0)| H(u; \kappa) \end{aligned} \tag{2.62}$$

In practice, the target distribution $\hat{Z}(s_t, a_t) := \mathcal{T}_D Z(s_t, a_t)$ is calculated parameter-wise as

$$\mathcal{T}_D Z_j(s_t, a_t) = R(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim P}[Z_j(s_{t+1}, a_{t+1}) \,|\, a_{t+1} = \underset{a}{\operatorname{argmax}}\, \mathbb{E}[Z(s_{t+1}, a)]], \quad \forall j.$$

This formulation aligns with the quantile regression loss calculation outlined in Algorithm 2.7. The remaining components of the algorithm are consistent with those in Algorithm 2.1 and Algorithm 2.6. QR-DQN introduces several notable advancements over C51. Firstly, it replaces the fixed quantile supports of C51 with dynamically adjustable quantile locations, enhancing robustness to probability mismatches. Secondly, QR-DQN is grounded in a rigorous theoretical framework, as it explicitly minimizes the Wasserstein-1 metric. Nevertheless, its reliance on discretization may introduce approximation errors.

Algorithm 2.7 Quantile Regression Q-Learning

Require: $N, \kappa,\ s_t, a_t, r_t, s_{t+1}, \gamma \in [0, 1]$

1: **# Compute distributional Bellman target**
2: $Q(s_{t+1}, a_{t+1}) := \sum_j \frac{1}{N} Z_j(s_{t+1}, a_{t+1})$
3: $a^* \leftarrow \arg\max_{a_{t+1}} Q(s_{t+1}, a_{t+1})$
4: $\mathcal{T} Z_j \leftarrow r_t + \gamma Z_j(s_{t+1}, a^*), \quad \forall j$
5: **# Compute quantile regression loss according to Equation** (2.62)
6: **return** $\sum_{i=1}^{N} \mathbb{E}_j \left[\rho_{\tau_i}^{\kappa}(\mathcal{T} Z_j - Z_{\theta_i}(s_t, a_t))\right]$

2.3.4.3 Implicit Quantile Networks

Implicit Quantile Networks (IQN) extend the Distributional Reinforcement Learning paradigm by improving upon the limitations of QR-DQN, primarily through a more flexible and expressive representation of the value distribution. Unlike QR-DQN, which utilizes a fixed set of quantile fractions, IQN introduces quantile regression to approximate the quantile function, $F_Z^{-1}(\tau)$, as proposed by [74]. This enables a more nuanced modelling of the return distribution. The goal is to model the random variable $Z_\tau := F_Z^{-1}(\tau)$, where τ is uniformly sampled, i.e. $\tau \sim \mathcal{U}([0, 1])$. This approach allows the return distribution to be implicitly defined by the quantile function $Z_\tau := F_Z^{-1}(\tau)$, which is learned via a neural network approximator. This representation captures the variability in returns across quantiles without explicitly parameterizing the distribution. In the control setting, IQN incorporates risk-sensitive learning by considering how a utility function U influences decision-making in the presence of risk. A concave

U reflects risk-averse behavior, a convex U corresponds to risk-seeking behaviour, and the identity function reflects risk neutrality. By sampling τ non-uniformly, the policy can emphasize different parts of the return distribution. For example, sampling τ near 1 focuses on upper quantiles, reflecting optimism and risk-seeking behaviour. This non-uniform sampling can be thought of as τ being re-weighted by some distortion risk measure $\xi : [0, 1] \mapsto [0, 1]$, which transforms the baseline uniform sampling. The distorted expectation of the value distribution is given by:

$$Q_\xi(s_t, a_t) := \mathbb{E}_{\tau \sim U([0,1])}[Z_{\xi(\tau)}(s_t, a_t)]. \tag{2.63}$$

The corresponding policy under ξ is derived through

$$\pi_\xi(s_t) = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\, Q_\xi(s_t, a). \tag{2.64}$$

Policy evaluation is conducted with Z_τ and $T_D Z_{\tau'}$, i.e. based on uniformly sampled τ sampling. The evaluation loss is analogous to the QR-DQN loss in (2.62):

$$J_\tau^{IQN}(s_t, a_t) := \sum_{i=1}^{N} \mathbb{E}_j[\rho_{\tau_i}^\kappa(\mathcal{T}_D Z_{\tau'}(s_t, a_t) - Z_\tau(s_t, a_t))], \tag{2.65}$$

where $\mathcal{T}_D Z_{\tau'} = R(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1}} \max_{a \in \mathcal{A}} Z_{\tau'}(s_{t+1}, \pi_\xi(s_{t+1}))$. Note that the distortion is applied to when taking an action, but the distributions for the TD-update are unaffected. IQN employs a novel mechanism for approximating the quantile function. The scalar τ is transformed into a high-dimensional embedding vector, which interacts with the state observation through an element-wise Hadamard product. This product is subsequently processed by fully connected layers to estimate the quantile value. More details on this step will be omitted since this work does not delve deeper into the implicit quantile approach.

2.3.5 Fixed-Moment Distributions

Fixed-moment distributions represent a specific class of approximations within the broader field of distributional reinforcement learning. These distributions are fully characterized by their first two moments: the mean and variance. By constraining the return distribution to this simplified representation, fixed-moment distributional reinforcement learning algorithms operate under strong assumptions about the underlying true distributions of returns. The fixed-moment approach offers computational simplicity, but it comes at the potential cost of accuracy. The rigid assumptions can lead to biased approximations of the true return distribution. However, this error introduces an intriguing tradeoff between variance and bias in learning. By reducing

the complexity of the distribution - requiring fewer parameters for approximation - fixed-moment algorithms can yield lower gradient variance during optimization. The reduction in variance can enhance learning stability and efficiency, making these algorithms appealing in scenarios where computational resources are limited, as was the case for executing experiments for this work. High variance might also impede effective policy updates.

2.3.5.1 Parameterized Distributions - Gauss

One of the simplest approaches to parameterize return distributions in reinforcement learning is to model them as Gaussian distributions. The Gaussian probability density function is defined as:

$$\varphi_{\mu,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{(x-\mu)}{\sigma}\right)^2}, \tag{2.66}$$

where μ represents the mean, and σ denotes the standard deviation. In the context of reinforcement learning, since $Q = \mathbb{E}[Z]$, the mean μ of the Gaussian can be directly replaced with Q. This makes the Gaussian distribution particularly straightforward to integrate into existing reinforcement learning frameworks. The Gaussian distribution has been utilized for modeling the policy, yielding strong empirical performance in algorithms such as PPO (Section 2.1.6.4) and SAC (Section 2.2.3). Modeling the return as a Gaussian is hypothesized to be both robust and equally capable of delivering high performance. This hypothesis is ultimately confirmed, see Sections 3.3 and 3.5. The use of Gaussian distributions for fixed-moment representations is advantageous due to their well-understood mathematical properties and computational efficiency. While the assumption of a symmetric and unimodal distribution may not always align with the true return distribution, the Gaussian provides a practical tradeoff between simplicity and performance.

2.3.5.2 Parameterized Distributions - Gauss Mixture Models

To allow for a more expressive representation, the return in distributional reinforcement learning can be modeled as a mixture distribution. One promising approach is to use Gaussian Mixture Models (GMMs). GMMs have been used for population modelling by Pearson as early as 1894 [77]. GMMs possess several useful mathematical properties, such as their ability to approximate arbitrary densities universally and to combine Gaussian kernels through convex combinations. In this work, GMMs have been investigated as models for the return in DistRL. The return distribution is expressed

as:

$$\varrho_{\mu,\sigma}(x) = \sum_{k=1}^{K} w_k \varphi_{\mu_k,\sigma_k}(x),$$

where w_k is the weight of the k-th kernel and φ_{μ_k,σ_k} denotes the Gaussian kernel with mean μ_k and variance σ_k^2. GMMs are particularly appealing for DistRL because they can inherently model multimodality, allowing the return distribution to capture diverse structural properties of the environment. This could facilitate deducing insights about the reward function or the dynamics of the environment. To assess the adequacy of GMMs in deep DistRL, experiments have been conducted simulating a $TD(0)$-step (Section 2.1.2.2) with a noisy target. The regularized Cramér metric (Equation (3.1)) was employed as a loss function, measuring the distance between the predicted and target distributions. Figure 2.2 demonstrates the results of fitting a simple two-kernel GMM neural network approximator with a single hidden layer containing 10 nodes. In this setup, the learnable parameters includes the kernel weights, means, and variances each represented by an output node. The simple approximator was able to fit the target distributions almost perfectly. Figures 2.2a and 2.2b illustrate the CDF and kernel mean fitting, respectively, for a target distribution with variance 10. Figures 2.2c and 2.2d show the corresponding results for a target with variance 1. Figures 2.2b and 2.2d reveal that while the kernel means converge to the target means, the fitting process is slow.

In contrast, as the complexity of the GMM neural network approximator increases, its performance in fitting the target deteriorates. Figure 2.3 shows that a more sophisticated approximator, with two hidden layers each containing 256 nodes, fails to generalize well to the target distributions with low and high variances. While the fitted distributions maintain the same expected values as the target, significant discrepancies are observed in the variances. This suggests that overparameterization in GMM neural network approximators may hinder their ability to model the target distribution effectively. In reinforcement learning, large, overparameterized models have been observed to achieve better generalization in diverse, procedurally generated settings, compared to simpler models [78, 79]. Further, from experiments conducted in Section 3.5, larger networks tend to yield returns higher then the returns obtained by simple approximators, within the same number of iterations.

These findings highlight the potential and limitations of GMMs for modeling the return in DistRL. Therefore, GMMs will not be employed in this work. Further research may focus on identifying configurations and regularization techniques that enable GMMs to balance expressiveness and stability in learning.

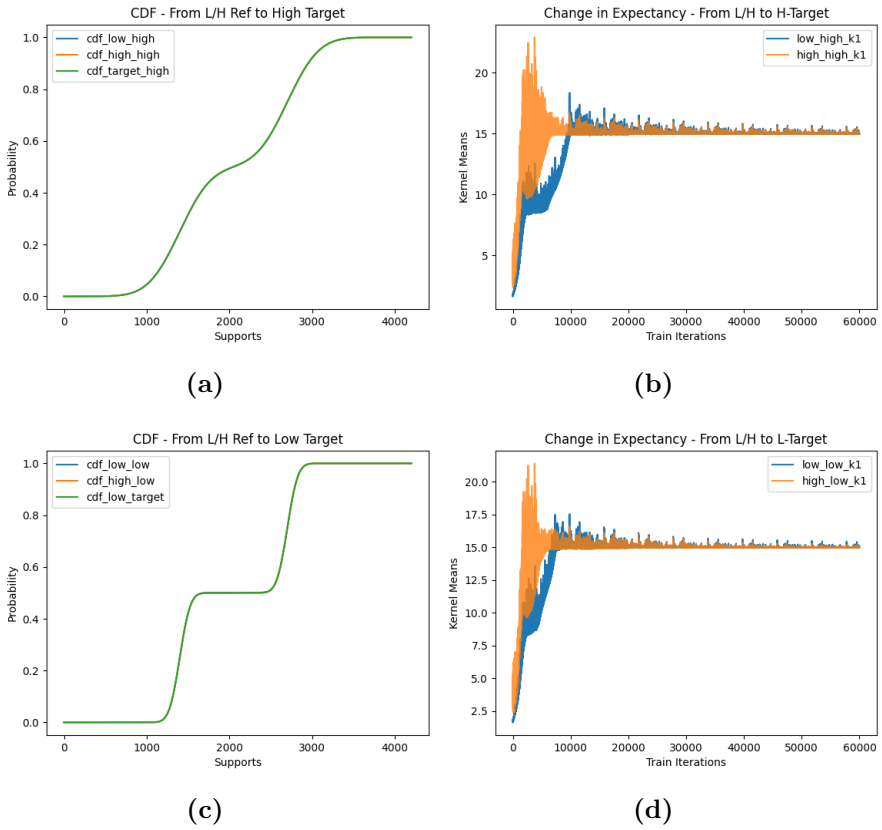


Figure 2.2: Plots (a) and (c) show fitting of a simple neural network GMM approximator with $K = 2$, consisting of one hidden layer with 10 nodes, to a high- and low-variance target from low (blue) and high (orange) reference distributions. All curves coincide, indicating perfecting fitting to the target. The neural network consists of one hidden layer containing 10 nodes. Figures (b) and (d) depict the change of the mean of one kernel per iteration in the neural network fitting process.

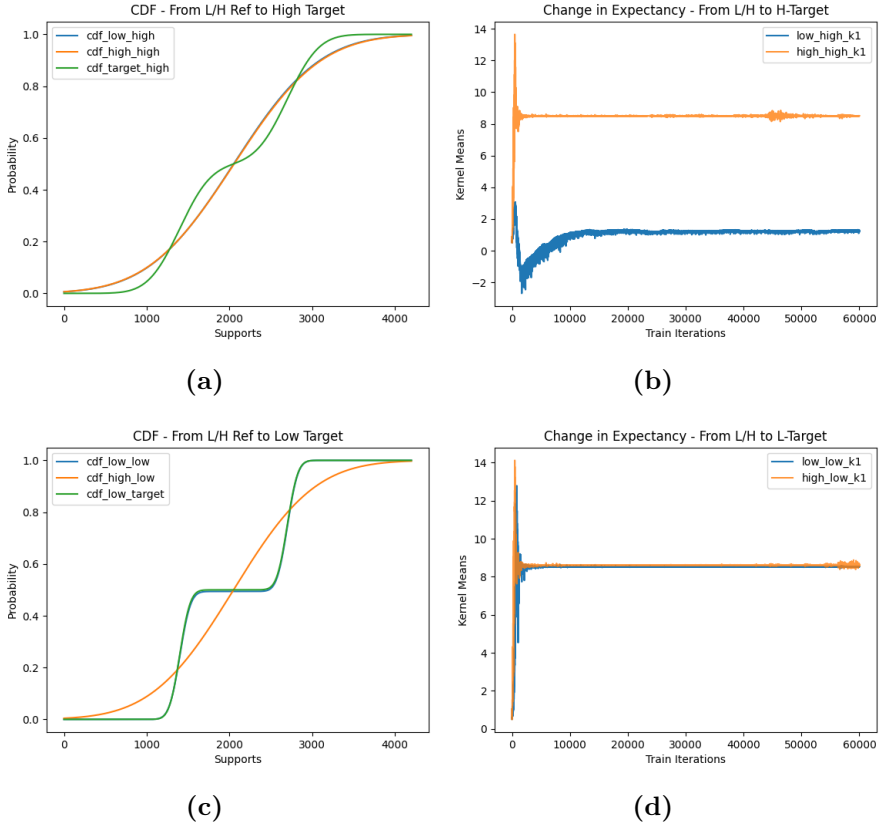


Figure 2.3: In this run, the neural network GMM approximator consists of two hidden layers, each containing 256 nodes. Plots (a) and (c) show fitting of a complex neural network GMM approximator with $K = 2$ to a high- and low-variance target from low (blue) and high (orange) reference distributions. In (a), the orange and blue curves conincide but deviate from the target CDF, indicating insufficient fitting to the target. However, in (c), only the low variance reference distribution (blue) is able to fit the target, while the high variance distribution (orange) fails to capture the target CDF. Figures (b) and (d) depict the change of the mean of one kernel per iteration in the neural network fitting process.

3

Distributional Reinforcement Learning in the Maximum-Entropy Setting Using the Cramér Metric

This chapter introduces a state-of-the-art novel Q-based (Section 2.1.6.1), distributional reinforcement learning (Section 2.3) algorithm with an entropy maximization objective (Section 2.2) in the actor-critic (Section 2.1.2.4) framework. The algorithm advances current state-of-the-art performance in reinforcement learning, and has a rigorous theoretical framework ensuring convergence and stable updates, even in presence of high environmental noise.

3.1 Related Work

Bellemare et al. [20] were the first to formalize and analyse the distributional approach to reinforcement learning, focusing on learning return distributions rather than expected returns. They employed the Wasserstein metric to measure distributional distances, proved convergence in the evaluation setting, and investigated a distributional policy improvement. It was shown that under mild assumptions, distributional policy evaluation and improvement converged to the optimal policy. This work culminated in the C51 algorithm, a critic-only method for discrete action spaces. However, as mentionend in Section 2.3, C51 employs a projection step not aligned with the theory of distributional evaluation and improvement. In [80] and [81], the concepts of SAC and DRL were applied simultaneously for the first time. The former method relies on the Kullback-Leibler divergence to measure the difference between two distributions. However, this conflicts with the properties required for distributional policy evaluation described in [20], as the distance becomes infinite if the distributions have dissimilar supports. To address this issue, the authors applied clipping, which negatively impacts

efficiency by preventing proper scaling. Moreover, the authors of this work were unable to reproduce the results using the provided software. The latter approach utilizes distributional reinforcement learning with quantile regression [73]. [81] reported superior performance compared to common reinforcement learning algorithms, achieved by adding a risk measure function to state-action values and using networks with more parameters than the baselines. Later, we will demonstrate that such measures are unnecessary for C-DSAC, as it inherently incorporates a 'risk-aversion' mechanism. The work of [82] focuses on the Cramér distance "in the setting of fixed quantile levels" and does not present an algorithm for continuous control.

3.2 The C-DSAC Framework

In the following sections, we explore the distributional approach to reinforcement learning and introduce a novel method for adaptively updating models of distributional Q-values using the regularized Cramér distance. Building on these foundations and incorporating maximum-entropy reinforcement learning,a Cramér-based Distributional Soft Actor-Critic (C-DSAC) framework is proposed. Moreover, corresponding formulas for implementation with neural networks are derived and experimental data demonstrating state-of-the-art performance across tested environments is presented. In order to explain the efficiency of the new approach, a thorough analysis of the algorithm's dynamics is conducted. The focus is on the algorithm's behaviour in presence of approximation errors and system noise and equations for its value gradients are derived showing that the adaptation to Q-values is slower at state action pairs where the variance of the distributional Q-model is large, i.e. confidence in its value is low. Furthermore, we emphasize how these equations reveal the algorithm's inherent mechanism for mitigating overestimation bias. The following theorems provide a theoretical basis for a Cramér-based distributional SAC approach.

3.2.1 The Regularized Cramér Distance

Choosing an appropriate metric to quantify the distance between random variables U and V is critical for policy evaluation and improvement in distributional reinforcement learning. [72] noted that a metric d must be both "ideal" according to [83] and suitable for gradient descent methods to be effective in machine learning with distributions.

Ideal metrics require to be sum invariant and scale sensitive. Sum invariance is the property for which the following holds:

$$d(A+U, A+V) \leq d(U, V), \tag{P1}$$

where A is a random variable independent of U and V. Scale sensitivity, the second required property, ensures that for $k > 0$ and some $p \geq 1$,

$$d(kU, kV) \leq |k|^{\frac{1}{p}} d(U, V). \tag{P2}$$

A third property is needed to guarantee unbiased sample gradients, expressed by

$$\mathbb{E}_{\mathbf{X_m} \sim \Gamma} \nabla_\theta d(\hat{\Gamma}_m, Q_\theta) = \nabla_\theta d(\Gamma, Q_\theta), \tag{P3}$$

where $\mathbf{X}_m$ is a vector with elements $X_1, \ldots, X_m$ drawn from a Bernoulli distribution Γ, and $\hat{\Gamma}_m := \frac{1}{m} \sum_{i=1}^{m} \delta_{X_i}$ is an approximate distribution formed by the samples with δ being Dirac functions at values X_i.

The regularized Cramér distance, also called energy distance, for two random variables U and V is defined by

$$d_c(U, V) := \int_{-\infty}^{\infty} (F_U(x) - F_V(x))^2 dx, \tag{3.1}$$

where F_U and F_V are the cumulative distribution functions of U and V respectively.

Lemma 3.1. The regularized Cramér distance d_c satisfies Properties (P1)-(P3) for $p = 2$.

Proof. See Theorem 2 of [72].

□

3.2.2 Objective

The aim is to find a policy $\pi^* = \text{argmax}_{\pi \in \Pi} J_H(\pi)$ that maximizes the entropy-augmented expectation over the distributional returns

$$J_H(\pi) := \sum_{t=0}^{T} \mathbb{E}_{(s_t, a_t) \sim \rho_\pi} \left[\sum_{\ell=t}^{T} \gamma^{\ell - t} \mathbb{E}_{s_\ell \sim P, a_\ell \sim \pi} [\mathbb{E}[R(s_\ell, a_\ell)] + \alpha \mathcal{H}(\cdot \mid s_\ell)] \right]. \tag{3.2}$$

In our approach, akin to distributional reinforcement learning, we explicitly model the reward as a random variable, thus utilizing a distributional value function. Similar to the SAC algorithm (Section 2.2.3), the negative log probability to measure entropy is chosen.

3.2.3 Policy Evaluation

We consider a distributional reinforcement learning setting (Section 2.3) under the regularized Cramér distance where we utilize the distribution over returns Z. Let

$$V_H(s_t) = \mathbb{E}_{a_t \sim \pi}[Z(s_t, a_t) - \alpha \log \pi(a_t \,|\, s_t)],$$

then the soft distributional Bellman operator $\mathcal{T}_H^\pi$ with $s_{t+1} \sim P(s_t, a_t)$ is defined as

$$\mathcal{T}_H^\pi Z(s_t, a_t) := R(s_t, a_t) + \gamma V_H(s_{t+1}) \,|\, s_{t+1} \sim P. \tag{3.3}$$

Theorem 3.2. *Let $\bar{d}_c(Z_1, Z_2) := \sup_{s,a} d_c(Z_1(s,a), Z_2(s,a))$ for two state-action value distributions Z_1 and Z_2. Then $\mathcal{T}_H^\pi$ is a γ-contraction in the regularized Cramér distance, i.e.*

$$\bar{d}_c(\mathcal{T}_H^\pi Z_1, \mathcal{T}_H^\pi Z_2) \leq \gamma \bar{d}_c(Z_1, Z_2). \tag{3.4}$$

Proof. Let $R_H(s_t, a_t) := R(s_t, a_t) + \alpha \mathcal{H}(\pi(\cdot \,|\, s_{t+1})) \,|\, s_{t+1} \sim P(s_t, a_t)$, then $\mathcal{T}_H^\pi Z(s, a_t) = R_H(s_t, a_t) + \gamma Z^\pi(s_{t+1}, a_{t+1}) \,|\, s_{t+1} \sim P(s_t, a_t), a_{t+1} \sim \pi(\cdot \,|\, s_t)$. It follows,

$$\begin{aligned}
&\bar{d}_c(\mathcal{T}_H^\pi Z_1(s_t, a_t), \mathcal{T}_H^\pi Z_2(s_t, a_t)) \,|\, s_{t+1} \sim P, a_{t+1} \sim \pi \\
&\quad = \sup_{s_t, a_t} d_c\left(R_H(s_t, a_t) + \gamma Z_1(s_{t+1}, a_{t+1}), R_H(s_t, a_t) + \gamma Z_2(s_{t+1}, a_{t+1})\right) \\
&\text{By property (P1)} \\
&\quad \leq \sup_{s_t, a_t} d_c\left(\gamma Z_1(s_{t+1}, a_{t+1}), \gamma Z_2(s_{t+1}, a_{t+1})\right) \,|\, s_{t+1} \sim P, a_{t+1} \sim \pi \\
&\text{By property (P2)} \\
&\quad \leq \gamma \sup_{s_t, a_t} d_c(Z_1(s_{t+1}, a_{t+1}), Z_2(s_{t+1}, a_{t+1})) \,|\, s_{t+1} \sim P, a_{t+1} \sim \pi \\
&\text{Taking the supremum over the state and action independent of } s_t, a_t \\
&\quad \leq \gamma \sup_{s \in \mathcal{S}, a \in \mathcal{A}} d_c(Z_1(s,a), Z_2(s,a)) = \gamma \bar{d}_c(Z_1, Z_2).
\end{aligned}$$

□

Since $\mathcal{T}_H^\pi$ is a γ-contraction in maximal form, it follows from the Banach fixed point theorem:

Corollary 3.1. *Given $Z_{k+1} := \mathcal{T}_H^\pi Z_k$, the series $\{Z_k\}$ converges to the fixed point (distribution) Z^π in the Cramér metric, i.e.*

$$\lim_{k \to \infty} \bar{d}_c(Z_k, Z^\pi) = 0 \quad \textit{with} \quad \mathcal{T}_H^\pi Z^\pi = Z^\pi. \tag{3.5}$$

3.2.4 Policy Improvement

Similar to [30] in Section 2.2.3.2, the information projection for policy improvement is used and the policy is enhanced utilizing the expectation $Q^\pi(s,a) = \mathbb{E}[Z^\pi(s,a)]$,

$$\pi_{k+1} = \operatorname*{argmin}_{\pi \in \Pi} d_{kl}\left(\pi(\cdot \,|\, s_t) \;||\; \frac{\exp(\frac{1}{\alpha}\mathbb{E}[Z^{\pi_k}(s_t, \pi(\cdot \,|\, s_t)])}{W^{\pi_k}(s_t)}\right). \tag{3.6}$$

Theorem 3.3. *Let*

$$\begin{aligned} Q_h^{\pi_k}(s_t, a_t) &:= \mathbb{E}[Z_h^{\pi_k}(s_t, a_t)] \\ &= \mathbb{E}R(s_t, a_t) + \mathbb{E}_{s_{t+1}\sim P, a_{t+1}\sim\pi_k}[\mathbb{E}[Z_h^{\pi_k}(s_{t+1}, a_{t+1})] - \log \pi_k(a_{t+1} \mid s_{t+1})] \end{aligned} \tag{3.7}$$

and assume $|\mathcal{A}| < \infty$. *If* π_{k+1} *is obtained by* (3.6), *then* $Q_h^{\pi_k}(s_t, a_t) \leq Q_h^{\pi_{k+1}}(s_t, a_t) \forall s \in \mathcal{S}$.

Proof. The proof relies on reducing the policy improvement to the SAC form using (3.7): Analogous as in the proof of Lemma 2.7, it can be shown that because of (3.6), it holds $\mathbb{E}_{a_t\sim\pi_{k+1}}[\mathbb{E}[Z^{\pi_k}(s_t, a_t)] - \alpha \log \pi_{k+1}(a_t \mid s_t)] \geq \mathbb{E}_{a_t\sim\pi_k}[\mathbb{E}[Z^{\pi_k}(s_t, a_t)] - \alpha \log \pi_k(a_t \mid s_t)]$. Then the right hand side is expanded similar as in Lemma 2.8 and it can be concluded that $\mathbb{E}[Z_h^{\pi_k}(s_t, a_t)] = Q_h^{\pi_k}(s_t, a_t) \leq \mathbb{E}[Z_h^{\pi_{k+1}}(s_t, a_t)] \forall s \in \mathcal{S}$. □

3.2.5 Policy Iteration

The proof closely follows that of classical reinforcement learning, with the difference of applying the expectation of the distribution.

Theorem 3.4. *If the reward is bounded,* $|\mathcal{A}| < \infty$ *and* $|\mathcal{S}| < \infty$, *repeatedly alternating between distributional policy evaluation, Theorem 3.2, and policy improvement* (3.6) *from any arbitrary policy* $\pi \in \Pi$, *converges to an optimal policy* π^* *according to* (3.2) *and optimal state-action value function* $\mathbb{E}[Z_h^*]$ *as in* (3.7) *under* π^*.

Proof. The assumptions dictate that there is a finite number of policies. From policy improvement it is known that $\mathbb{E}[Z_h^{\pi_{k+1}}] \geq \mathbb{E}[Z_h^{\pi_k}]$ for the series $\{\pi_k\}$, which is bounded above by $\mathbb{E}[Z_h^*] = \mathbb{E}[Z_h^{\pi^*}]$. At convergence, π^* must be the optimal policy of (2.40) since it can not be further improved. □

The theorems above provide a theoretical basis for a Cramér-based distributional SAC approach (C-DSAC). In Section 3.3 we use these insights to construct the C-DSAC algorithm.

3.3 Implementation

In this section, the theoretical results in the previous sections are utilized to derive parameterized objective functions suitable for implementation. To handle the large state and action spaces in complex tasks, neural networks are used as parameterized function approximators; the aim is to optimize their parameters. The parameters for

the random value variable and policy will be denoted by θ and ϕ, respectively. It is assumed that the value variable follows a Gaussian distribution with expectation Q_{θ_1} and standard deviation σ_{θ_2}, i.e. C-DSAC is implemented as a fixed-moment algorithm as described in Section 2.3.5.1.

3.3.1 Objective Functions and Algorithm

Formulas are derived for implementation with parameterized function approximators. Neural networks are utilized for their high generalization capabilities. Previously observed states and actions are stored in a replay buffer $\mathcal{B}$. Implementing the Double-Q method described in Section 2.1.5.1, target networks $\bar{\theta}, \bar{\pi}$ are employed for the critic and actor respectively to further reduce approximation errors.

Evaluation The neural network parameters of Z_θ are computed by the Cramér loss derived from (3.3):

$$J_Z(\theta) := \mathbb{E}_{(s_t,a_t)\sim\mathcal{B}}[d_c(Z_\theta(s_t,a_t), \overline{\mathcal{T}}_H Z_{\bar{\theta}}(s_t,a_t))], \tag{3.8}$$

where $\overline{\mathcal{T}}_H Z_{\bar{\theta}}(s_t,a_t) := R(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim\mathcal{B},a_{t+1}\sim\bar{\pi}}[V_H^{\bar{\pi}}(s_{t+1})]$. The gradient of (3.8), $\nabla_\theta J_Z(\theta)$, is approximated by the gradient of the empirical loss:

$$\hat{\nabla}_\theta J_Z(\theta) = \frac{1}{|\hat{\mathcal{B}}|}\sum_{(s_t,a_t)\in\hat{\mathcal{B}}}[\nabla_\theta d_c(Z_\theta(s_t,a_t), \hat{\mathcal{T}}_H Z_{\bar{\theta}}(s_t,a_t))] \tag{3.9}$$

regarding a mini-batch of sampled tuples $\hat{\mathcal{B}} \subset \mathcal{B}$ and an empirical target approximation $\hat{\mathcal{T}}_H Z_{\bar{\theta}}(s_t,a_t) := R(s_t,a_t) + \gamma\mathbb{E}_{s_{t+1}\sim\hat{\mathcal{B}},a_{t+1}\sim\bar{\pi}}[V_H^{\bar{\pi}}(s_{t+1})]$. In order to simplify the computation of the value distribution, it is assumed that its probability density function is Gaussian denoted by

$$\varphi_{Q,\sigma}(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{1}{2}\left(\frac{(x-Q)}{\sigma}\right)^2}$$

regarding a mean value $Q \in [\underline{Q}, \overline{Q}] \subset \mathbb{R}$ and a variance $\sigma \in [\underline{\sigma}, \overline{\sigma}] \subset \mathbb{R}_+$. For terminal states in deterministic environments, the standard deviation of the target is set to 0_+. The cumulative distribution function of $\varphi_{Q,\sigma}$ is denoted by

$$F_{Q,\sigma}(x) = \int_{-\infty}^{x} \varphi_{Q,\sigma}(t)dt.$$

Let $Q_{\theta_1}(s,a)$ and $\sigma_{\theta_2}(s,a)$ be neural network models of Q and σ over $\mathcal{S}\times\mathcal{A}$ respectively. Then $Z_\theta(s,a)$ denotes a parameterized Gaussian value distribution with parameters $\theta = (\theta_1, \theta_2)$, defined by the distribution function

$$f_{Z_\theta(s,a)} = \varphi_{Q_{\theta_1}(s,a),\sigma_{\theta_2}(s,a)}.$$

Improvement The policy improvement follows from the information projection equation (3.6). Similar to [30] and [69], a reparameterized policy is used to obtain the actions, $a_t = g_\phi(s_t; \xi_t)$ and the aim is to minimize the resulting loss

$$J_\pi(\phi) := \mathbb{E}_{s_t \sim \mathcal{B}, \xi_t \sim \mathcal{N}(0,1)}[\alpha \log \pi_\phi(g_\phi(s_t; \xi_t) \mid s_t) - \mathbb{E}[Z_\theta(s_t, g_\phi(s_t; \xi_t))]] \tag{3.10}$$

to calculate meaningful and low-variance policy gradients $\nabla_\phi J_\pi(\phi)$. The gradient of the empirical loss regarding a mini-batch $\hat{\mathcal{B}} \subset \mathcal{B}$ is thus

$$\hat{\nabla}_\phi J_\pi(\phi) = \frac{1}{|\hat{\mathcal{B}}|} \sum_{s_t \in \hat{\mathcal{B}}, \xi} \log \pi_\phi(a_t \mid s_t) + (\nabla_{a_t} \log \pi_\phi(a_t \mid s_t) - \nabla_{a_t} Q(s_t, a_t)) \nabla_\phi g_\phi(s_t; \xi_t). \tag{3.11}$$

Similar to the value, the action distribution π_ϕ is modeled as a Gaussian. Additionally, the noise $\mathcal{N}(0, 1)$ for each action component is defined as a standard Gauss distribution.

Algorithm The C-DSAC algorithm is presented in Algorithm 3.1, with the Cramér gradient update described in line 11.

Algorithm 3.1 C-DSAC (Cramér-based Distributional Soft Actor-Critic)

1: initialize parameters θ, $\bar{\theta}$, ϕ, $\bar{\phi}$
2: initialize learning rates β_Z, β_π
3: $\mathcal{B} \leftarrow \emptyset$
4: **repeat**
5: **for each** environment step **do**
6: select action $a_t \sim \pi_\phi(\cdot \mid s_t)$
7: observe transition $s_{t+1} \sim P(s_t, a_t)$
8: observe reward $r_t \sim R(s_t, a_t)$
9: $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, a_t, r_t, s_{t+1})\}$
10: **for each** gradient step **do**
11: $\theta \leftarrow \theta + \beta_Z \hat{\nabla}_\theta J_Z(\theta)$, s. (3.9)
12: $\phi \leftarrow \phi + \beta_\pi \hat{\nabla}_\phi J_\pi(\phi)$, s. (3.11)
13: update target networks: $\bar{\theta} \leftarrow \tau\theta + (1-\tau)\bar{\theta}$; $\bar{\phi} \leftarrow \tau\phi + (1-\tau)\bar{\phi}$
14: **until** stopping criterion
15: **return** Z_θ, π_ϕ

3.4 Analysis

The effect of the regularized Cramér distance is investigated on the algorithm dynamics and the aim is to provide a possible explanation for its observed state-of-the-art performance. As discussed in the following sections, C-DSAC has an inherent mechanism to counteract overestimation bias. It is shown that the update of the neural network $Q_{\theta_1}(s_t, a_t)$ using gradients $\nabla_{\theta_1} J_Z(\theta)$ of the Cramér loss function $J_Z(\theta)$ improves the Q-model most where the variance $\sigma(s_t, a_t)$, which is a measure of the uncertainty of Z_θ, is small.

3.4.1 Confidence-Driven Model Updates

To prevent convoluted notation, the regularized Cramér distance is denoted in terms of probability density function of a current distribution $\varphi_{Q,\sigma}$ and a target distribution $\varphi_{Q',\sigma'}$ by

$$C(Q,\sigma) := d_c(\varphi_{Q,\sigma}, \varphi_{Q',\sigma'}) = \int_{-\infty}^{\infty} \big(F_{Q,\sigma}(x) - F_{Q',\sigma'}(x)\big)^2 dx,$$

where Q' and σ' represent the Gaussian parameters of the target distribution.

Lemma 3.5. Let $\varphi_{Q,\sigma}$ and $\varphi_{Q',\sigma'}$ be a current and a target distribution, respectively. Then

$$\frac{\partial}{\partial Q} C(Q,\sigma) = -\frac{2}{\sigma} B(Q,\sigma), \tag{3.12}$$

where

$$B(Q,\sigma) := \int_{-\infty}^{\infty} (F_{Q,\sigma}(x) - F_{Q',\sigma'}(x)) \varphi_{Q,\sigma}(x) dx. \tag{3.13}$$

Proof. Let $f(x) = \varphi_{(0,1)}$ be the standard normal density function and $\Phi = F_{0,1}$ the standard normal cumulative distribution function. From the chain rule we have

$$\frac{\partial}{\partial Q} F_{Q,\sigma} = \frac{\partial}{\partial Q} \Phi\left(\frac{x-Q}{\sigma}\right) = -\frac{1}{\sigma} f\left(\frac{x-Q}{\sigma}\right) = -\frac{\varphi_{Q,\sigma}}{\sigma}.$$

Hence,

$$\begin{aligned}
\frac{\partial}{\partial Q} C(Q,\sigma) &= \int_{-\infty}^{\infty} 2(F_{Q,\sigma}(x) - F_{Q',\sigma'}(x)) \frac{\partial}{\partial Q} F_{Q,\sigma}(x) dx \\
&= \int_{-\infty}^{\infty} -2(F_{Q,\sigma}(x) - F_{Q',\sigma'}(x)) \frac{\varphi_{Q,\sigma}(x)}{\sigma} dx \\
&= -\frac{2}{\sigma} \int_{-\infty}^{\infty} (F_{Q,\sigma}(x) - F_{Q',\sigma'}(x)) \varphi_{Q,\sigma}(x) dx
\end{aligned}$$

□

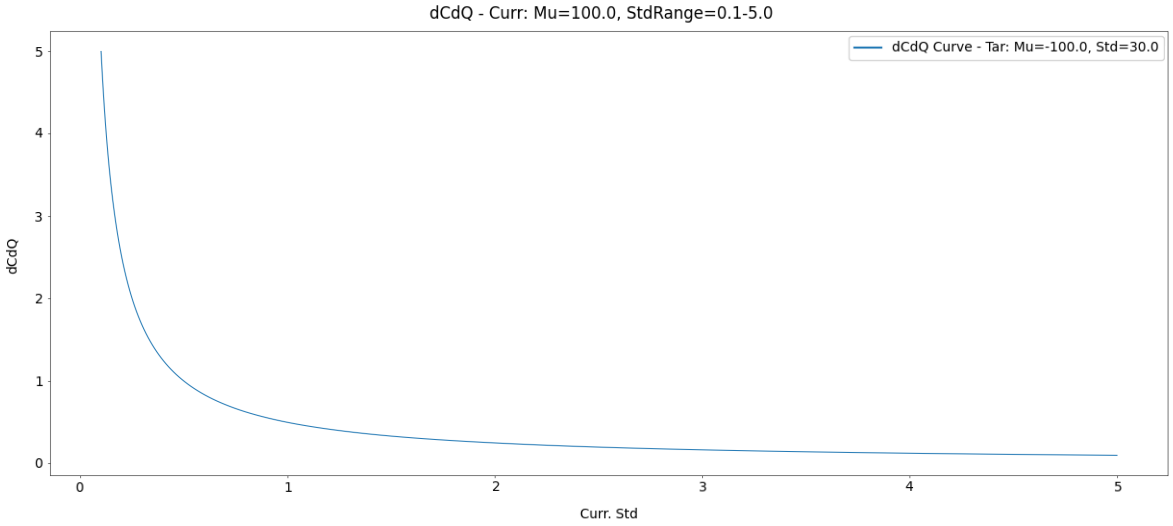


Figure 3.1: Value of $\frac{\partial}{\partial Q}C(Q,\sigma)$ for varying $\sigma \in [\underline{\sigma}, \overline{\sigma}]$

Lemma 3.6. Let $\varphi_{Q,\sigma}$ and $\varphi_{Q',\sigma'}$ be a current and a target distribution, respectively. Then

$$\lim_{\sigma\to\infty} \frac{\partial}{\partial Q} C(Q,\sigma) = 0.$$

Proof. Since $F_{Q,\sigma}(x) \in [0,1]$ and $\|\varphi_{Q,\sigma}\|_1 = 1$, it holds from the Hölder inequality

$$|B(Q,\sigma)| \leq \|(F_{Q,\sigma} - F_{Q',\sigma'}) \cdot \varphi_{Q,\sigma}\|_1 \leq \|F_{Q,\sigma} - F_{Q',\sigma'}\|_\infty \cdot \|\varphi_{Q,\sigma}\|_1 \leq 1.$$

This gives

$$\lim_{\sigma\to\infty} \left| \frac{\partial}{\partial Q} C(Q,\sigma) \right| = \lim_{\sigma\to\infty} \frac{2}{\sigma} \cdot |B(Q,\sigma)| \leq \lim_{\sigma\to\infty} \frac{2}{\sigma} = 0.$$

□

Proposition 3.7. Denote by $\Psi_\theta(s_t,a_t) := \frac{\partial}{\partial Q_{\theta_1}(s_t,a_t)} d_c(Z_\theta(s_t,a_t), \mathcal{T}^\pi Z(s_t,a_t))$ a gradient weight of the Cramér loss function (3.8) such that $\nabla_{\theta_1} J_Z(\theta) = \mathbb{E}_{(s_t,a_t)\sim\mathcal{B}}[\Psi_\theta(s_t,a_t)\nabla_{\theta_1} Q_{\theta_1}(s_t,a_t)]$. Then

$$\lim_{\sigma_{\theta_2}(s_t,a_t)\to\infty} \Psi_\theta(s_t,a_t) = 0. \tag{3.14}$$

Proof. From Lemma 3.5 we have

$$\Psi_\theta(s_t,a_t) = \frac{\partial}{\partial Q_{\theta_1}(s_t,a_t)} d_c(Z_\theta(s_t,a_t), \mathcal{T}^\pi Z(s_t,a_t)) = -\frac{2B(Q_{\theta_1}(s_t,a_t), \sigma_{\theta_2}(s_t,a_t))}{\sigma_{\theta_2}(s_t,a_t)}$$

The statement follows from Lemma 3.6. □

Figure 3.1 shows that the gradient weight $\Psi_\theta(s_t, a_t)$ is quickly reducing, if the variance $\sigma_{\theta_2}(s_t, a_t)$ is increased. The analysis shows that the convergence speed of $Q_{\theta_1}(s_t, a_t)$ depends on the accuracy of $Z_\theta(s_t, a_t)$ and the variance $\sigma_{\theta_2}(s, a)$. If $Z_\theta(s_t, a_t)$ and $\mathcal{T}^\pi Z(s_t, a_t)$ are not similar, then $B(Q_{\theta_1}(s_t, a_t), \sigma_{\theta_2}(s_t, a_t))$ is not small. In this case $\Psi_\theta(s_t, a_t)$ is large, if the variance $\sigma_{\theta_2}(s_t, a_t)$ is small, and has a relatively large impact on the gradient $\nabla_{\theta_1} J_Z(\theta)$, provided that $\nabla_{\theta_1} Q_{\theta_1}(s_t, a_t)$ is not small. Hence, $Q_{\theta_1}(s_t, a_t)$ converges faster at state action pairs (s_t, a_t) where the variance $\sigma_{\theta_2}(s_t, a_t)$ is small, i.e. where the deviation of Q-values is small. The value update is therefore called *confidence-driven*.

3.4.2 Effect on Overestimation Bias

A well-known difficulty in Q-based algorithms is the overestimation of state-action Q-values when using function approximators, e.g. neural networks, in presence of noise, see [47]. In this section, we show that the influence of the overestimation error at (s_t, a_t) on the gradient of the Cramér loss function (3.8) is small, if the variance $\sigma_{\theta_2}(s_t, a_t)$ of $Z_\theta(s_t, a_t)$ is large. We denote by $\tilde{Q}$ and Q^* a noisy and an exact Q-value, respectively. Consider a noisy target

$$\tilde{Y}(s_t, a_t) := R(s_t, a_t) + \gamma Z(s_{t+1}, a_{t+1}), \quad a_{t+1} = \operatorname*{argmax}_{a \in \mathcal{A}} \tilde{Q}(s_{t+1}, a)$$

and an exact target by

$$Y^*(s_t, a_t) := R(s_t, a_t) + \gamma Z(s_{t+1}, a_{t+1}), \quad a_{t+1} = \operatorname*{argmax}_{a \in \mathcal{A}} Q^*(s_{t+1}, a).$$

Proposition 3.8. Let $\Delta\Psi_\theta(s_t, a_t) := |\tilde{\Psi}(s_t, a_t) - \Psi^*(s_t, a_t)|$ be the gradient weight error of the Cramér loss function (3.8) regarding noisy and exact gradient weights, where

$$\tilde{\Psi}(s_t, a_t) := \frac{\partial}{\partial Q_{\theta_1}} d_c(Z_\theta(s_t, a_t), \tilde{Y}(s_t, a_t)), \quad \Psi^*(s_t, a_t) := \frac{\partial}{\partial Q_{\theta_1}} d_c(Z_\theta(s_t, a_t), Y^*(s_t, a_t)).$$

Then

$$\lim_{\sigma_{\theta_2}(s_t, a_t) \to \infty} \Delta\Psi_\theta(s_t, a_t) = 0. \tag{3.15}$$

Proof. Let B^* and $\tilde{B}$ the value B of (3.13) calculated using Y^* and $\tilde{Y}$ respectively. From Lemma 3.5 we have

$$\left|\tilde{\Psi}(s_t, a_t) - \Psi^*(s_t, a_t)\right| = 2 \left| \frac{\tilde{B}_\theta(s_t, a_t)}{\sigma_{\theta_2}(s_t, a_t)} - \frac{B^*_\theta(s_t, a_t)}{\sigma_{\theta_2}(s_t, a_t)} \right| \leq \frac{2}{\sigma_{\theta_2}(s_t, a_t)},$$

since $|B^*_\theta(s_t, a_t)| \leq 1$ and $|\tilde{B}_\theta(s_t, a_t)| \leq 1$. This proves (3.15). □

Figure 3.2 shows that the gradient weight error $\Delta\Psi_\theta(s_t, a_t)$ is quickly reducing if the variance $\sigma_{\theta_2}(s_t, a_t)$ is increased. This demonstrates that the influence of the overestimation error at (s_t, a_t) on the gradient $\hat{\nabla}_\theta J_Z(\theta)$ of the Cramér loss function (3.8) is small, if the variance $\sigma_{\theta_2}(s_t, a_t)$ is large.

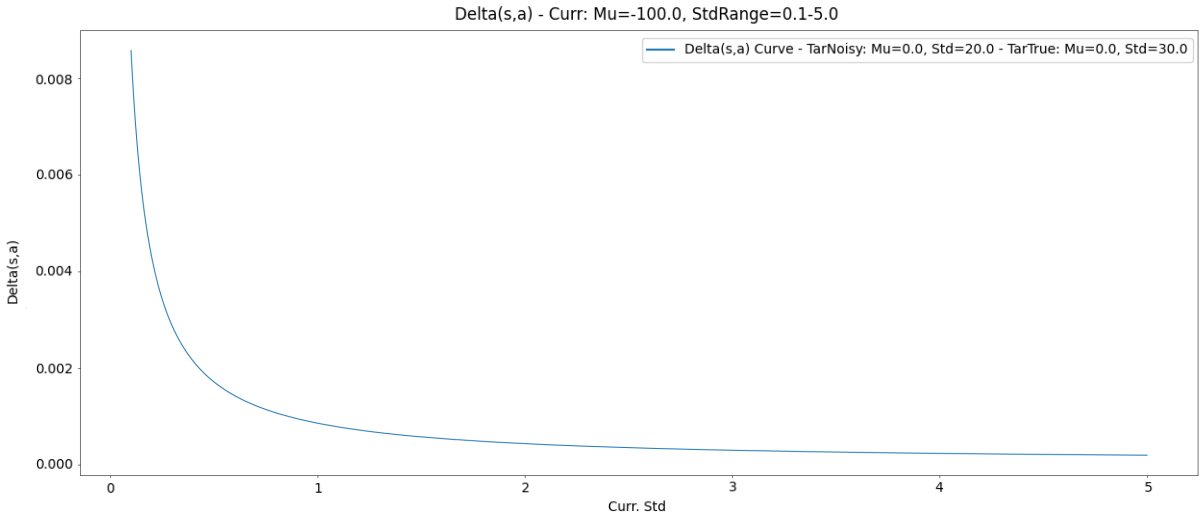


Figure 3.2: Value of $\Delta\Psi$ for varying $\sigma \in [\underline{\sigma}, \overline{\sigma}]$

3.5 Experiments

The experiments are conducted with the same hyperparameter values as in [30], except for C-DSAC-specific ones (e.g, limits on standard deviation, ... etc.). Auxiliary details to the main algorithm and "code-level optimization" can have a significant impact on the performance of an algorithm [84]. It is assumed that the original implementation used for the baseline results in [30] did not employ additional optimization techniques such as value function clipping, orthogonal initialization, layer scaling, learning rate annealing, reward/observation clipping, or observation and reward normalization. To be consistent with these assumptions, we will refrain from employing these code-level optimization techniques in our implementation. If the following experiments would be run with the same network architecture as in the SAC experiments, C-DSAC would have an unfair advantage due to its dual-output critic. The last hidden layer is reduced by one neuron. As a result, our critic has slightly fewer parameters than those reported in the original SAC configuration, see Appendix 3.5.1. Although [81] and [80] claim

superior performance over SAC baselines, it is important to note that the former uses more complex neural network architectures with batch normalization for the value approximator. We were unable to replicate the results of the latter.

3.5.1 Hyperparameters

The hyperparameters values in the C-DSAC experimental runs were carefully selected to ensure a fair and consistent comparison with the SAC baselines.

Table 3.1: C-DSAC Hyperparameters

Parameter	**Value**
Shared	
Optimizer	Adam [85]
Learning rate	$3 \cdot 10^{-4}$
Discount factor (γ)	0.99
Replay buffer size	10^6
Actor architecture	(256, 256)
Minibatch size	256
Layer activation	ReLU
Target smoothing coefficient (τ)	0.005
Target update interval	1
Gradient steps	1
SAC	
Critic architecture	(256, 256)
C-DSAC	
Critic architecture	(256, 255)
Min., Max. Std.	0.01, 1000
Gauss supports	31
Gauss bounds	15σ

3.5.2 Comparable Evaluation with SAC Baselines

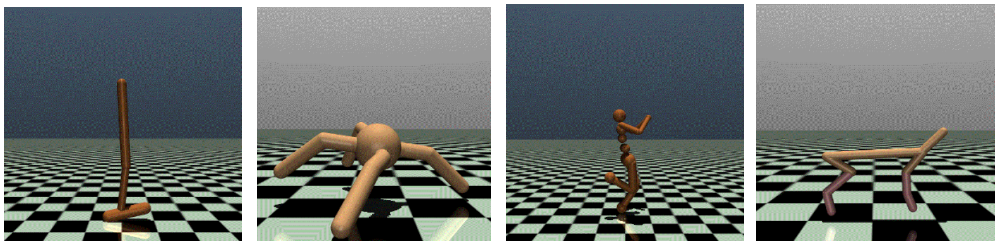
The goal of the experiments is to evaluate the impact of the Cramér loss on the entropy-augmented actor-critic framework across diverse and complex environments, see Figure 3.3: Hopper-v4, $(\mathcal{S} \times \mathcal{A}) \in \mathbb{R}^{11} \times \mathbb{R}^3$; Ant-v4 $(\mathcal{S} \times \mathcal{A}) \in \mathbb{R}^{111} \times \mathbb{R}^8$; Humanoid-v4 $(\mathcal{S} \times \mathcal{A}) \in \mathbb{R}^{376} \times \mathbb{R}^{17}$; HalfCheetah-v4 $(\mathcal{S} \times \mathcal{A}) \in \mathbb{R}^{17} \times \mathbb{R}^6$; Walker2d-v4 $(\mathcal{S} \times \mathcal{A}) \in \mathbb{R}^{17} \times \mathbb{R}^6$. Following the recommendations of [86], the evaluation rollout for the mean

action of the target policy is reported every 1,000 iterations, with an episode length of 1,000 steps. The agent is trained for one million iterations in each environment. The results are averaged over five random seeds. The graphs in Figure 3.4 showcase (among others) the evaluation performances of C-DSAC and SAC in the benchmark environments (Figure 3.3). Especially in the most difficult environment, Humanoid-v4, C-DSAC clearly outperforms SAC and reaches an average reward of about 5730, a value that is achieved by SAC only after approximately four million iterations. Note that at the end of the C-DSAC training, the curve still has an upward trend. Although C-DSAC and SAC reach the same average rollout in the Hopper-v4 environment, C-DSAC does so considerably faster. The instability after 400,000 iterations can be due to the occurrence of catastrophic forgetting in one of the runs. However, it manages to recover. In the Walker2d-v4 environment, C-DSAC learns much quicker and obtains a higher average reward, albeit a less stable training process. The least stable training occurs in the Ant-v4 environment, where high fluctuations can be observed. The average seems to surpass the SAC baselines. The best performing target and online policies support that claim, see Table 3.2. In HalfCheetah-v4, C-DSAC obtains an average reward of less than 10000 in the end of training, a value well surpassed by SAC. However, evaluation of the best online policy indicates that C-DSAC can reach a similar performance, see Table 3.2.

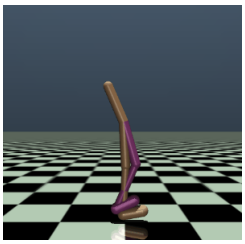
To provide more details about the performance, Table 3.2 presents the best model performances across all environments, reporting average episodic rewards over 100 runs. When there is a significant performance gap between target and online policies, results for both are included.

Table 3.2: Average performance over 100 runs of the best models in the training runs

Environment	Best Performance
Hopper-v4	Target: 3352 ± 97
Ant-4	Target: 4375 ± 299 Online: 5376 ± 79
Humanoid-v4	Target: 5715 ± 255
HalfCheetah-4	Target: 9124 ± 309 Online: 10023 ± 228
Walker-v4	Target: 4808 ± 223

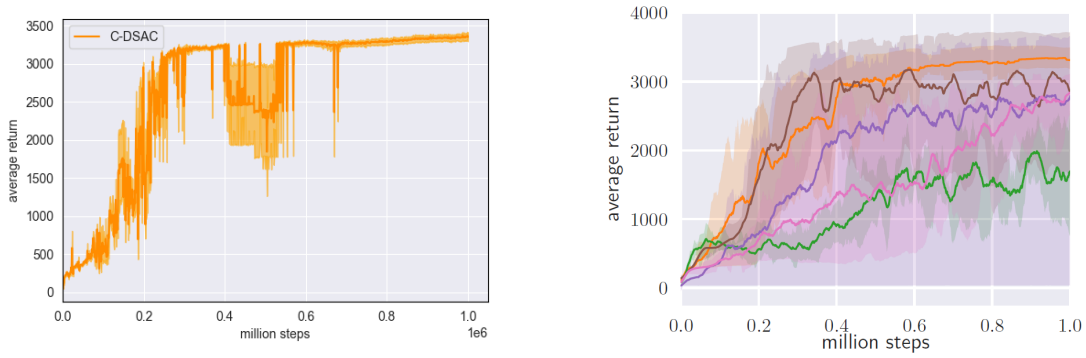


(a) Hopper-v4 **(b)** Ant-v4 **(c)** Humanoid-v4 **(d)** HalfCheetah-v4

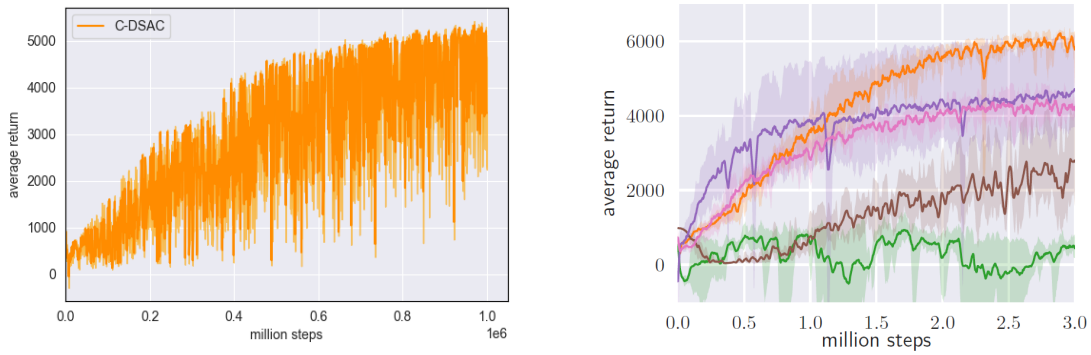


(e) Walker2d-v4

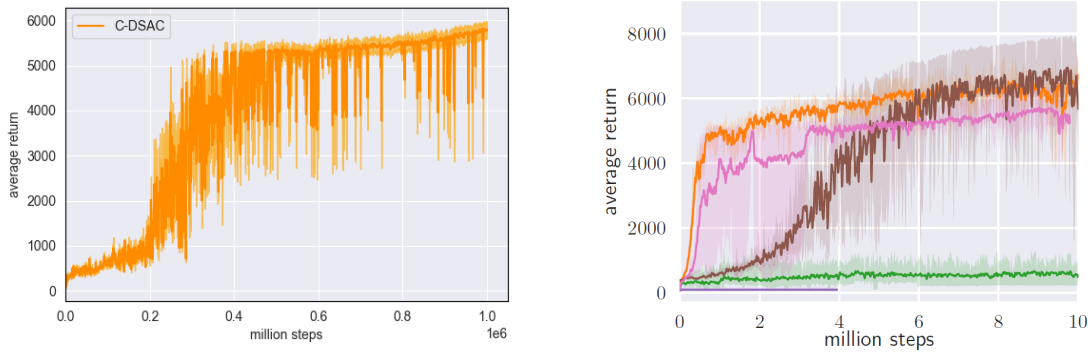
Figure 3.3: Testing environments



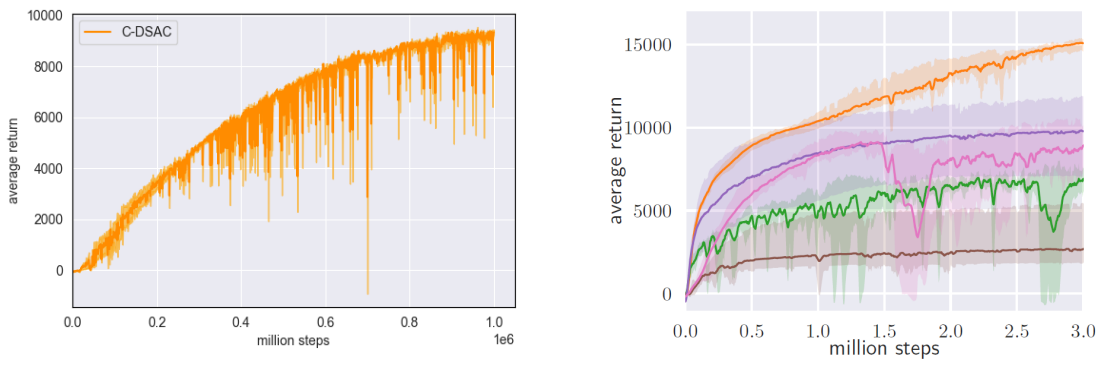
(a) Hopper



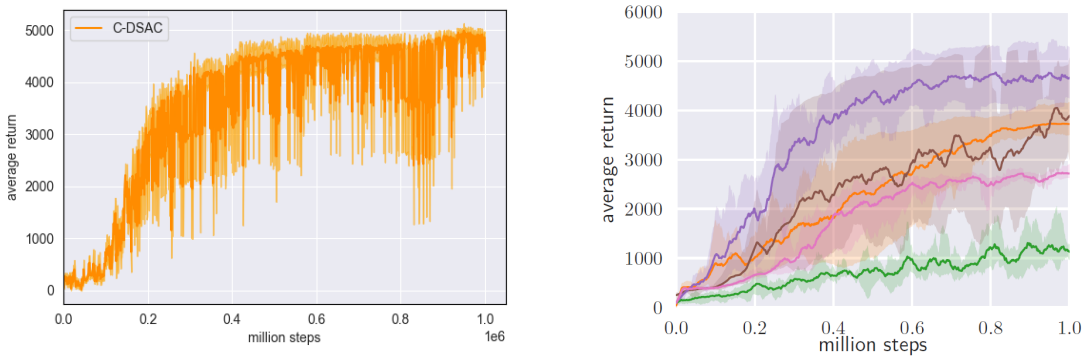
(b) Ant (left one million, right three million steps)



(c) Humanoid (left one million, right ten million steps)



(d) HalfCheetah (left one million, right three million steps)



(e) Walker

Figure 3.4: Orange curves represent C-DSAC (left) and SAC (right, from [30]). Other curves show performances of DDPG (green), PPO (brown), SQL (pink), and TD3 (violet).

3.5.3 Comparative Implementation Evaluation against a Standard Baseline

To assess the quality of the C-DSAC implementation - and therefore its effect on the performance, its code is modified to revert it to SAC by adjusting the loss function and critic network architecture. The experiments are repeated on HalfCheetah-v4 with five runs, Figure 3.5. The SAC hyperparameter are set to values given in Appendix 3.5.1. The custom SAC implementation based on the C-DSAC code achieved an average reward of slightly above 3000, compared to over 10000 in the original SAC implementation after one million iterations. This indicates that the C-DSAC implementation negatively impacts overall performance.

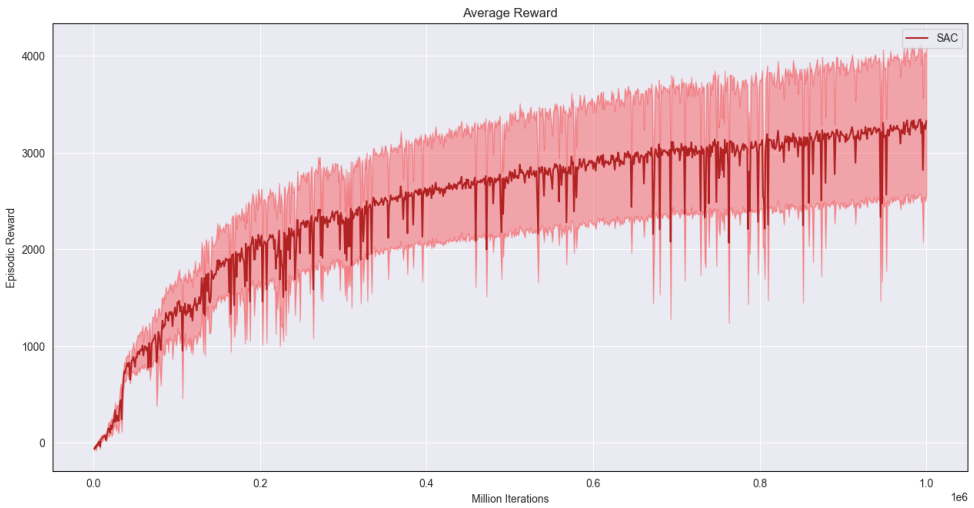


Figure 3.5: Performance of SAC based on C-DSAC implementation on HalfCheetah-v4

3.6 Conclusion

This chapter presented C-DSAC, a distributional maximum-entropy deep reinforcement learning algorithm. It uses a loss function based on the regularized Cramér distance for calculating expected Q-values and related variances. It was demonstrated that distributional soft policy iteration based on the regularized Cramér distance converges to an optimal policy. Numerical experiments show that C-DSAC outperforms SAC in most benchmark environments despite a sub-optimal implementation. The perfor-

mance seems to be superior especially for complex environments, like the Humanoid-v4 environment.

The presented analysis suggests that the superior performance of C-DSAC may among others be attributed to a confidence-driven update of Q-values. It was shown that gradient weights of the loss function are small at state-action pairs with high variance. This results in a more conservative update for high-variance targets, reducing adaptation to potentially overestimated values. In this context, high variance indicates low confidence in the target, which improves only after repeated observations. We observed that as a consequence, C-DSAC learns Q-values in a two-phase process: First, building confidence in the target, then adapting to its expected value. While this confidence-building phase may imply slow and unstable learning, the numerical results indicate that the reduction in the effect of overestimation outweighs the additional cost.

There are many ways to improve C-DSAC. Experiments with different initialization values can be carried out, particularly for the variance approximation parameters. For a fair comparison with SAC baselines, automatic temperature adjustment was omitted, but could be integrated to enhance performance. Additionally, an exploration strategy based on the value distribution can be developed. C-DSAC models the return with a fixed-moment distribution. Efforts can be made to efficiently apply GMMs with complex neural network approximators to allow for a more expressive return distribution. Alternatively, the Crameŕ loss can be used in conjunction with implicitly modeling the distribution through quantile functions, similar to IQN (Section 2.3.4.3). Another promising direction for research is the utilization of task and reward decomposition based on the shape of the value distribution.

4

Model-Enhanced Reinforcement Learning for Robotic Manipulation with Inverse Kinematics

This chapter explores the application of RL algorithms in the field of robotics and presents a novel approach designed specifically for robotic manipulation tasks. The proposed RL algorithm, termed "Reinforcement Learning - Inverse Kinematics" (RL-IK) [87], integrates the knowledge of a robot's kinematics and can be interpreted as a model-based RL algorithm. This allows for more efficient learning by leveraging the underlying physical model of the robot, particularly because movement of the robot and planning the task are separated and assigned to different modules. The focus of this research is on multi-joint robots in $3D$ space, controlled by policies that are learned within the actor-critic (AC) framework, see Section 2.1.2.4. The algorithm aims to enhance the performance of robots in tasks involving manipulation and positioning by combining inverse kinematics with reinforcement learning for planning and decision-making at a higher abstraction level. According to ISO 8373, a robot is defined as an autonomous, programmed actuator ensemble capable of locomotion, manipulation, or tasks involving positioning. An industrial robot, specifically, is described as an autonomous, reprogrammable, multi-purpose manipulator with three or more programmable axes, which may be stationary or mounted on a mobile platform [88]. In this context, RL-IK is particularly suited for industrial robots performing complex manipulation tasks. Although the experiments in this chapter are conducted using the Franka Emika Panda robot [89], Figure 4.1, RL-IK is generalizable and can be applied to a wide range of robotic platforms, making it a versatile tool for various robotic manipulation applications.

Figure 4.1: Franka Emika Panda robot with a gripper end-effector

4.1 Forward Kinematics

Robotic systems consist of rigid links connected by joints, forming a kinematic chain that defines the robot's geometry and degrees of freedom. Each link and joint is associated with a reference coordinate frame to ensure consistency in the description of joint configurations and the kinematic state. Forward Kinematics focuses on determining the position and orientation of the end-effector – the last link of the robot used for manipulation and other tasks – as a function of the robot's joint configuration. Examples of end-effector tasks include gripping, tool operation and interaction with the environment. The end-effector can move freely, thus rendering the kinematic chain as open. The robot's configuration in joint space can be described as a vector of n joint parameters:

$$\mathbf{v}_t = [\mathbf{v}_1, \mathbf{v}_2, \ldots, \mathbf{v}_n],$$

where $\mathbf{v}_i$ represents joint-specific parameters such as angles for revolute joints or displacement for prismatic joints. In robotics, the goal is to find a sequence

$$\{\mathbf{v}_{t_1}, \ldots, \mathbf{v}_{t_T}\}$$

to complete a given task without violating dynamic and environmental constraints. To accurately describe the influence of joint parameters on the end-effector, a consistent application of transformations between coordinate frames, starting from the base link to the end-effector, is required. A widely adopted convention in mechatronics is the Denavit-Hartenberg (DH) convention, which standardizes the assignment of coordinate frames along the kinematic chain. A transformation between frames consists of rotational and translational components. Rotations are represented using a 3×3 rotational matrix $\mathbf{R}$, such that the orthogonality property $\mathbf{R}\mathbf{R}^T = \mathbf{I}$ holds. Further, the

determinant is constrained to be 1 ensuring preservation of link lengths and reference frame orientation. Rotations are defined around the z-axis (yaw). Displacements are represented by a 3×1 translation vector $\mathbf{d}$.

Under the Denavit-Hartenberg (DH) convention, illustrated in Figure 4.2, the assignment of coordinate frames follows a standardized methodology to describe transformations between links. The z-axis is aligned with the axis of rotation or translation for each joint, while the x-axis for the next frame is defined as the common normal between the z-axes, with the new origin positioned at the intersection between the subsequent z-axis and the common normal. The transformations between coordinate frames are characterized by four parameters. The angle ϑ_{k-1} specifies the rotation required to align the base frame x-axis with that of the next frame. The parameter $r_{k-1,k}$ represents the length of the common normal between the z-axes of consecutive joints. The parameter d_k denotes the distance along the previous z-axis to the intersection with the new x-axis. Finally, the angle $\alpha_{k-1,k}$ defines the rotation around the new x-axis necessary to align the two z-axes. With these four parameters, the frame transformations can be described in terms of screw displacements consisting of a translation **TRA** and rotation function **ROT** around the z- and x-axis. The transformation matrix from frame $k-1$ to frame k is given by

$$^{k-1}\mathbf{T}_k(d_{k-1}, \vartheta_{k-1}, r_{k-1,k}, \alpha_{k-1,k}) = \mathbf{Z}_{k-1}(d_{k-1}, \vartheta_{k-1}) \cdot \mathbf{X}_k(r_{k-1,k}, \alpha_{k-1,k}), \tag{4.1}$$

where

$$\mathbf{Z}_{k-1}(d_{k-1}, \vartheta_{k-1}) = \mathbf{TRA}_{\mathbf{Z}_{k-1}}(d_{k-1}) \cdot \mathbf{ROT}_{\mathbf{Z}_{k-1}}(\vartheta_{k-1})$$

and

$$\mathbf{X}_k(r_{k-1,k}, \alpha_{k-1,k}) = \mathbf{TRA}_{\mathbf{X}_k}(r_{k-1,k})\, \mathbf{ROT}_{\mathbf{X}_k}(\alpha_{k-1,k}).$$

The matrices associated with translation and (counterclockwise) rotation on these axes are explicitly defined by

$$\mathbf{TRA}_{\mathbf{Z}_{k-1}}(d_{k-1}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_{k-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{ROT}_{\mathbf{Z}_{k-1}}(\vartheta_{k-1}) = \begin{bmatrix} \cos(\vartheta_{k-1}) & -\sin(\vartheta_{k-1}) & 0 & 0 \\ \sin(\vartheta_{k-1}) & \cos(\vartheta_{k-1}) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

and

$$\mathbf{TRA}_{\mathbf{X}_k}(r_{k-1,k}) = \begin{bmatrix} 1 & 0 & 0 & r_{k-1,k} \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \quad \mathbf{ROT}_{\mathbf{X}_k}(\alpha_{k-1,k}) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha_{k-1,k}) & -\sin(\alpha_{k-1,k}) & 0 \\ 0 & \sin(\alpha_{k-1,k}) & \cos(\alpha_{k-1,k}) & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

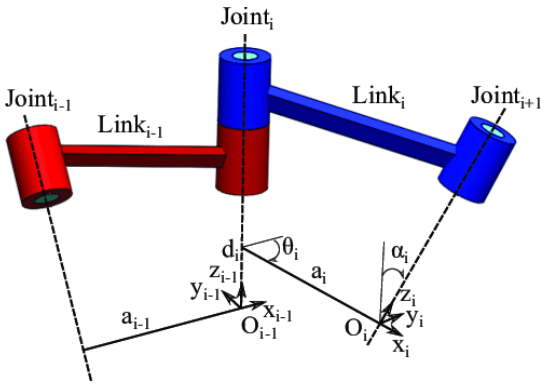


Figure 4.2: Illustration of the Denavit-Hartenberg convention [90]. Reproduced under CC BY 4.0 license (`https://creativecommons.org/licenses/by/4.0/`).

Performing the sequential matrix operations between two links in the kinematic chain as described in (4.1) yields

$$ {}^{k-1}\mathbf{T}_k = \begin{bmatrix} \cos(\vartheta_{k-1}) & -\sin(\vartheta_{k-1})\cos(\alpha_{k-1,k}) & \sin(\vartheta_{k-1})\sin(\alpha_{k-1,k}) & r_{k-1,k}\cos(\vartheta_{k-1}) \\ \sin(\vartheta_{k-1}) & \cos(\vartheta_{k-1})\cos(\alpha_{k-1,k}) & -\cos(\vartheta_{k-1})\sin(\alpha_{k-1,k}) & r_{k-1,k}\sin(\vartheta_{k-1}) \\ 0 & \sin(\alpha_{k-1,k}) & \cos(\alpha_{k-1,k}) & d_{k-1} \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{4.2} $$

Note that the function arguments in Equation (4.2) were omitted to prevent convoluted notation. By sequentially applying these transformations across all links in the chain, the end-effector's position and orientation relative to the base frame can be calculated as a function of the angular displacement $\boldsymbol{\vartheta}$ and linear displacement $\mathbf{d}$. Note that $\mathbf{r}$ and $\boldsymbol{\alpha}$ are constants derived from the robot's geometry and are therefore not explicitly written as function arguments. The overall transformation is then given by

$$ \mathbf{T}_{EE}(\mathbf{d}, \boldsymbol{\vartheta}) = {}^{1}\mathbf{T}_n(\mathbf{d}, \boldsymbol{\vartheta}) = {}^{1}\mathbf{T}_2(d_1, \vartheta_1)\, {}^{2}\mathbf{T}_3(d_2, \vartheta_2) \ldots {}^{n-1}\mathbf{T}_n(d_{n-1}, \vartheta_{n-1}). $$

For the rest of the work, it is assumed that the kinematic chain does not include prismatic joints, i.e. d_{k-1} is a constant. The end-effector's state can be expressed as

$$ \mathbf{f}(\boldsymbol{\vartheta}) = {}^{1}\mathbf{T}_n(\boldsymbol{\vartheta}). \tag{4.3} $$

4.2 Inverse Kinematics

Inverse kinematics (IK) is the inverse operation of forward kinematics, aiming to determine a joint configuration $\boldsymbol{\vartheta}^*$ that positions and orients the end-effector at a desired target $\mathbf{p}^*$ while respecting physical constraints. For rigid end-effectors, analytical solutions exist only for 6-DoF manipulators. When the robot deviates from this configuration, the resulting system of equations might become under- or overdetermined. In overdetermined systems, additional constraints (e.g., energy efficiency, collision avoidance) must be considered, which often complicates finding a solution. Singularities –configurations where the robot loses a degree of freedom – must be exempt from the set of solutions. Jacobian-based numerical methods are widely used to address these challenges. These methods leverage the Jacobian matrix, which relates joint velocities to end-effector velocities. The roll η, pitch ξ, and yaw ψ can be extracted from the transformation matrix of the end-effector in (4.3):

$$\begin{aligned} \eta &= \operatorname{atan2}(T_{EE_{32}}, T_{EE_{33}}) \\ \xi &= \operatorname{atan2}\left(-\frac{T_{EE_{31}}}{\sqrt{T_{EE_{32}}^2 + T_{EE_{33}}^2}}\right) \\ \psi &= \operatorname{atan2}(T_{EE_{21}}, T_{EE_{11}}), \end{aligned} \tag{4.4}$$

where M_{ij} indicates the i-th row in column j of matrix $\mathbf{M}$ and atan2 is the piecewise defined inverse tan-function that is well-defined everywhere except at $\operatorname{atan2}(0,0)$. Then, the end-effector's position and orientation can be compactly expressed as

$$\mathbf{p}(\boldsymbol{\vartheta}) = \begin{bmatrix} x & y & z & \eta & \xi & \psi \end{bmatrix},$$

where the elements depend on $\mathbf{f}(\boldsymbol{\vartheta})$ according to Equation (4.4) This representation simplifies the subsequent IK equations. The end-effector's rate of change as a function of the joint configuration can be expressed by the Jacobian

$$\mathbf{J}_p := \frac{\partial \mathbf{p}(\boldsymbol{\vartheta})}{\partial \boldsymbol{\vartheta}}.$$

The joint configuration is iteratively updated by

$$\boldsymbol{\vartheta}_{\ell+1} = \boldsymbol{\vartheta}_\ell + \Delta\boldsymbol{\vartheta}. \tag{4.5}$$

Using a modified Gauss-Newton approach and utilizing the Moore-Penrose pseudoinverse of the Jacobian $\mathbf{J}_p^+ = (\mathbf{J}_p^T \mathbf{J}_p)^{-1} \mathbf{J}_p^T$, the update can be approximated as

$$\Delta\boldsymbol{\vartheta} \approx \mathbf{J}_p^+ (\mathbf{p}^* - \mathbf{p}(\boldsymbol{\vartheta})).$$

The joint configuration is updated until the error defined by the difference between the current state and target state $\mathbf{p}^*$ satisfies the convergence criterion $|\mathbf{p}^* - \mathbf{p}| \leq \boldsymbol{\epsilon}$. This update arises from a linearization around $\mathbf{p}(\boldsymbol{\theta})$. It ensures minimal change in the joint configuration in the direction of a (local) minimum. The pseudo-inverse of the Jacobian is employed to address its potential non-invertibility, which arises due to the Jacobian often being non-square in robotic scenarios. However, this approach results in large joint updates in or near singular configurations, where the Jacobian loses rank. To mitigate numerical instabilities, the Levenberg method replaces the pseudo-inverse with a "dampened" inverse [91]:

$$\Delta\boldsymbol{\vartheta} = (\mathbf{J}_p^T\mathbf{J}_p + \nu\mathbf{I})^{-1}\mathbf{J}_p^T(\mathbf{p}^* - \mathbf{p}(\boldsymbol{\vartheta})), \tag{4.6}$$

where $\mathbf{I}$ is the identity matrix and $\nu \geq 0$ is the damping factor. Note that Equation (4.6) reduces to the modified Gauss-Newton if $\nu = 0$, whereas it approaches a gradient descent step for large ν,

$$\Delta\boldsymbol{\vartheta} \approx \frac{1}{\nu}\mathbf{J}_p^T(\mathbf{p}^* - \mathbf{p}(\boldsymbol{\vartheta})) \quad \text{if} \quad \nu >> 0.$$

The identity matrix induces an update that affects all parameter directions equally, meaning that differences in parameter contributions to the error is disregarded. To respect the error surface curvature, the identity matrix is replaced by the Gram matrix of the Jacobian, yielding the Levenberg-Marquardt update step [92]:

$$\Delta\boldsymbol{\vartheta} = \left(\mathbf{J}_p^T\mathbf{J}_p + \nu\,\mathrm{diag}(\mathbf{J}_p^T\mathbf{J}_p)\right)^{-1}\mathbf{J}_p^T(\mathbf{p}^* - \mathbf{p}(\boldsymbol{\vartheta})) \tag{4.7}$$

In the Levenberg-Marquardt algorithm (LMA), the current joint configuration is updated according to Equation (4.7) until the error between $\mathbf{p}^*$ and $\mathbf{p}$ is sufficiently small. LMA balances the effect of Gauss-Newton and gradient descent updates, with the former offering precision and the latter providing stability.

4.3 Separation of Planning and Execution of Movement: RL-IK

Motion planning is an abstraction of end-effector movements, where the manipulations required to achieve a task are represented as a sequence of end-effector states over discrete time steps. In the context of robot-specific reinforcement learning (RL), it is natural to distinguish between motion planning and the physical execution of the robot's movements. This separation allows for a more modular and interpretable control

architecture. "Reinforcement Learning - Inverse Kinematics" (RL-IK) operates on the basis of this task separation explicitly: At each decision step, the RL agent generates a target position and orientation for the end-effector based on its policy:

$$\mathbf{a}_t = \mathbf{p}^* \sim \pi(s_t).$$

The low-level controller, implemented using the Levenberg-Marquardt Algorithm (LMA), then assumes the responsibility of achieving the desired position and pose. No additional high-level actions are inferred from the RL policy until the end-effector reaches the target configuration. While the inverse kinematics phase is active, constraint violations – such as collisions, excessive force application, or breaches of time limits – must still be monitored. If such a violation occurs, the "IK phase" is prematurely terminated, the environment resets, and an appropriate reward is emitted to guide future policies. A detailed description of the RL-IK algorithm is described in Algorithm 4.1.

The adoption of LMA as a low-level controller, instead of allowing the RL agent to directly control individual joints, offers several advantages:

1. Reduction in action space: Regardless of the robots DoF, the action space will always consist of a fixed number of actions that describe the end-effector's position and orientation. This reduces the dimensionality of the RL problem and shifts computational complexity from the sample-inefficient RL component (see Section 2.1.3.3) to the computationally efficient Levenberg-Marquardt optimization unit [93].
2. Experimental simulations indicate that robot movements exhibit smoother trajectories with RL-IK. This smoothness can likely be attributed to joint updates that align with consistent spatial directions dictated by the LMA solver.
3. While LMA is generally robust to rank loss, a poorly chosen damping factor ν can still result in singular configurations. In such cases, the agent may struggle to achieve its objectives. However, since the agent is designed to maximize a task-specific reward, it implicitly learns to avoid configurations that hinder efficient movement and manipulation capabilities.
4. Instead of evaluating and improving the policy at every intermediate step, it suffices to perform these operations after the end-effector transitions from its current pose $\mathbf{p}$ to the target pose $\mathbf{p}^*$. This design generates fewer but qualitatively superior experience points. While this approach could theoretically exacerbate the credit assignment problem, experimental results in Section 4.5 indicate significant faster convergence to optimal rewards and improved training stability compared to direct joint-level control methods.

Algorithm 4.1 RL-IK

Initialize $\boldsymbol{\vartheta}$
$\mathcal{B} \leftarrow \emptyset$
$\mathbf{p}_t \leftarrow \mathbf{p}(\boldsymbol{\vartheta})$ # from forward kinematics
repeat
    Receive initial observation state s_1
    for each environment step **do**
        $\mathbf{p}_t^* \sim \pi(s_t)$
        repeat
            $\Delta\boldsymbol{\vartheta}$ from Equation(4.7) using $\mathbf{p}_t^*$ and $\mathbf{p}_t$
            $\boldsymbol{\vartheta} \leftarrow \boldsymbol{\vartheta} + \Delta\boldsymbol{\vartheta}$
            $\mathbf{p}_t \leftarrow \mathbf{p}(\boldsymbol{\vartheta})$ # from forward kinematics
            if constraint violation **then**
                Break
        until $|p - p^*| \leq \epsilon$
        Observe transition $s_{t+1} \sim P(s_t, a_t)$
        Observe reward r_t
        $\mathcal{B} \leftarrow \mathcal{B} \cup \{(s_t, \mathbf{p}_t^*, r_t, s_{t+1})\}$
    Perform gradient steps
until Stopping criterion
return models

4.4 Reward Engineering for RL-IK

In RL-IK, designing an effective reward function is a critical factor for achieving successful motion planning and control. The reward function must carefully balance multiple objectives, including event-free motion trajectories (where no constraint violations occur), temporal efficiency, control cost, and partial task fulfilment. An improperly balanced reward structure can significantly hinder convergence or even result in undesirable "suicidal" behaviour, as discussed in Section 2.1.4. The reward function in RL-IK is composed of the following sub-rewards:

1. **Costs:** These include penalties for excessive time use and high control effort. Both must be minimized to ensure efficiency and smooth operation.
2. **Critical Rewards:** Collisions must be heavily penalized, but the temporal context is crucial. Collisions occurring earlier in the trajectory should be penalized more severely than those occurring later, as earlier collisions often indicate more

fundamental planning errors. However, severely punishing later collisions will hinder exploration and encourage the agent to terminate episodes early.

3. **Positive Reward:** Successfully gripping the target object and moving it towards the target location provides a positive reinforcement signal, incentivizing goal achievement.
4. **Ambivalent Rewards:** Factors such as the distance to the target object and alignment with the target position can be interpreted both positively and negatively, depending on the specific stage of the task and the overarching objective.

In a real-world scenario, image-based distance approximators and tactile registrators might be utilized to obtain spatial proximity data and detect collisions. This nuanced reward design ensures that the RL-IK can optimize both high-level motion planning and low-level control execution, ensuring to complete the robotic task efficiently.

4.5 Experiments

The performance of RL-IK was assessed in a simulated robotic task illustrated in Figure 2.1. Successful task completion meant that the robot picked up a target object, transported it over a wall and placed it in a tray without colliding with the environment. PPO (Section 2.1.6.4) and a modified version of DDPG (Section 2.1.6.5) known as TD3 were employed to solve the task. IK-augmented variants of these algorithms, PPO-IK and TD3-IK, were then used in the same environment. Figure 4.3 shows the performances of PPO and PPO-IK over five runs. Because each step in the IK variant is a planned sub-trajectory rather than an incremental movement of the end-effector, PPO-IK needs considerably fewer "RL iterations" than PPO to converge to a task-solving reward. To allow comparison, the performance is reported as a function of wall-clock time in hours. All experiments were run on the same system and under the same circumstances. Figure 4.4 shows the performance for a similar set of experiments conducted with the TD3 algorithm. Although both variants do not manage to solve the task in a feasible time, TD3-IK converges to a significantly higher suboptimal reward. Because TD3 is highly hyperparameter sensitive, it is hypothesized that the parameters were not correctly configured for the task.

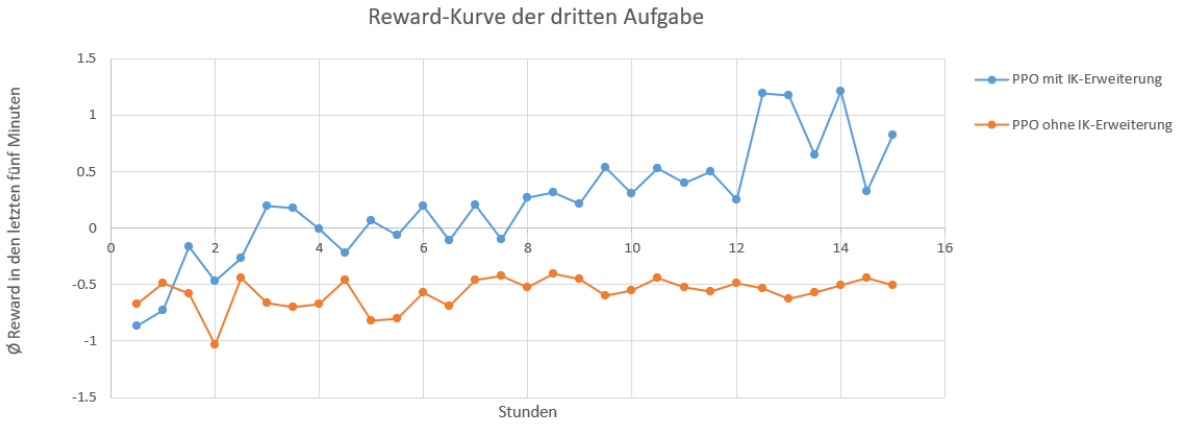


Figure 4.3: PPO used in the robotic task with and without inverse kinematics; graph taken from [87]. The difference in step count between the base and IK-augmented variant is drastic in favour of PPO-IK, therefore the graph illustrates the agents performance in hours.

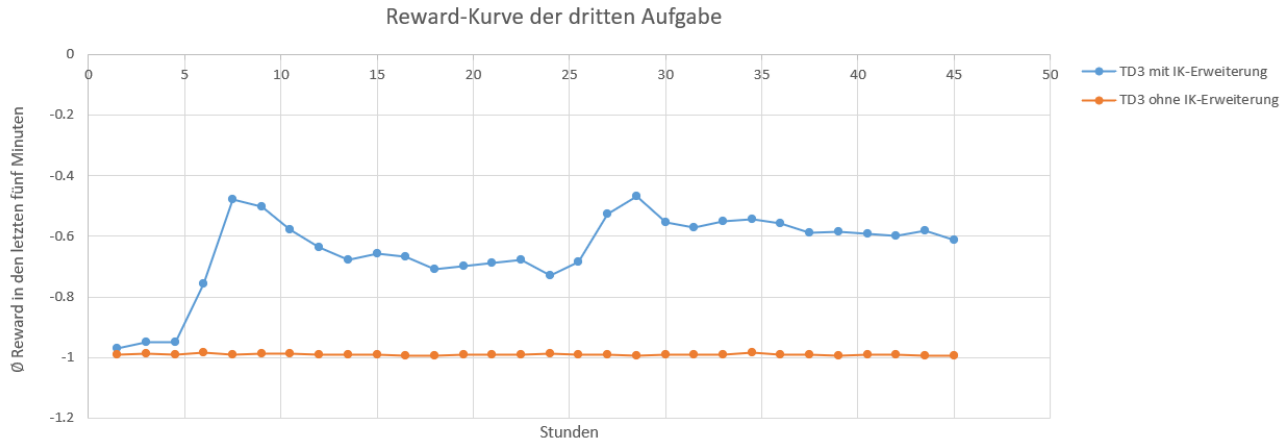


Figure 4.4: TD3 used in the robotic task described in Figure 2.1 with and without inverse kinematics; graph taken from [87]. The difference in step count between the base and IK-augmented variant is drastic in favour of TD3-IK, therefore the graph illustrates the agents performance in hours.

5

Introduction to Supervised Learning and Ensemble Strategies

As shown in Figure 1.1, machine learning encompasses a diverse range of paradigms aimed at enabling data-driven generation of models. While the primary focus of this thesis has been on reinforcement learning due to its applicability in developing advanced models for control, it is essential to acknowledge the complementary role of supervised learning (SL) in addressing specific challenges, particularly in machine perception and state identification. Supervised learning operates on labelled datasets, where the model learns to map input data to corresponding outputs based on provided examples. Through an iterative process, the model minimizes the discrepancy between predicted and actual outputs. In the SL paradigm, models adjust their parameters by comparing predicted outputs with ground-truth labels, reinforcing accurate predictions and correcting deviations. This chapter provides an introduction to the fundamental principles of SL and delves deeper into a methodology often employed in SL: Ensemble learning.

5.1 Supervised Learning

Supervised Learning is a foundational paradigm in machine learning, wherein models learn from labelled data (interpreted as supervisory signal, usually annotated by humans) to make predictions. Central to supervised learning is constructing a predictive model that can generalise well to unseen instances by learning patterns from labelled examples. In supervised learning, the overarching objective is to generate a statistical model reproducing a mapping that accurately predicts output labels from input features. A general SL task is to learn a function $f : \mathbb{R}^{\ell} \to \mathbb{R}^{m}$ to approximate an unknown

mapping $\Psi : \mathcal{X} \to \mathcal{Y}$ between the input space $\mathcal{X}$ and output space $\mathcal{Y}$. Typically, the sets $\mathcal{X}$ and $\mathcal{Y}$ are not known and we only have a finite set of matching observations $\{(\mathbf{x}_j, \mathbf{y}_j)\}_{j\in J} \subset \mathcal{X} \times \mathcal{Y}$ and J is its index set. The finite set of observations on which the model is trained is referred to as the training set. It must be stressed that in supervised learning, the potentially noisy labels $\hat{\mathbf{y}} = \mathbf{y} + \boldsymbol{\epsilon}$, are considered to be reasonably accurate $\frac{1}{J}\sum_{j\in J} |\mathbf{y_j} - \hat{\mathbf{y}}_\mathbf{j}| \approx 0$. Neural network-based models play a pivotal role in approximating high-generalization-capable functions, i.e. functions that can perform well for data outside of the training set. With a hindsight to Section 1.2.3, gradients calculated from a given loss defined by the labels and the model's predictions, iteratively adjust network parameters. The choice of the neural network architecture often depends on the complexity of the problem and available computational capabilities. Image-based SL often require CNNs for efficient learning whereas transformer networks are well suited for problems in which long-term dependencies of the data dominate [12].

5.2 Ensemble Learning

Inspired by the "Wisdom of Crowds" principle [94], ensemble learning seeks to form a function $\hat{f}$ by aggregating multiple weaker and diverse base learners (BLs) [95] to manage the bias-variance trade-off. When neural networks are employed as base learners, the approach is termed Deep Ensemble Learning (DEL) [96].

To formally define the convex ensemble model, consider a finite set of n base learners given by $\mathcal{H} = \left\{h_i : \mathbb{R}^\ell \to \mathbb{R} \mid i \in [n]\right\}$. The ensemble model prediction is given by

$$\hat{f}(\mathbf{x}) = \sum_{i\in[n]} \alpha_i h_i(\mathbf{x}), \tag{5.1}$$

where α_i are weights of the individual BLs. In ensemble learning, the weights α are usually restricted to the standard simplex:

$$\Delta_n = \left\{ \alpha \in \mathbb{R}^n \;\middle|\; \sum_{i=1}^{n} \alpha_i = 1,\ \alpha_i \geq 0,\ i \in [n] \right\}. \tag{5.2}$$

An important property and motivation of ensemble learning is that the final ensemble model is not restricted to the model space of the base learners, *i.e.,* the combination of base learners spans a larger model space. Thus, the base learners can be restricted to a far simpler model space than the final ensemble model. This property is further described in the following proposition.

Proposition 5.1. Given a set of base learners $h_1, \ldots, h_n \in \mathcal{F}$, the ensemble model $\hat{f} := \sum_{i=1}^{n} \alpha_i h_i$ lies within the convex hull of $\mathcal{F}$, but it does not necessarily belong to the model space $\mathcal{F}$.

Proof. Let $\mathcal{F}$ be defined as the model space of neural networks with a single ReLU node, *i.e.*, piecewise linear functions with one constant and one affine part. Consider a convex combination f of two base learners from $\mathcal{F}$. Then f is an ensemble model that is a piecewise linear function with one constant part and two affine parts and hence not an element of $\mathcal{F}$. □

5.2.1 Bootstrap Aggregating

Bootstrap aggregating, commonly referred to as bagging, is a widely used variance reduction technique in ensemble learning. It leverages multiple base learners trained on different subsets of the original training dataset to enhance predictive performance and robustness. By mitigating variance, bagging is particularly effective when applied to well-performing base learners or scenarios where data is sparse in specific regions of the training data manifold. The subsets used to train the base learners are generated through bootstrapping, a process that involves sampling with replacement from the original training set. These subsets maintain the same size as the original dataset but typically contain duplicate samples. As a result, each subset represents approximately 63.2% of the unique training data, as shown in [97]. A predefined number of n base learners are then independently trained on these bootstrapped subsets. The final ensemble prediction is obtained by averaging their outputs, assigning an equal weight to each learner: The weight of $h_i(\mathbf{x})$ in Equation (5.1) becomes $\alpha_i = \frac{1}{n} \quad \forall i \in [n]$. Bagging can be computationally expensive, as the base learners must be reasonably performative, requiring relative complex base learners. In summary, bagging is a powerful technique for reducing variance and improving the generalization capabilities of predictive models, particularly in cases where base learners are sufficiently performant and computational constraints are manageable.

5.2.2 Boosting

Boosting is an ensemble learning technique designed to reduce bias by sequentially training base learners. The key idea is that each subsequent model "boosts" the ensemble's overall performance by focusing on and correcting the mistakes made by its predecessors. In each iteration, the training data is re-weighted to reflect the model's

accuracy on individual samples. Misclassified or poorly predicted examples are assigned higher weights, incentivizing subsequent models to focus on these challenging data points. This iterative re-weighting mechanism enables boosting algorithms to progressively refine their predictive performance. Once a sufficient number of base learners have been trained, their predictions are aggregated using Equation (5.1). Base learners that demonstrate higher accuracy, particularly on difficult examples, are assigned greater model weights α_i. The base learners employed in boosting are generally weak and computationally simple. Some boosting approaches use decision stumps as base learners. Prominent examples of boosting algorithms include AdaBoost [98, 99], Gradient Boosting Machines [100] and Integer Programming Boost [101]. Prior to neural network-based machine learning, these algorithms have been used across diverse domains due to their ability to generate a complex predictor from simple base learners.

5.2.3 Voting

Voting in ensemble learning refers to an approach in which independently trained base learners collaboratively contribute to the final prediction. This methodology is predominantly employed in classification tasks, where each base learner casts a "vote" for a specific class label. Since base learners are trained on the same training data set, the effectiveness of the voting mechanism hinges on the diversity of these learners. Diversity in prediction can be achieved through techniques like random initialization in stochastic models, see Section 2.1.5.4. Without sufficient diversity, the ensemble risks become biased towards shared weaknesses among the base learners. Two principal voting schemes are commonly used: hard voting and soft voting. In the hard voting scheme, each base learner outputs a discrete class prediction, and the final ensemble prediction corresponds to the majority class label among the votes. This approach assumes that each learner contributes equally to the final decision. In contrast, the soft voting scheme aggregates the predicted probability distributions from all base learners. The class with the highest aggregated probability is selected as the final prediction. Soft voting often leverages the confidence levels of individual learners, leading to more nuanced and potentially more accurate ensemble predictions.

5.3 Relevance for Robotics

Perception is a cornerstone of effective robotic control, enabling robots to interpret and respond to highly dynamic environments. Perception involves reliably identifying the robot's state space (see Chapter 2), including detecting target objects, obstacles,

agent states, poses, and potential collisions. This capability is essential for decision-making processes and fundamental to control-generating algorithms described in earlier chapters.

Robotic perception relies on computer vision and "artificial tactile systems". Computer vision allows robots to process and understand visual data. Vision modules can be independently trained with on annotated datasets to identify key elements of the environment, such as targets, obstacles, or paths, and even perform tasks like depth estimation from standard visual sensors. Similarly, tactile systems enable robots to sense and interpret physical interactions in the environment. Supervised learning approaches can help these systems differentiate between desirable physical contact, such as pushing a swing door, and unintended collisions. It will be shown in Chapter 6 that ensemble models are generally more performative than their simpler counterparts. This improvement, however, comes with a significant computational overhead, as ensembles are complex structures. In real-world robotic applications, state-space identification and environmental interpretation must occur in real-time to ensure short response time. Balancing the accuracy benefits of ensemble learning with the computational constraints of real-time inference remains a critical challenge in deploying these models effectively on robotic hardware.

Beyond perception, supervised learning can play a role in robotic control, particularly through imitation learning, commonly referred to as behavioural cloning [102]. In this paradigm, the control problem is initially framed as a supervised learning problem, where the robot learns to imitate expert demonstrations. This approach provides a "warm start" for reinforcement learning algorithms, allowing them to begin training from an informed initial policy rather than random exploration.

The integration of supervised and ensemble learning strategies into robotic perception and control pipelines represents a step toward more autonomous and reliable systems. Vision and tactile systems, trained with supervised learning, can deliver robust sensory information representing the observation space in reinforcement learning algorithms. The ongoing challenge lies in bridging the gap between computational complexity and real-time deployment to enable robot operations in highly dynamic environments.

6 Optimizing Ensemble Models via Column Generation

This chapter presents a novel boosting algorithm that integrates principles from column generation, a well-established technique in classical optimization, with neural-network-based supervised learning. The proposed approach, named data-reduction ensemble learning algorithm (DDA), optimizes an ensemble of models by solving a linear programming (LP) problem defined by the predictions on the training dataset.

The algorithm constructs an optimal convex combination from a set of base learners, leveraging the concept of a soft margin commonly employed in support vector machines. Through this optimization, the ensemble aims to maximize predictive performance on the training data while establishing and maintaining generalization capabilities. The set of base learners is iteratively extended. The generation of new base learners is guided by the ensemble's performance on specific training data points. Neural network approximators and supervised learning techniques are employed to fit new and diverse base learners to correct the ensembles previous mistakes. This work attempts to bridge a methodological gap between traditional optimization techniques and modern neural network-based learning frameworks.

6.1 Introduction

In recent years, there has been a growing interest among researchers in bridging the fields of mathematical optimization and machine learning by applying optimization methodologies to machine learning problems. A particularly promising avenue of exploration lies in ensemble learning—a widely adopted approach in the machine learning community at the time of writing [103, 96, 104, 105]. Motivated by the structural parallels between ensemble learning and Column Generation (CG), the authors have

recently focused on investigating ensemble methods in the context of "classical" optimization. Ensemble learning methods combine multiple models to enhance predictive performance. There are plenty of real-world applications of ensemble learning such as object identification [106, 107, 108, 109], fault detection and diagnosis [110, 111, 112], semantic segmentation, edge detection [113, 114, 110], and for black box optimization [115].

It is known that deep learning networks may require vast computational effort to perform well in complex tasks. For example, state-of-the-art large language models can have more than 100 billion parameters [116]. From that point of view, combining easier or smaller learners may be an attractive approach. Ensemble learning shares similarities with the concept of Column Generation (CG), see [117, 105]. The latter computes solution estimates by combining partial solution candidates (options) in master optimization problems, which provide information to generate more options (columns). Efficient CG algorithms for solving nonlinear optimization problems, like in [118], are based on fast generation of initial columns in a starting phase and solving low-dimensional (pricing) sub-problems for generating columns. Ensemble models typically combine simpler base learners to obtain a more accurate and complex model. [119] show that the generalization error of an ensemble model strongly depends on the so-called diversity, see Section 6.2.4. Our research focuses on investigating whether base learners can be generated in a manner analogous to Column Generation (CG), utilizing subsets of the data rather than the entire training set. This approach aims to enhance both the diversity and generalizability of the resulting ensemble model while simultaneously reducing the computational overhead associated with processing the full training dataset for each new base learner The main research question in this chapter is how to efficiently decompose the optimization problem of training an ensemble model for classification problems and understanding the effect of dynamic data selection. One of the goals of this work has been to develop a decomposition framework that splits the full optimization problem into smaller sub-problems with optimality and convergence guarantees. This chapter demonstrates that such a decomposition approach exists under these assumptions, attempting to improve understanding in using global optimization techniques for determining globally optimal ensemble models.

To show this, a specific ensemble learning algorithm is designed which improves an initial model of unspecified complexity by generating and combining new base learners into an ensemble. The base learners and the initial model are combined into an ensemble by solving a linear programming (LP) master problem, which maximizes the soft margin of the ensemble classifier. The resulting dual solutions of the master problem are further

used in sub-problems, which train new base learners to boost the previous ensemble. Several types of sub-problems are considered for training base learners in the algorithm. To illustrate the efficiency of the ensemble algorithm, we measure the diversity and the empirical generalization error on several test instances. The main contribution of this chapter is summarized as follows:

- A novel method is designed for (binary) classification tasks, characterized by the following key properties:
  - Enabling a dual decomposition that splits the classification problem into sub-problems considering small base learners and a sub-set of the data points. In this way, the learning task is decomposed into simpler sub-problems with fewer variables.
  - The method is a framework for decomposing certain classification problems, that is globally convergent if sub-problems are solved to global optimality.
  - The data sub-sets defining the sub-problems are computed using sparse dual solutions of the LPBoost master problem, see [117]. The choice of the number of data points in the set defines a trade-off between accuracy and computational complexity.
  - Sub-problems are learning problems with either linear or nonlinear loss.
  - The base learners generated by the method are diverse.
- We show the method can reliably improve an existing (ensemble) classifier.
- By analysing the validation data we found that the generalization error decreases as diversity increases, thus supporting the use of model diversity as a performance measure.
- The proposed algorithm computes an optimal ensemble model within the convex hull of a given model space. Our results indicate that the ensemble model has the potential to achieve accuracy surpassing that of individual base learners. Notably, the ensemble convex hull constitutes a richer model space, as the base learner model space forms a proper subset of it. Additionally, we present a general theoretical result characterizing the ensemble model space.

This chapter is structured as follows. Section 6.2 focuses on LPBoost-based ensemble learning and the mechanisms involved in the algorithm, namely data selection, diversity and ensemble learning. Section 6.3 describes the algorithm in a modular fashion; first it provides an overview over the general scheme and then describes the core features of the algorithm. Section 6.4 presents experiments with standard LPBoost and the new variant for two data sets with varying difficulties. Section 6.5 summarizes our findings.

6.2 Diverse Ensemble Learning and LPBoost

We provide background on the problem formulation, LPBoost, and some fundamental theory. We first sketch the problem formulation and the ensemble learning concept in Section 6.2.1. Then, we describe the LP master problem of LPBoost and the pricing problem in Section 6.2.2. We give an intuition for the potential of only considering a sub-set of data points in the generation of base learners by exploiting the sparsity in Section 6.2.3. Section 6.2.4 describes the role of model diversity based on considerations from literature.

6.2.1 Problem formulation and ensemble learning (EL)

In this chapter, a binary classification problem is considered with $\mathcal{Y} = \{-1, 1\}$. For one single arbitrary observation (x, y), y denotes the target of this observation with $y \in \{-1, 1\}$. The binary classifier $f : \mathbb{R}^\ell \to \{-1, 1\}$ as $f(x) = \text{sign}(\hat{f}(x))$ is defined by the score function $\hat{f} : \mathbb{R}^\ell \to \mathbb{R}, \hat{f} \in \mathcal{F}$. Given a model space $\mathcal{F}$, the learning task aims at obtaining a predictive score model $\hat{f}^* \in \mathcal{F}$ which minimizes the expectation of a loss function $\mathcal{L}$

$$\hat{f}^* = \text{argmin} \left\{ \mathbb{E}[\mathcal{L}(\hat{f}(x), y)] : \hat{f} \in \mathcal{F} \right\}, \tag{6.1}$$

where $\mathbb{E}$ denotes the expectation regarding a joint probability distribution $P(x, y)$ over $\mathcal{X} \times \mathcal{Y}$. As we are typically restricted to a finite training set $T = \{(\mathbf{x}_j, \mathbf{y}_j)\}_{j \in J}$, the learning task is usually approximated by

$$\hat{f}^* = \text{argmin} \left\{ \sum_{j \in J} \mathcal{L}(\hat{f}(x_j), y_j) : \hat{f} \in \mathcal{F} \right\}. \tag{6.2}$$

Throughout the chapter, we assume that the model space $\mathcal{F}$ is limited to a predefined ML architecture, *e.g.,* a ANN with a specific number of nodes and layers. In binary classification, a loss function could be for instance a linear classification error (6.18) or a non-linear binary cross entropy loss (6.17). Considering a training set $T := \{(\mathbf{x}_j, y_j)\}_{j \in J}$, the learning task in ensemble learning is two-fold: learn the ensemble weights α and construct the set of base learners $\mathcal{H}$. Consider for instance deep neural networks with a given architecture. The base learners h_i are thus parameterized by the weights and biases of the ANN. To simplify the notation, we define

$$h_i(\mathbf{x}) := \varphi(\mathbf{x}; \mathbf{W}_i), \tag{6.3}$$

where φ denotes the input-output mapping of a ANN with weights and biases given by a parameter vector $\mathbf{W}_i \in \mathbb{R}^k$. We are assuming that all the weights and biases are

bounded, *i.e.*, $\|\mathbf{W}_i\|_1 \leq M < \infty$. The space of base learners is then reduced to the Hilbert space $\mathcal{F} = \{\varphi(\mathbf{x}; \mathbf{W}) \mid \mathbf{W} \in [-M,\ M]^k\}$. Furthermore, by limiting the number of BLs in the ensemble model to n, the ensemble learning problem can be written as

$$\begin{aligned} \min_{\alpha, h_1, \ldots, h_n} \quad & \sum_{j \in J} \mathcal{L}\left(\sum_{i \in [n]} \alpha_i h_i(\mathbf{x}_j), y_j\right) \\ \text{s.t.} \quad & \alpha \in \Delta_n, \\ & h_i \in \mathcal{F}, \quad i \in [n], \end{aligned} \tag{6.4}$$

where Δ_n denotes the standard n-dimensional simplex. The minimizer of (6.4) is an optimal ensemble model for the given training set T, loss function $\mathcal{L}$, base learner architecture, and number of base learners. By Proposition 2.1, we know that the ensemble model given by the minimizer of problem (6.4) is within the convex hull of $\mathcal{F}$, denoted $\text{conv}(\mathcal{F})$. With an additional assumption on the number of base learners we get a more general optimality property for the ensemble model given by problem (6.4), as described in the following proposition.

Proposition 6.1. Given that the number of base learners in problem (6.4) is sufficiently large, then the resulting ensemble model will also be optimal within $\text{conv}(\mathcal{F})$ for the given objective function.

Proof. Follows from Carathéodory's theorem [120]. □

Propositions 5.1 and 6.1 are significant because they establish that the ensemble model, derived by solving (6.4), is optimal within a broader and more complex model space compared to the space of the base learners. This insight serves as a primary motivation for designing an algorithm that iteratively optimizes over the comparatively simpler base learners to effectively address (6.4) and construct a robust ensemble model.

Each base learner h_i in problem (6.4) is parameterized by k parameters. The complete problem contains $k \times n + n$ variables. In practice, however, a substantial number of auxiliary variables may be required to represent the activation states of the base learners for each data point, significantly increasing the dimensionality of the optimization problem. Even when disregarding this large variable count, problem (6.4) remains inherently challenging due to the non-convexity, as formally described in the following proposition.

Proposition 6.2. Unless $\mathcal{F}$ is a singleton set, problem (6.4) is in general non-convex.

Proof. With $\mathcal{F}$ not restricted to a singleton set, the terms $\alpha_i h_i(\mathbf{x}_j)$ in the objective results in bilinear terms as function h_i is parameterized by weights and biases. □

Considering that non-trivial classification problems typically require DNNs with thousands to millions of parameters, problem (6.4) still remains well out of reach of today's general purpose deterministic global optimization solvers. To solve problem (6.4), we present an algorithm inspired by the LPBoost algorithm [117] that follows a column generation approach. By this decomposition approach, we are able to determine one base learner, *i.e.,* column, at a time and thus limit each sub-problem to a smaller sub-set of variables. The presented algorithm also allows to only consider a sub-set of the training data when determining a new base learner, further simplifying each sub-problem.

6.2.2 LPBoost and its LP master problem

To determine an optimal convex combination of a given finite set of fixed base learners, the LPBoost algorithm of [117] considers an LP optimization problem which maximizes the soft margin of the combined base learners in a boosting approach. They consider a given set of base learners $\mathcal{H} = \{h_i | i \in [n]\}$. The LP model determines an ensemble model $f(x) = \sum_{i \in I} a_i h_i(x)$ that maximizes the soft margin of training data set $\{X, Y\}$:

$$\begin{aligned} \max_{\alpha,\xi,\rho} \quad & \rho - \lambda \sum_{j \in J} \xi_j \\ \text{s.t.} \quad & y_j \sum_{i \in [n]} \alpha_i h_i(\mathbf{x}_j) + \xi_j \geq \rho, \quad j \in J \\ & \alpha \in \Delta_n \\ & \xi_j \geq 0, \qquad\qquad j \in J \\ & \rho \in \mathbb{R} \end{aligned} \tag{6.5}$$

where $y_j \sum_{i \in [n]} \alpha_i h_i(\mathbf{x}_j)$ is the margin of data point (x_j, y_j) for a finite set of base learners. Variable ξ_j is the margin slack variable of $(x_j, y_j) \in \mathcal{X} \times \{-1, 1\}$ and λ a positive penalty parameter, *e.g.,* $\lambda = 0.1$ which influences the size of the non-zero dual

variables. A dual problem of (6.5) is given by

$$\begin{aligned} \min_{\mathbf{u},\beta} \quad & \beta \\ \text{s.t.} \quad & \sum_{j\in J} u_j y_j h(\mathbf{x}_j) \leq \beta, \quad h \in \mathcal{H} \\ & \sum_{j\in J} u_j = 1 \\ & 0 \leq u_j \leq \lambda, \qquad\qquad j \in J \end{aligned} \tag{6.6}$$

In this formulation, u represents a dual solution of master problem (6.5), where u_j represents the miss-classification cost assigned to data point (x_j, y_j). Following the column generation approach in [117], a new base learner $\hat{h}$ can be generated or selected by solving the following pricing problem, which in the machine learning context represents the BL generation

$$\hat{h} = \arg\max_{h\in\mathcal{N}} \sum_{j\in J} u_j y_j h(x_j), \tag{6.7}$$

where $\mathcal{N} \subset \mathcal{F}$ is a large finite set of base learners and u is the information of the master problem in terms of a dual solution of (6.5). Leveraging column generation to solve problem (6.5) provides a computationally efficient framework for managing a potentially vast set of base learners $\mathcal{N}$ by iteratively operating on a smaller, more tractable subset $\mathcal{H}$. The objective (6.7) can be used as a termination criterion of the boosting method, see [117]. Note that the method in [117] is limited to a large set of predefined base learners. Our approach extends this framework by also considering the generation of new base learners. This is done by solving the less restrictive pricing problem, similar to the approach in [121]

$$\hat{h} = \arg\max_{h\in\mathcal{F}} \sum_{j\in J} u_j y_j h(x_j). \tag{6.8}$$

Since this method does not rely on a limited set of initial base learners, it is more flexible and potentially yields better solutions.

6.2.3 Sparse dual solution, dynamic data selection and BL generation

Numerical experiments show that the dual master problem (6.6) solution u is typically a sparse vector, with most of its components taking values of either 0 or λ. In linear programming, a sparse vector means that most of its entries are equal to 0. Let

$$R := \{(x_j, y_j) \in T : u_j > 0\} \tag{6.9}$$

be the set of data points with a non-zero dual value, which we refer to as active data points. Note, that a dual value of $u_j = 0$ indicates that the associated data point is well-classified with a sufficient margin, as shown in [117]. Since $u \in \Delta_{|J|}$ and $u_j = \lambda$ if $u_j > 0$, the number of non-zero components of u is $|R| = \frac{1}{\lambda}$ for $\frac{1}{\lambda} \in \mathbb{N}$. The relative number of active data points is $\nu := \frac{1}{\lambda|T|}$, such that $|R| = \nu|T|$. Using ν as a parameter to control the number of active points, we can set $\lambda = \frac{1}{\nu T}$ in (6.6) to influence the sparsity of the dual solution that allows us to identify critical data points and discard the irrelevant data points.

For the non-active data points in $T \backslash R$, the objective function contribution coefficient $u_j = 0$ in BL generation sub-problem (6.8) is zero, such that the pricing problem (6.8) is equivalent to

$$\hat{h} = \arg\max_{h \in \mathcal{F}} \sum_{(x_j, y_j) \in R} u_j y_j h(x_j). \tag{6.10}$$

This means that we can limit the generation of new learners to the active data points in R, which tend to be miss-classified or well-classified with insufficient margin. Numerical experiments showed that, in practice, it is more effective to use a nonlinear objective function in the BL generation problem

$$\hat{h} = \arg\max_{h \in \mathcal{F}} \sum_{(x,y) \in R} \mathcal{L}(h(x), y) \tag{6.11}$$

rather than the traditional linear objective in (6.10). It is noteworthy that sub-problems in learning tasks are generally not solved to global optimality, attributable to their complexity. Note that since data set R is changing in each iteration, the related base learners tend to be more diverse which is an additional advantage of working with a smaller set of active points. Section 6.2.4 summarizes some results of [119], showing that the generalization error of an ensemble model is related to the diversity of its base learners.

6.2.4 Diversity measurement for ensemble

An interesting approach for analysing an ensemble model is to compute the diversity of its base learners. [119] show that the generalization error of an ensemble depends mainly on the diversity and accuracy of the base learners. We will also use the diversity concept for analysing the investigated ensemble models. The following describes this approach. Consider an error function for binary classification based on $\text{margin}(f(x), y) = yf(x)$

defined as

$$\mathrm{Err}(f(x), y) := \begin{cases} 1 & \text{if margin}(f(x), y) = -1 \\ 0.5 & \text{if margin}(f(x), y) = 0 \\ 0 & \text{if margin}(f(x), y) = 1. \end{cases}$$

Hence, $\mathrm{Err}(f(x), y) = -\frac{1}{2}(yf(x) - 1)$. The generalization error of f is defined by

$$G := \mathbb{E}[\mathrm{Err}(f(x), y)].$$

The diversity of f at $x \in \mathcal{X}$ is defined by the difference between ensemble error and the weighted average error of individual classifiers (base learners):

$$\begin{aligned} \mathrm{div}(f(x), y) :=& \mathrm{Err}(f(x), y) - \sum_{i \in I} \alpha_i \mathrm{Err}(f_i(x), y) \\ =& \frac{1}{2}\mathrm{margin}(f(x), y) - \frac{1}{2}\sum_{i \in I} \alpha_i \mathrm{margin}(f_i(x), y). \end{aligned} \tag{6.12}$$

The relation between generalization error and diversity according to [119] is

$$G = \frac{1}{2}\left(1 - \mathbb{E}[\mathrm{sign}(\gamma(x))]\right),$$

where $\gamma(x) := \mathrm{sign}(\mathrm{div}(f(x), x)) - 2\mathrm{div}(f(x), x))$. This shows that increasing the diversity at points where $\gamma(x)$ is small, will decrease G. In particular, the defined diversity measure and G varies in different ranges of diversity

- When $\mathrm{div}(f(x), y)$ is negative, the ensemble classifier has miss-classified instance x. Increasing diversity will improve the ensemble performance.
- When $\mathrm{div}(f(x), y)$ is positive, diversity needs to be decreased to increase the generalization performance of the ensemble.

More details of derivations for diversity and generalization errors can be found in [119]. Note that the ensemble model in [119] combines the classifier output of base learners $f_i(x) \in \{-1, 1\}$ instead of score function $h_i(x)$, which is different from the ensemble described in Section 6.2. We show in Section 6.4 experimentally that for base learners obtained from solving sub-problem (6.11), the generalization error is decreasing and the average diversity is increasing iteratively.

6.3 A Data-Reduction Ensemble Learning Algorithm (DDA)

The initialization procedure, designed to generate an initial set of base learners, accelerates the computation process. The core of the algorithm involves solving a sequence

of master problems (6.6) to determine convex combinations of base learners, alongside solving sub-problems (6.10) or (6.11) to generate new base learners.

We present an ensemble learning algorithm, which iteratively generates base learners $h_i(x) \in [-1, 1]$. The algorithm defines low-dimensional BL generation sub-problems using dynamic data selection. The sub-problems can be solved using an arbitrary method. The initialization procedure, designed to generate an initial set of base learners, accelerates the computation process. The core of the algorithm involves solving a sequence of master problems (6.6) to determine convex combinations of base learners, alongside solving sub-problems (6.10) or (6.11) to generate new base learners. We show in Section 6.3.2 that if the sub-problems, *i.e.*, pricing problems, are solved exactly, then the algorithm converges to an optimal solution of problem (6.4), which is equivalent to problem (6.14).

6.3.1 Algorithm

Algorithm 6.1 describes the new ensemble learning algorithm. It consist of three phases, named initial, generation and refinement phase. The initial phase generates a base-line model h using procedure SOLVEINITIAL by fitting the training set, such that h initiates the base learner set $\mathcal{H}$. In the generation and refinement phases, diverse base learners are generated and added to base learner set $\mathcal{H}$. After adding a new base learner, the primal and dual solution (α, u) of LP master problem (6.5) are used to update ensemble model $\hat{f}$ by calling function SOLVEMPANDUPDATEENS. Employing the optional generation phase in conjunction with the refinement phase typically yields superior results compared to using the refinement phase alone. The refinement phase also extends $\mathcal{H}$, but the base learners generated in this phase are fitted on a sub-set of active training data determined by u.

In order to accelerate the CG procedure, in the beginning, columns are generated using a start heuristic disregarding the master problem. This is motivated by a potentially poor search direction if the master problem is defined by few columns, see [118]. Our ensemble algorithm employs a similar strategy, which we refer to as the generation phase. This phase is based on error-driven bootstrap aggregation [122] and is outlined in Algorithm 6.2. Error-based bootstrapping generates a new data set S of size $|S| = |T|$. Unlike conventional bootstrapping that forms data set S by random sampling from the original data set with replacement, error-based bootstrapping adds to the set of miss-classified data points $M \subset T$ a uniform sample of $T \setminus M$ with replacement, such that it focuses on miss-classified instances from a base learner h. This procedure is denoted by function BOOTSTRAPDATA(h, T) in Line 6 of Algorithm 6.1.

Algorithm 6.1 Main Algorithm, Dynamic Data-Reduction Ensemble Learning

Input: Training data $T = X \times Y \subset \mathcal{X} \times \{-1, 1\}$

Parameters: BL configuration and data reduction value $\nu \in (0, 1)$ defining $\lambda = \frac{1}{\nu|T|}$

Output: $f(\cdot)$

1: **function** DDA
2: $h \leftarrow$ SOLVEINITIAL(T) `# initial phase`
3: $\mathcal{H} \leftarrow \{h\}$, $R \leftarrow \emptyset$
4: **repeat** `# generation phase (optional)`
5: $\hat{R} = R$
6: $S \leftarrow$ BOOTSTRAPDATA(h, T)
7: $h \leftarrow$SOLVEBTSUBPROBLEM(S)
8: $\mathcal{H} \leftarrow \mathcal{H} \cup \{h\}$
9: $(\hat{f}, u) \leftarrow$ SOLVEMPANDUPDATEENS($\mathcal{H}, T$)
10: $R \leftarrow$ ACTIVEDATA(T, u)
11: **until** $|R \cap \hat{R}| > \sigma|R|$
12: **repeat** `# refinement phase`
13: $R \leftarrow$ ACTIVEDATA(T, u)
14: $h \leftarrow$SOLVEREFSUBPROBLEM(R)
15: $\mathcal{H} \leftarrow \mathcal{H} \cup \{h\}$
16: $(\hat{f}, u) \leftarrow$ SOLVEMPANDUPDATEENS($\mathcal{H}, T$)
17: **until** stopping criterion
18: **return** sign($\hat{f}$)

The miss-classified set is obtained according to the predictions of h instead of preliminary ensemble $\hat{f}$. If instead the predictions of $\hat{f}$ are used, *i.e.,* BOOTSTRAPDATA($\hat{f}, T$) is called (see Algorithm 6.2), the generation phase might not terminate quickly and transition to the subsequent refinement phase takes longer. The set of miss-classified data points is updated in each iteration. In Lines 7 and 8 of Algorithm 6.1, a new base learner $h_i(x)$ is fitted on S by SOLVEBTSUBPROBLEM(S) and added to set $\mathcal{H}$. SOLVEMPANDUPDATEENS($\mathcal{H}$, T) in Line 9 of Algorithm 6.1 constructs a new master problem (6.6) given the updated base learner inferences and labels. The master problem is solved by an LP solver, and it returns the primal and dual solution. The ensemble score function $\hat{f}$ is obtained via the primal solution which contains the base learner weight vector.

The details of SOLVEMPANDUPDATEENS can be found in Algorithm 6.3. The

Algorithm 6.2 Generates error-based bootstrapping data set S of size $|S| = |T|$

```
1: function BootstrapData(h, T)
2:     M ← missClassifiedData(h, T)
3:     S ← Uniform sample of |T| − |M| data points from T
4:     S ← S ∪ M
5:     return S
```

sparse dual solution u is used to determine the set of active data (6.9). Function ActiveData(T, u) in Line 10 of Algorithm 6.1 selects active data from T fulfilling (6.9) and returns set R.

Algorithm 6.3 Primal/dual solution of master-problem and update ensemble

```
1: function solveMPandUpdateEns(H, T)
2:     (α, u) ← primalDualSolveMP(H, T, λ)
3:     f̂ = Σ_{h_i ∈ H} α_i h_i
4:     return (f̂, u)
```

The generation phase stops if the difference between former active data set $\hat{R}$ and the current active set R is not significant. The stopping criterion parameter σ determines the minimum change in the active data set to continue the iterations in the generation phase. It is recommended to set its value to at least 0.2, indicating a change of 20%. Note that a small value of σ can result in the generation of many base learners that do not contribute to the ensemble. The refinement phase of Algorithm 6.1 augments the set $\langle$ by adding base learners $h_i(x)$. It achieves this by solving the LP master problem, assigning weights to data points and training base learners on sufficiently weighted points. The function ActiveData() is called to determine the training set R, which is then used by solveRefSubProblem() to solve either (6.11) or (6.10), depending on whether a linear or nonlinear objective is used. Since data points with large weights are selected, Parameter ν is used in Algorithm 6.3 to calculate λ in (6.5) adjusting the sparsity of the dual solution to provide a limit on the number of considered data points. We discuss the choice of its value in Section 6.4. The termination criterion of the refinement phase in Algorithm 6.1 can be based on the dual feasibility criterion from the original LPBoost algorithm, see [117]. If the following condition is met, the base learner can not further improve the ensemble:

$$\sum_{j \in J} u_j y_j h_i(x_j) \leq \beta + \epsilon, \tag{6.13}$$

where ϵ is a small positive value. There are two main advantages of the column generation approach in Algorithm 6.3 for solving (6.4). First, it decomposes the problem into simpler sub-problems compared to simultaneously optimizing all base learners and their combination. Secondly, it allows us to only consider a small sub-set of active data points which results in easier sub-problems for determining new base learners, *i.e.,* easier pricing problems.

6.3.2 Convergence

We analyze the convergence properties of Algorithm 6.1 and focus on the soft margin loss function, although the algorithm is not limited to this specific loss function. We show that Algorithm 6.1 computes an optimal solution of the extended master problem

$$\begin{aligned} \max_{f,\xi,\rho} \quad & \rho - \lambda \sum_{j \in J} \xi_j \\ \text{s.t.} \quad & y_j f(\mathbf{x}_j) + \xi_j \geq \rho, && j \in J \\ & \xi_j \geq 0, && j \in J \\ & \rho \in \mathbb{R}, \quad f \in \operatorname{conv}(\mathcal{F}). \end{aligned} \tag{6.14}$$

Note that problem (6.14) is equivalent to problem (6.4) when using the soft margin loss function and allowing sufficiently many base learners. The proof requires a few intermediate results that are presented in Propositions 6.3, 6.4 and 6.5.

Proposition 6.3. Problem (6.14) is convex.

Proof. Consider the feasible set of (6.14) as

$$\Omega := \{(f, \xi, \rho) : y_j f(\mathbf{x}_j) + \xi_j \geq \rho,\ j \in J,\ \rho \in \mathbb{R},\ f \in \operatorname{conv}(\mathcal{F})\}.$$

Let $g_1 = (f_1, \xi_1, \rho_1)$, $g_2 = (f_2, \xi_2, \rho_2)$ and $g = (f, \xi, \rho) := tg_1 + (1-t)g_2$, $t \in [0, 1]$. From $g_1, g_2 \in \Omega$ we have

$$y_j f(\mathbf{x}_j) + \xi_j = y_j(tf_1(\mathbf{x}_j) + (1-t)f_2(\mathbf{x}_j)) + t\xi_{1j} + (1-t)\xi_{2j} \geq t\rho_1 + (1-t)\rho_2 = \rho.$$

Since $f = tf_1 + (1-t)f_2 \in \operatorname{conv}(\mathcal{F})$, $g \in \Omega$, which proves Ω is convex. □

Proposition 6.4. Problem (6.14) is equivalent to the semi-infinite dual-LP:

$$\begin{aligned} \min_{\mathbf{u},\beta} \quad & \beta \\ \text{s.t.} \quad & \sum_{j \in J} u_j y_j h(\mathbf{x}_j) \leq \beta, \quad h \in \mathcal{F} \\ & \sum_{j \in J} u_j = 1 \\ & 0 \leq u_j \leq \lambda, && j \in J. \end{aligned} \tag{6.15}$$

Proof. Consider the Lagrangian of (6.5)

$$c(f,\rho,\xi,\mu) := \rho - \lambda \sum_{j\in J} \xi_j + \sum_{j\in J} u_j(y_j f(\mathbf{x}_j) + \xi_j - \rho).$$

Then

$$\begin{aligned} c^*(u) :=& \max\left\{c(f,\rho,\xi,u) : (f,\xi,\rho) \in \operatorname{conv}(\mathcal{F}) \times \mathbb{R}_+^{|J|} \times \mathbb{R}\right\} \\ =& \max_{\rho\in\mathbb{R}}(1 - \sum_{j\in J} u_j)\rho + \max_{\xi\in\mathbb{R}_+^{|J|}} \sum_{j\in J}(u_j - \lambda)\xi_j + \max_{f\in\operatorname{conv}(\mathcal{F})} \sum_{j\in J} u_j y_j f(\mathbf{x}_j) \\ =& \max_{f\in\operatorname{conv}(\mathcal{F})} \sum_{j\in J} u_j y_j f(\mathbf{x}_j) = \max_{h\in\mathcal{F}} \sum_{j\in J} u_j y_j h(\mathbf{x}_j). \end{aligned} \tag{6.16}$$

Since problem (6.14) is convex, strong duality applies, *i.e.,* the optimal value of (6.14) is equal to

$$c^* := \min\left\{c^*(u) : u \in \mathbb{R}_+^{|J|}\right\} = \min\left\{\beta : \beta \geq c^*(u),\, u \in \mathbb{R}_+^{|J|}\right\}.$$

For dual points with $c^*(u) < \infty$, we have $\sum_{j\in J} u_j = 1$ and $0 \leq u_j \leq \lambda$, $j \in J$. From (6.16) follows

$$c^* = \min\left\{\beta : \beta \geq \max_{h\in\mathcal{F}} \sum_{j\in J} u_j y_j h(\mathbf{x}_j),\, \sum_{j\in J} u_j = 1,\, 0 \leq u_j \leq \lambda,\, j \in J\right\}.$$

Hence, the optimal value of (6.15) is c^*. □

Proposition 6.5. If the loop in the refinement phase of Algorithm 6.1 is not stopped, and pricing problem (6.8) is solved exactly, the primal-dual solution $(\hat{f}, u)$ converges towards a primal-dual solution of extended master problem (6.14).

Proof. Let U denote the set of dual points u computed by Algorithm 6.1. Since U is bounded, there exists a subsequence $\{u\} \subset U$ converging to a point $\hat{u}$, where u is a dual solution of (6.6). Let $\mathcal{H}$ be the set of base learners related to the subsequence. Then $\hat{u}$ solves (6.6) regarding $\mathcal{H}$. Let $\hat{h}$ denote the solution of pricing problem (6.8) at $\hat{u}$. Since $\hat{u}$ is a limit point and by continuity of the dual function, we have

$$\max_{h\in\mathcal{H}} \sum_{j\in J} \hat{u}_j y_j h(\mathbf{x}_j) = \sum_{j\in J} \hat{u}_j y_j \hat{h}(\mathbf{x}_j) = \lim_{u\to\hat{u}} \max_{h\in\mathcal{F}} \sum_{j\in J} u_j y_j h(\mathbf{x}_j).$$

Hence, we can replace $\mathcal{H}$ by $\mathcal{F}$ in (6.6), which shows that $\hat{u}$ is a solution of (6.15). Since (6.15) is the dual of (6.14), the proposition follows. □

The theoretical results show that Algorithm 6.1 can generate an optimal solution of problem (6.4). Keep in mind that this requires the BL generation sub-problem to be solved exactly. Even if the dimension of a sub-problem is much smaller than of the original problem, it may still not be feasible to solve the problem exactly. The results show that the algorithm follows a solid decomposition technique and if the sub-problems are solved approximately then the algorithm becomes more heuristic.

6.4 Numerical Experiments

We study the characteristics of Algorithm 6.1 on a set of binary classification problems (6.1). Algorithm 6.1 was implemented in Python, as a part of the general ensemble framework DECOLEARN [1] which contains learning methods to generate four types of initial models and base learners with any configuration in the generation and refinement phases: MLP, CNNs, CART-based decision trees and XGBoost models [123, 124, 125]. The experiment focuses on convolutional neural networks (CNN) base learners. The pricing sub-problems (6.10) or (6.11) which yields new base learners, is solved using TensorFlow 2.3. This approach does not guarantee global optimality for the pricing sub-problems, but allows to efficiently work with non-trivial base learners. Solving the pricing sub-problems to guaranteed global optimality in a reasonable time is still not feasible for CNN base learners of relevant size. DECOLEARN also uses GUROBI 9.5.1 for solving the LP master problem. All computational experiments were carried out on a computer with i7-1165G7 4-Core 2.80 GHz CPU and 16GB RAM.

The test instances for binary classification were obtained from the CIFAR-10 data set [126]. The data set contains 60,000 32×32 color images in 10 classes, with 6,000 images per class. There are 50,000 training images and 10,000 test images, divided into five training batches and one test batch, each with 10,000 images. The test batch contains 1,000 randomly-selected images from each class, while the training batches contain the remaining images in random order. However, some training batches may contain more images from one class than another, with exactly 5,000 images from each class in total. For our experiments on binary classification using CIFAR-10, we chose two pairs of labels with varying degrees of similarity. The labels of categories 3 and 8 are considered easy and labels 3 and 2 hard to distinguish.

For base learner generation with a non-linear loss, we used CNNs with VGG architectures [127] in both the generation and refinement phase based on the binary cross

[1] see `https://github.com/ouyang-w-19/decolearn`

entropy:

$$\mathcal{L}(h(x), y) = y \log(Q(h(x)) + \epsilon) + (1 - y) \log(1 - Q(h(x)) + \epsilon), \tag{6.17}$$

where the clipping is defined as $Q(y) := \max(\min(y, 1-\epsilon), \epsilon)$. The tolerance ϵ is added for stability reasons and is a small positive number, e.g. 10^{-7}. If the base learners have an output range of $[0, 1]$, the output is re-scaled to $[-1, 1]$ or alternatively to $\mathbb{R}$. Note that (6.17) requires a different binary label type for base learner training, namely $y \in \{0, 1\}$. The complexity of the models were adjusted to the degree of similarity between two labels. The details of the algorithm parameters, model types and hyper-parameters are documented in Appendix A, Tables 6.8 and 6.9.

Evaluating the algorithm involves several key performance indicators, including ensemble binary accuracy, diversity, and the solution to the master problem at each iteration. The diversity is defined per data point. To describe the overall diversity of an ensemble, we will use the average diversity over the training set defined by

$$\overline{\mathrm{div}}(f) := \frac{1}{|T|} \sum_{j \in T} \mathrm{div}(f(x_j), y_j).$$

We compare different variants of the algorithm, which differ in the types of sub-problems and parameter configurations. One of the variants solves sub-problems with active data only and another variant encompasses all training data when solving the "sub-problem". For the data selection based ensemble method, we use two variants. The linear variant (LL) applies the standard LPBoost with a sub-problem that minimizes the reduced cost (pricing problem) in the refinement phase with a linear expected loss function

$$\max_{h \in \mathcal{F}} \sum_{(x_j, y_j) \in T} u_j y_j h(x_j). \tag{6.18}$$

We compare that to a nonlinear variant called NL which considers sub-problems on the active set using the nonlinear loss function (6.17). We will first focus on the characteristics of the variants of DDA in the generation phase and then on their behaviours in the refinement phase.

6.4.1 Generation Phase

The generation phase is optional and serves the purpose of generating base learners through an error-based bootstrapping approach. The master problem is only solved to calculate an ensemble in each iteration and its corresponding active set R, providing a stopping criterion for this phase. Active columns in the master problem refer to the

columns with non-zero weights of the corresponding base learner. In the best case, the generated columns remain active in the refinement phase. This is illustrated when running the DDA generation phase on an instance of CIFAR-10, where after 2 iterations the phase ends and all generated base learners stay active, see Figure 6.1. The case in Fig. 6.2 illustrates a run over MNIST instances where four out of six BLs generated in the generation phase have zero weights in the next phase.

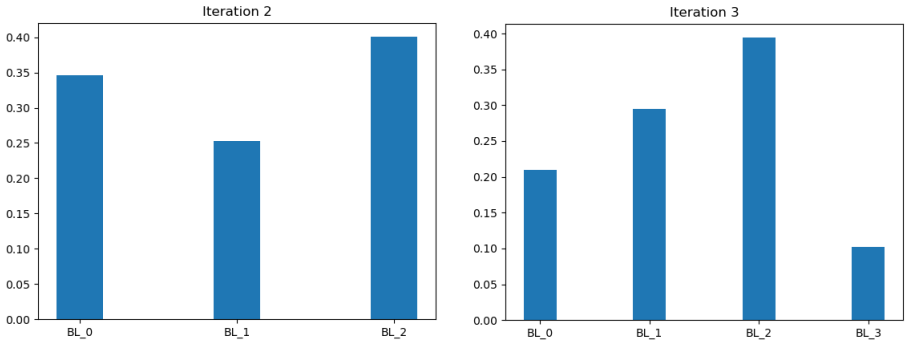


Figure 6.1: Generated base learners with their master problem weight. The generation phase ends after 2 iterations. In the next iteration, the old base learners remain active together with a new active column.

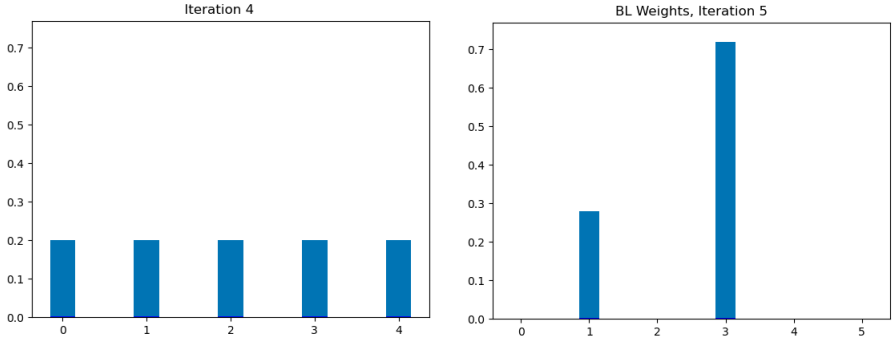


Figure 6.2: Six base learners (corresponding weights on the y-axis) are generated in the generation phase. As shown by the second figure, most of the columns become inactive in the next iteration.

We conducted experiments by varying the stopping criterion of the generation phase. Specifically, we compare a variant with a fixed number of 5 iterations to one that uses the criterion based on the relative progress described in Line 11 of Algorithm 6.1. The results, shown in Table 6.1, present the average performance over 20 runs of DDA,

one iteration following the generation phase, for both stopping criterion variants. The number of inactivate columns corresponds to the ensemble obtained from the generation phase. A relative criterion has a relatively smaller number of inactive columns than a fixed number of 5 iterations. The number of iterations for the relative criterion is 2.6 on average. The last row gives the number of miss-classified instances in the validation set of the ensemble after the generation. This illustrates that with more iterations, the result generalizes better. Specifically, continuing bootstrap aggregation for an average of twice as many iterations results in an ensemble that generalizes more effectively, with an improvement factor of approximately two.

Table 6.1: Effect of fixed versus relative stopping criterion on number of columns becoming inactive, number of iterations and miss-classified data in the validation set

Stopping criterion:	Fix 5 iterations	relative progress
Inactivated col. (frac.)	0.49 ± 0.15	0.29 ± 0.23
Iterations	5	2.6 ± 0.66
nr. Miss-classified train	2084 ± 949	2271 ± 1064
nr. Miss-classified val.	335 ± 184	649 ± 461

To study the contribution of the generation phase to DDA, we run the algorithm with and without the phase for an instance of CIFAR-10 with binary labels (categories 3 and 2). Ten runs of the algorithm are performed and Table 6.2 presents the obtained statistical results. We measured the classification quality for the final ensemble as the number of miss-classified data points of the training set and of the validation set. The results show the average values over ten runs with the standard deviation. Table 6.2 indicates that the generation phase can improve the quality of the final ensemble, that is, the number of miss-classified data points (testing/validation set) decrease from 723 to 576 on average. On the other hand, the average computational time of DDA is doubled when adding the generation phase.

The next section will illustrate that a big improvement (lower) in miss-classification score may occur immediately after the generation phase at a relatively low computational cost.

6.4.2 Refinement Phase

The refinement phase adds base learners to set $\mathcal{H}$. We used two data sets that vary in difficulty to compare the behavior for the variants LL and NL. Experiment A uses

Table 6.2: DDA with and without generation phase on categories 3 and 2 of the CIFAR-10 data set

DDA	without gen. phase	with gen. phase
Initial nr. Miss-classified train	4882 ± 303	
Initial nr. Miss-classified val	978 ± 54	
nr. Miss-classified train	3024 ± 1511	1607 ± 772
nr. Miss-classified val	723 ± 200	576 ± 34
Total time (s)	205.84 ± 177.15	412.22 ± 234.26

categories 3 and 8 from the CIFAR-10 dataset and is considered easy. Experiment B run on categories 3 and 2 from the same dataset and is considered more challenging. We measure the miss-classification counts over the validation and training set and the algorithm tun time.

Table 6.3: Comparing LL and NL variant on category 3 and 8 instances of CIFAR-10, Experiment A

	LL variant	NL variant
nr. Runs	10	10
Initial nr. Miss-classified train	3283 ± 1342	
Initial nr. Miss-classified val	654 ± 267	
nr. Miss-classified train	1924 ± 767	840 ± 274
nr. Miss-classified val	389 ± 146	194 ± 49
Total time (s)	65.21 ± 40.61	51.53 ± 20.85

Tables 6.3 and 6.4 show the results of the two variants over the two experiments. In both cases, the nonlinear loss function variant appears to lead to less miss-classifications. The performance on the validation set approximates the generalization capability of the method. The total computational time varies significantly, but seems reasonable for both variants.

Figure 6.3 contains various graphs depicting different measures for the experiment. The x-axis gives the number of iterations (or number of BLs) and the y-axis the miss-classification score of training set and validation set for the LL and NL variant of the DDA algorithm. Experiment A refers to the run on the CIFAR-10 instance with categories 3 and 8. Note that the miss-classification goes down fast when using the nonlinear loss function.

Table 6.4: Comparing LL and NL variant results on categories 3 and 2 instances of CIFAR-10, Experiment B

	LL variant	NL variant
nr. Runs	10	10
Initial nr. Miss-classified train	4661 ± 312	
Initial nr. Miss-classified val	933 ± 60	
nr. Miss-classified train	4248 ± 642	1607 ± 772
nr. Miss-classified val	861 ± 124	576 ± 34
Total time (s)	717.19 ± 717	412.22 ± 234.26

We now study the behaviour of both variants of DDA using the data of Experiment B. Instead of using a stopping criterion in the refinement phase, we fix the number of iterations. By doing so, we can illustrate that the miss-classification does not improve anymore for the LL variant after the sixth iteration. Using the nonlinear loss function, the miss-classification keeps an improving trend when generating more base learners, *i.e.,* with increasing iterations. This is illustrated in Figure 6.4. Interestingly, the average diversity has the same tendency as illustrated in Figures 6.4e and 6.4f.

In our experiment, we observe the behaviour of the algorithm when we vary the data reduction parameter ν, which influences the number of data points that are considered active, setting $\lambda = \frac{1}{\nu|T|}$ in (6.6). We use the data of Experiment B and use five runs. We measure the miss-classification of the data and the total running time. The resulting values are reported in Table 6.5.

Table 6.5: Results for Experiment B data varying data reduction $\nu \in (0, 1)$

ν	Miss-classified train	Miss-classified val.	Total time
0.1	2380 ± 908	625 ± 72	254.27 ± 227.60
0.4	2520 ± 862	592 ± 114	134.47 ± 77.12
0.5	2061 ± 486	511 ± 52	138.88 ± 93.65
0.6	2324 ± 163	510 ± 26	112.93 ± 63.15
0.7	2273 ± 132	502 ± 24	135.25 ± 49.40
0.8	3828 ± 670	787 ± 112	236.41 ± 227.02
1.0	4365 ± 442	874 ± 78	94.87 ± 29.49

Table 6.6: Highest base learner and ensemble accuracy with strong base learners (s. 6.10), data reduction $\nu = 0.5$

	Highest BL Accuracy		Highest Ensemble Accuracy	
	Train	Validation	Train	Validation
Run 1	94.16	78.80	**96.80**	**82.00**
Run 2	94.33	78.20	**94.91**	**78.65**
Run 3	95.08	79.25	**95.47**	**79.45**
Run 4	95.25	78.65	**95.48**	78.65
Run 5	74.68	73.60	**75.05**	**77.40**

6.4.3 Comparison of base learner and ensemble accuracy

In our last series of experiments we compare the highest base learner and ensemble accuracy, to measure the predictive prowess of the model. The first experiment is run with base learners that have an average validation accuracy of 77.7% over five runs. They are considered to be strong base learners. The second experiment is run with weaker base learners that have an average validation accuracy of 72.47% over five runs. The hyper-parameter configuration for the strong and weak base learners is listed in Table 6.10. They are considered to be weak base learners.

Table 6.7: Highest base learner and ensemble accuracy with weaker base learners (s. 6.10), data reduction $\nu = 0.5$

	Highest BL Accuracy		Highest Ensemble Accuracy	
	Train	Validation	Train	Validation
Run 1	71.44	71.6	**81.8**	**79.3**
Run 2	74.41	73.2	**83.37**	**78.65**
Run 3	73.38	73.35	**83.7**	**78.4**
Run 4	73.15	70.65	**84.08**	**79.85**
Run 5	75.25	73.55	**82.54**	**78.45**

6.5 Discussion and Conclusion

This chapter introduces and analyses an algorithm that iteratively enhances an ensemble of base learners using a column generation approach. Following an initial improvement through error-based bootstrapping, the algorithm employs a data selection

strategy that prioritizes active data points, i.e., those with non-zero dual values, which are particularly sensitive to the margin (objective). The implementation of the algorithm is part of the ensemble framework DECOLEARN and it allows various experimental setups to reliably improve an initial solution by adding base learners to an ensemble. All experiments were executed in a binary classification setting. The focus of the algorithm is on the refinement phase that performs linear programming on an iteratively growing set of score functions to maximize a soft margin. The LP constructs an ensemble from weighted score functions, populates the set of active data until an adjustable limit is reached and sets a stopping criterion for the algorithm. In numerical experiments, we investigated a variant with the traditional linear loss function and a nonlinear loss function based on cross entropy.

Experiments demonstrate that the nonlinear variant, which focuses on solving nonlinear sub-problems defined by active data only, is potentially more effective in reducing generalization error compared to the variant that employs a linear loss function. Additionally, experiments with this variant reveal a correlation between generalization error and a broader definition of ensemble diversity, based on base learners $h_i(x) \in [-1, 1]$, thus relaxing the original assumption set by [119]. We hypothesize that, because the underlying distribution of the active data set can deviate significantly from the original distribution, the proposed algorithm enhances both accuracy and diversity, particularly when the active data set –by definition, containing the most challenging data points–undergoes significant changes. We summarize our findings by the following bullet points:

- We have presented a column generation based decomposition algorithm DDA for optimizing ensemble models, optimizing the ensemble one base learner at a time and only using a sub-set of the training data for each base learner.
- We showed that algorithm DDA computes an optimal ensemble model in the convex hull of a given ANN model space.
- The results in Table (6.6) and (6.7) show that the generated ensemble models have the potential to reach an accuracy which is significantly better than the accuracy of the individual base learners, demonstrating that the ANN model space used in the experiments is a proper sub-set of its convex hull.
- Using reduced data and non-linear sub-problems seems beneficial to the LPBoost approach.
- We pose the following hypothesis: Diversity emerges from fitting the model with active data set R. The size of R can be chosen deliberately small. However, a small active data set may deteriorate accuracy.

- Model diversity as defined by [119], seems to be correlated to the generalization error.
- We found that the generation phase can improve the final ensemble results at higher computational cost.
- The implementation of the generation phase accelerates convergence. It is also recommended not to use a fixed iteration number as the stopping criterion during the generation phase because it significantly increases the likelihood of generating ineffective base learners."
- Ultimately, the additional computational cost to generate an ensemble model does not justify the minor accuracy improvements of the model.

Appendix: Configuration of Algorithm Meta-parameters and Model Hyper-parameters

The settings of the algorithm meta-parameters in the numerical experiments are summarized in Table 6.8. The model type of the base learners obtained in both the generation and refinement phases is set as CNN. The threshold of active data is set as zero, that is, all instances with non-zero dual values are considered active. The meta-parameter ν of Algorithm 6.3 is taken as $\nu = 0.2$, while the meta-parameter σ of the termination condition for the generation phase in Algorithm 6.1 is set to $\sigma = 0.2$.

Table 6.8: Decolearn Metaparameters

Parameter Name	Value
Nu	2e-1
Generation model type	CNN
Refinement model type	CNN
Combining Mechanism	Averaging
Active Data Threshold ϕ	0.0
Min. Generation Phase Change σ	2e-1

Base Learner Hyper-parameters for DDA The hyper-parameters and configurations for the base learners in the DDA algorithm are presented in Table 6.9. The configurations of base learners in the generation phase are slightly different from the ones in the refinement phase, *i.e.,* , the number of episodes in the refinement phase is increased from five to eight. The NL variant uses binary cross entropy, while the LL

variant is based on a linear loss. Table 6.10 shows the base learner configuration in the DDA algorithm of base learners that significantly differ in their accuracy.

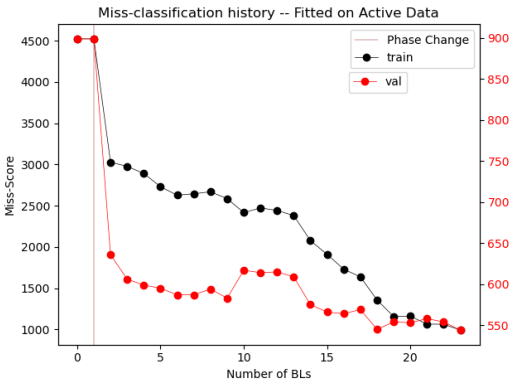
Table 6.9: Configuration of base learners for NL and LL comparison.
BCE: BinaryCross-entropy, L: Linear

Parameter Name	Generation	Refinement
Architecture	VGG	VGG
Learning Rate	1e-3	1e-3
Loss	BCE/LL	BCE/LL
Activation Function	tanh	tanh
Last Layer Activation	tanh	tanh
Batch Size	50	50
Episodes	5	8
Layers	(32, 32, 16)	(32, 32, 16)
Filter Size	(2, 2)	(2, 2)
Max. Pooling Filter	(2, 2)	(2, 2)
FC-Layers	(8, 1)	(8, 1)

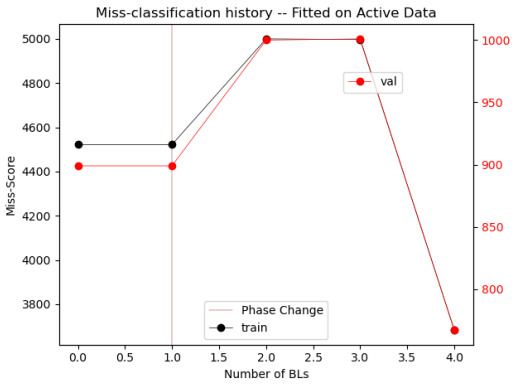
The investigation of this chapter has been published in [128].

Table 6.10: Configuration of base learners. BCE: Binary Cross-entropy.

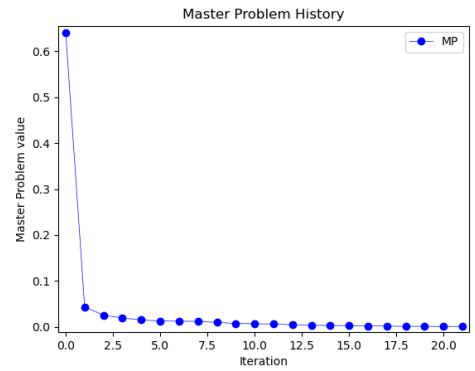
	Strong BLs		Weak BLs	
Parameter Name	Generation	Refinement	Generation	Refinement
Architecture	VGG	VGG	VGG	VGG
Learning Rate	1e-3	1e-3	1e-3	1e-3
Loss	BCE	BCE	BCE	BCE
Activation Function	tanh	tanh	tanh	tanh
Last Layer Activation	sigmoid	sigmoid	relu	relu
Batch Size	25	25	25	25
Episodes	7	7	7	7
Layers	(64, 32, 32)	(64, 32, 32)	(64, 32, 32)	(64, 32, 32)
Filter Size	(2, 2)	(2, 2)	(2, 2)	(2, 2)
Max. Pooling Filter	(2, 2)	(2, 2)	(2, 2)	(2, 2)
FC-Layers	(15, 1)	(15, 1)	(15, 1)	(15, 1)



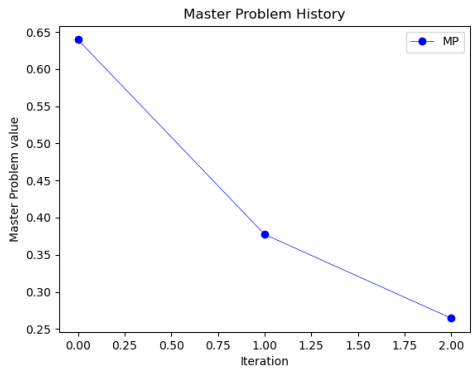
(a) Miss-classification increasing iterations NL variant DDA



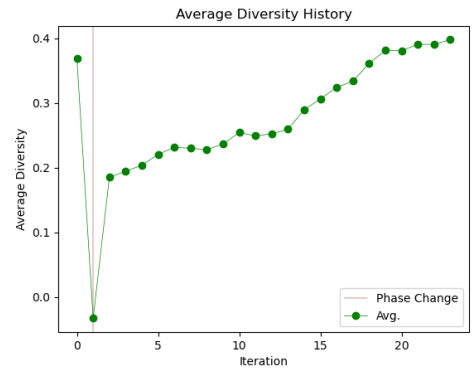
(b) Miss-classification increasing iterations LL variant DDA



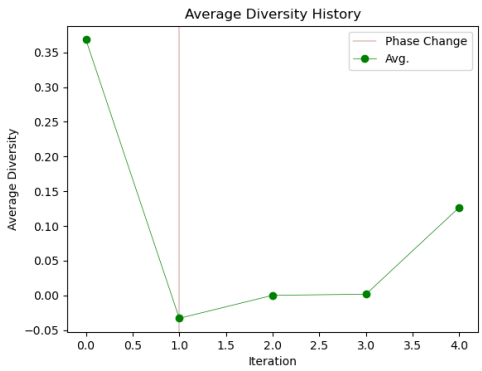
(c) Master objective function value over refinement iterations for NL variant



(d) Master objective function value over refinement iteration for LL variant

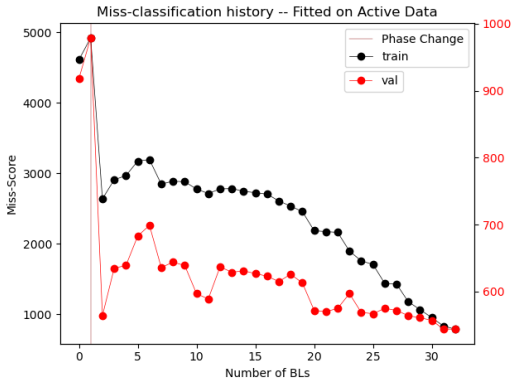


(e) Average diversity developemnt for NL variant DDA

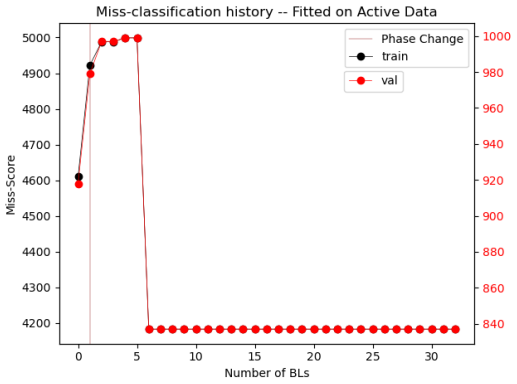


(f) Average diversity development LL variant

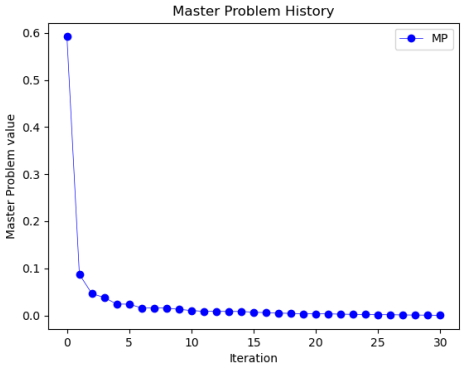
Figure 6.3: Experiment B. Miss-classification for increasing iterations (generated BLs) for the LL and NL variants



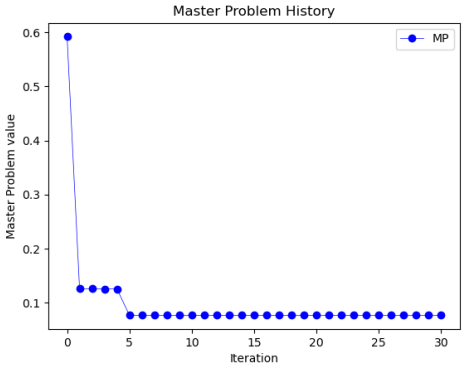
(a) Miss-classification increasing iterations NL variant DDA



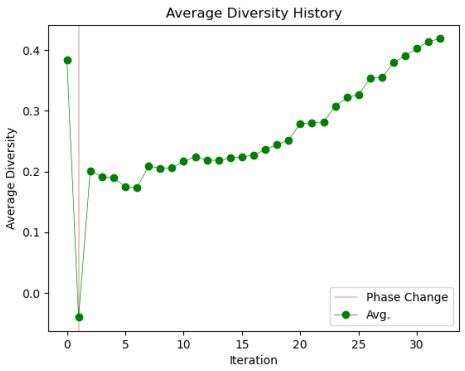
(b) Miss-classification increasing iterations LL variant DDA



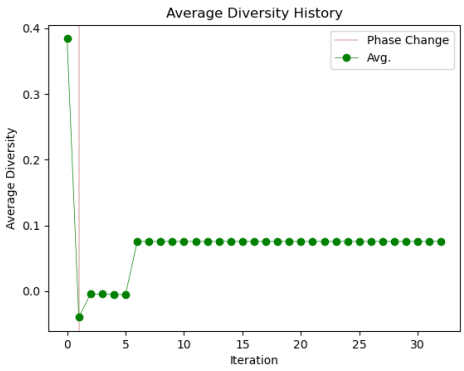
(c) Master objective function value trend of the NL variant for refinement phase



(d) Master objective function value trend of the LL variant for refinement phase



(e) Average diversity trend of NL variant



(f) Average diversity trend of LL variant

Figure 6.4: Experiment fixing 30 refinement phase iterations. Miss-classification of data points for NL and LL variant

7 Latent Space Boosting

The previous chapter introduced a linear methodology for constructing ensembles. This chapter presents a non-linear ensemble generation technique that leverages the properties of neural networks. Concretely, the approach exploits the fact that neural networks, as composite functions, map input data through multiple layers of abstraction. For classification tasks, the last layer encodes the distribution over class predictions, representing the highest level of abstraction. The hidden layer activations (outputs) – often referred to as latent variables – encode rich representations at lower levels of abstraction. This chapter introduces an innovative approach that constructs ensembles not from the final output layer but instead from these latent variables. The approach operates on the assumption that latent variables, when taken as a whole from one layer, offer a much richer representation of the data distribution. By forming the ensemble from latent variables and separately training a discriminator on top of them, the proposed method enhances the generalization capacity of the ensemble. To further improve the accuracy of the ensemble, the training loss function is augmented with an additional diversity term. This term explicitly measures the difference between the parameters of each base learner and the aggregate parameters of the ensemble. The objective is to encourage base learners to make accurate predictions while establishing parameter configurations that are sufficiently distinct from those of other learners. Ideally, each base learner should occupy an orthogonal direction in the function space, ensuring maximum independence and minimal redundancy among base learners.

7.1 Introduction

Currently minor improvements in image classification tasks are achieved by significantly increasing model complexity. This trend has been ongoing for the last years and high performance models now usually have millions of parameters [15, 129, 130]. Inspired by the results of [131], we investigate the potential of Ensemble Learning (EL) to iteratively extend an ensemble of feature extractors and fit a classifier (combination layer) on the features as latent variables. A new EL algorithm is implemented as part of the EL-framework DECOLEARN, containing the LPboost EL-method of [128]. The main differences of the new EL approach compared to other work of the authors [128] are

1. Formulating a non-linear network-based ensemble learning problem to solve classification tasks, replacing linear ensemble construction methods.
2. The base learners are used to generate non-linear feature transformations. These transformations are integrated into the neural network through a combination layer.

Hence, our contribution to the work of [131] is to generate a novel ensemble architecture through a boosting approach in an efficient manner. Efficiency is defined to be the ratio between accuracy and model complexity. An approximate measure for model complexity is the total number of base and classifier model parameters. The base learners, serving as feature transformers, are intentionally overfitted on a subset of the training data, aligning with the approach of [131]. We sketch the idea of boosting in the latent space in Section 7.2 and a description of the algorithm in Section 7.3. We reflect on the research question in Section 7.4.

7.2 Latent Space Boosting

A general learning task is to learn a function $f : \mathbb{R}^l \to \mathbb{R}^m$ to approximate an unknown mapping $\Psi : \mathcal{X} \to \mathcal{Y}$ between the input space $\mathcal{X}$ and output space $\mathcal{Y}$. Typically the sets $\mathcal{X}$ and $\mathcal{Y}$ are not known and we only have a finite set of matching observations $\{(\mathbf{x}_j, \mathbf{y}_j)\}_{j \in J} \subset \mathcal{X} \times \mathcal{Y}$. We consider multi-class classification problems, that is $\mathcal{Y} = [M] := \{1, 2, \ldots, M\}$. Given a training set $T = X \times Y$ and a model space $\mathcal{F}$, the problem aims to obtain a predictive model $f^* \in \mathcal{F}$ which minimizes an expected loss function $\mathcal{L}$, i.e. a risk, regarding hypothesis $f \in \mathcal{F}$

$$f^* = \operatorname{argmin}\{\mathbb{E}[\mathcal{L}(f(x), y)] : f \in \mathcal{F}\}, \tag{7.1}$$

where $\mathbb{E}$ denotes the expectation regarding a joint probability distribution $P(x, y)$ over $\mathcal{X} \times \mathcal{Y}$. The training set T is thought of as drawn i.i.d. from $P(x, y)$.

Let ϕ denote a final layer and g a transformer (transformation layer) and $f = \phi \circ g$ be an approximated solution of (7.1). The image space of g is called *latent space*. A latent space Base Learner (BL) is any weak transformer that maps the features from the original space to points in the latent space. The classifier takes the transformations of the base learners as input. Together they compromise the new ensemble architecture.

Let $\mathcal{G} = \{g_1, \ldots, g_Q\}$ be a set of latent space BLs. Consider a combination layer MLP, represented as $\hat{\phi}(z)$ and a transformation layer $G = (g_q)_{q \in [Q]}$ mapping from X to Z^Q. Here, $Q = |\mathcal{G}|$ denotes the index of transformers in the ensemble and Z^Q represents the latent space considering all transformers. Each base learner is fitted on a subset of data $S \subset X \times Y$, called working set. "Fitting" is a standard learning problem and there is a large variety of efficient tools for solving such problems to at least local optimality.

An ensemble learning (EL) classifier $\hat{f}(x, w) = \hat{\phi}(G(x, w_1); w_2)$ is a parameterized machine learning model, where $(w_{1q})_{q \in [Q]}$ is the parameter vector of the transformation layer and w_2 is the parameter vector of the combination layer. Figure 7.1 depicts this concept. The complexity of the transformers are arbitrary. Experiments with both low and high model complexities will be carried out to investigate performance.

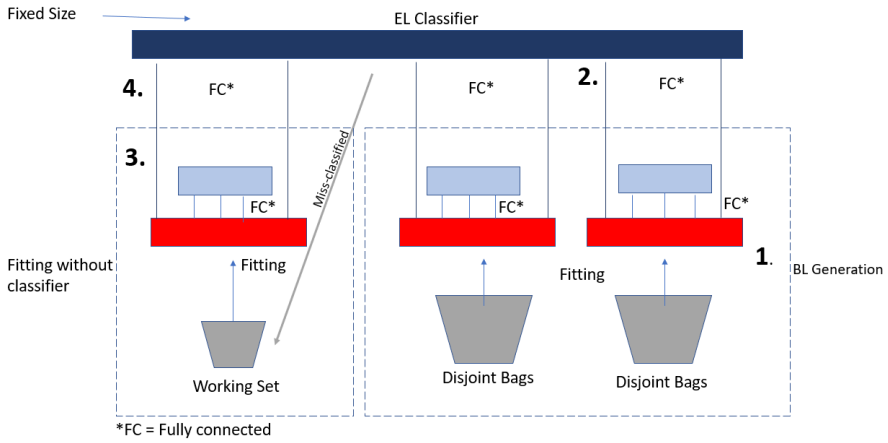


Figure 7.1: 1. Base learners are fitted on disjoint bags, Point 1. The classifier is fitted on $\mathcal{G}$ while w_1 is being fixed (Point 3). Generation of working set from miss-classified data points and the new base learner are fitted on the working set. parameters w_2 is fixed (Point 4). Retrain the classifier on the new $\mathcal{G}$

7.3 Algorithm

Algorithm 7.1 Latent Space Boosting

1. Partition training dataset T into p subsets $S_1, \ldots, S_p$
2. For all $j \in [p]$: solve LatentSpaceSubproblem(S_j) and add the transformation $\tilde{g}$ to $\mathcal{G}$
3. Compute $\hat{f}$ by solving CombinationLearningProblem($\mathcal{G}$)
4. Determine the set S of miss-classified data points of $\hat{f}$
5. Solve DiverseLatentSpaceSubProblem(S) and add the transformation $\tilde{g}$ to $\mathcal{G}$
6. Compute $\hat{f}$ by solving CombinationLearningProblem($\mathcal{G}$)
7. Repeat steps 4-6 until stopping criterion is reached
8. Return $f(x) = \text{argmax}_{m \in [M]} \hat{f}_m(x)$

Algorithm 7.1 describes the new boosting method to be investigated. It generates initial base learners by solving thesub-problem LatentSpaceSubProblem(S):

$$\tilde{\phi} \circ \tilde{g} = \text{argmin}\{ \sum_{(x,y) \in S} \mathcal{L}(f(x, w), y) : w \in \mathcal{W}_{\mathcal{CNN}} \}, \tag{7.2}$$

where $\mathcal{L}$ denotes the loss of point (x, y) and $\mathcal{W}_{\mathcal{CNN}}$ denotes the parameter space of a convolutional neural network, although the concept is also applicable to other learners. Each iteration of the main loop (steps 4-6) generates a new (diverse) base learner fitted on the miss-classified data by solving sub-problem DiverseLatentSpaceSubProblem(S):

$$\tilde{\phi} \circ \tilde{g} = \text{argmin}\{ \sum_{(x,y) \in S} \mathcal{L}(f(x, w), y) - \delta ||\bar{w}(\mathcal{G}) - w||^2 : w \in \mathcal{W}_{\mathcal{CNN}} \}, \tag{7.3}$$

where $\bar{w}(\mathcal{G})$ denotes the average network parameters of the previously generated base learners of $\mathcal{G}$, as suggested in [132]. An ensemble learning classifier is obtained by solving the sub-problem CombinationLearningProblem($\mathcal{G}$):

$$\hat{f} = \text{argmin}\{ \sum_{(x,y) \in T} \mathcal{L}(f(x, w), y) : w \in \mathcal{W}_{\mathcal{MLP}}(\mathcal{G}) \}, \tag{7.4}$$

where $\mathcal{W}_{\mathcal{MLP}}(\mathcal{G})$ denotes the parameter space of a MLP regarding base learners $\mathcal{G}$.

7.4 Results and Conclusion

This chapter presented an exploratory investigation into a novel ensemble learning strategy, termed Latent Space Boosting. The approach deviates from conventional ensemble methodologies by aggregating the latent representations of base learners, rather than their final-layer outputs. While this concept introduces an innovative perspective to ensemble learning, its practical implications remain limited.

Numerical experiments indicate that the proposed method results in a slight reduction in empirical generalization loss and are therefore not discussed in detail. However, the improvement is marginal at best, and the performance gains are insufficient to justify the additional computational complexity introduced by each base learner. Therefore, while the strategy demonstrates theoretical interest, it does not offer tangible benefits for real-world applications.

In conclusion, Latent Space Boosting should be regarded primarily as an academic exercise that enriches the understanding of ensemble strategies, but it lacks immediate applicability in practical scenarios.

8 Conclusions and Future Work

8.1 Conclusion

The main objective of this thesis was to develop a novel DRL algorithme informed by an extensive literature review, analysis of existing RL and DRL approaches, and exploratory experiments with statistical models. To this end, a maximum-entropy, state-of-the-art, Cramér-based DRL algorithm, termed Cramér-based Distributional Soft Actor-Critic, C-DSAC, was proposed and rigorously investigated. In Chapter 2, foundational concepts of RL, DRL, and entropy-augmented RL were reviewed. Section 2.3 provided a detailed examination of moment-agnostic and fixed-moment distributional approaches. Gaussian and Gaussian Mixture Models were analysed as viable modelling choices for the latter. Experimental results indicated that Gaussian models are both expressive and numerically stable, making them suitable for distributional modelling. However, the restrictive assumption of a specific return distribution introduces a parametric approximation bias. Empirical evidence in Chapter 3 showed that the benefits of using Gaussian models outweigh the additional bias.

The main contribution of this thesis, the C-DSAC algorithm, was introduced in Chapter 3. C-DSAC is a novel off-policy, maximum-entropy distributional RL Algorithm within the Actor-Critic framework, demonstrating superior performance compared to existing RL algorithms. The C-DSAC convergence properties were thoroughly analysed and it was demonstrated that C-DSAC converges in the policy evaluation setting. Moreover, under mild assumptions described both in Chapter 3 and in Section 2.1.2, C-DSAC was shown to converge in the control setting. Subsequent analysis revealed that policy iteration with C-DSAC leads to convergence to the optimal value distribution and optimal policy. A noteworthy discovery during this investigation is

the phenomenon of confidence-driven model updates: state-action values with high variance, induced either by approximation noise or inherent stochasticity in the environment, are interpreted as having low confidence in its accuracy, resulting in updates that have a less effect on the approximator parameters. This mechanism was proven to mitigate overestimation bias effectively. Empirical evaluations in complex benchmark environments demonstrated that C-DSAC achieves higher returns in fewer iterations compared to established RL algorithms, albeit with some trade-off in stability. **These findings support the conclusion that C-DSAC surpasses current state-of-the-art RL/DRL algorithms.**

Chapter 4 explored a novel model-augmented RL meta-approach tailored for robotic applications, named Reinforcement Learning-Inverse Kinematics (RL-IK). This approach decouples the planning and execution tasks, assigning an RL algorithm for motion planning and the Levenberg-Marquardt algorithm for movement execution. Experimental results demonstrated the superiority of RL-IK over vanilla RL variants, indicating high potential for addressing challenging robotic control tasks.

Chapters 6 and 7 shifted focus to the supervised learning (SL) aspects of the thesis. Chapter 6 examined an ensemble learning approach that combines strong base learners by solving a linear program defined from their predictions on the training set. Chapter 7 investigated a non-linear ensemble approach where each base learner is defined to be a sub-graph of the entire ensemble graph. After adding each base learner graph to the ensemble, a multi-layer perceptron (MLP) classifier is retrained on the resulting extended latent representation of the input. While both methods reduce empirical generalization error, the non-linear graph-based approach performs slightly better. However, the marginal improvement in generalization error does not justify the substantial computational overhead of training additional complex base learners. These results suggest that future research in SL should prioritize the development of innovative network architectures rather than ensemble methods.

In conclusion, this thesis achieves the defined objectives. The development, analysis, and theoretical validation of the novel state-of-the-art DRL algorithm C-DSAC represent a significant contribution to the field. Parallel research efforts in SL provided insights into ensemble learning, ultimately recommending a shift toward innovative network designs over computationally intensive ensemble methods. This shift should mirror the transformative progression from recurrent neural networks to transformers for sequential prediction problems. Future research directions are discussed in the following section.

8.2 Future Work

Several promising research avenues can be explored to further advance the field of (distributional) reinforcement learning and – more generally – machine learning. Investigating fundamental aspects of machine learning can even contribute to a broader understanding of human intelligence and consciousness. In the immediate future, improvements to C-DSAC may include enhancing value representations through multi-modal distributions. Leveraging the meta-learning techniques discussed in Section 2.1.5 could further optimize performance. Additionally, incorporating an adaptive temperature adjustment mechanism, as proposed in [133], may improve the exploration-exploitation trade-off, leading to more robust learning dynamics.

In the broader context, reinforcement learning is poised to play a dominant role in the pursuit of Artificial General Intelligence (AGI). For instance, the Alberta Plan [51] outlines a research strategy that emphasizes continuous RL in real-world environments while considering computational constraints and multi-agent interactions. This plan introduces a generalized agent concept that extends traditional RL architectures by integrating additional perception, transition, policy, and critic modules, each optimizing distinct objectives. The transition module, in particular, holds significant potential for implementing temporal abstraction, akin to RL-IK (Chapter 4). However, unlike RL-IK, which relied on a predefined low-level controller, the Alberta Plan envisions a more flexible, model-free learning paradigm where models are learned and continuously refined through experience. Future research could explore the adaptation of these principles within the context of distributional RL.

Another promising research direction is the integration of RL with foundation models, such as large language models (LLMs). Foundation models encode vast amounts of knowledge about the world from a human perspective. This knowledge can be leveraged to effectuate a warm-start mechanism for decision-making problems. Conversely, RL algorithms are increasingly used in fine-tuning LLMs, as evidenced by the recent application of a pure RL flywheel to optimize model behaviour [134]. While RL-driven training loops hold promise for other domains, current approaches to most problems remain computationally prohibitive, necessitating the innovation of more efficient RL training paradigms.

Beyond algorithmic advancements, a paradigm shift in the architecture of function approximators may drive the next major leap in performance. This paradigm shift would be analogous to the transition from RNNs to transformers in sequence modelling tasks. Notably, approaches based on the transformer architecture have already

been explored in offline RL settings with "Decision Transformers", yielding promising yet debated results [135, 136]. Another critical consideration is the role of hardware acceleration in RL and ML in general. Just as Tensor Processing Units exhibit large performance advantages in deep learning, tailored hardware could significantly enhance RL scalability. Quantum Reinforcement Learning, grounded in fundamental quantum principles, represents a promising frontier. The inherent properties of quantum mechanics – superposition, interference, and entanglement – could significantly enhance state-space exploration, action sampling, and model optimization. However, defining quantum RL metrics remains a challenging task. An alternative approach to novel hardware involves the integration of biological neural netorks into machine learning. The human brain demonstrates exceptional efficiency in both spatial and energetic terms, suggesting that biologically inspired architectures may hold the key to more advanced approaches in this field with profound implications to RL.

Ultimately, continued advancements in machine learning will not only drive technological progress but may also provide deeper insights into the fundamental nature of intelligence itself. Given the striking parallels between biological and artificial learning systems – both in their ability to interact with the world and form abstractions – it is the author's firm belief that future developments in this field will contribute significantly to the understanding of cognition, creativity, and intelligence.

A

Publications arising from this Thesis

The investigation for the present thesis resulted in a number of publications.

A.1 Journal Publication

Vanya Aziz, Ivo Nowak, Ouyang Wu, Eligius M. T. Hendrix, and Jan Kronqvist. On optimizing ensemble models using column generation. *J Optim Theory Appl*, 203(1):1794–1819, 2024

Impact factor JCR 2023: 1.6. Subject categories: Applied Mathematics - Q2 (89/332); Operations Research & Management Science - Q3 (65/106);

A.2 Submitted Journal Publication

Vanya Aziz, Ivo Nowak, and Eligius M. T. Hendrix. Distributional RL Using the Cramér distance. *J. Mach. Learn. Res., submitted*, 2024

Impact factor JCR 2023: 4.1. Subject categories: Automation & Control Systems - Q1 (21/84); Computer Science, Artificial Intelligence - Q2 (45/197).

A.3 Publication in International Conference Proceedings

Vanya Aziz, Ivo Nowak, Ouyang Wu, Jan Kronqvist, and Eligius M. T. Hendrix. Boosting via Latent Representation. In *Proceedings OLA'2023 International Conference on Optimization and Learning*, pages 150–153, 2023

Resumen en Español

8.4 Resumen

El término "Industria 4.0" describe la cuarta revolución industrial y se caracteriza por la integración de la tecnología digital en los procesos de manufactura. Los conceptos transformadores de la Industria 4.0 permiten una producción económica con tamaños de lote radicalmente pequeños, lo que exige niveles sin precedentes de automatización y adaptabilidad. Estos requisitos en las instalaciones de producción requieren sistemas que actúen de manera autónoma y se optimicen a sí mismos. El campo del aprendizaje automático ofrece soluciones prometedoras para lograr los objetivos de la Industria 4.0, particularmente al permitir la toma de decisiones basada en datos y mecanismos de control adaptativos. Esta disertación se centra en el Deep Reinforcement Learning, (DRL), un enfoque basado en redes neuronales para resolver Procesos de Decisión de Markov en espacios de alta dimensionalidad con dinámicas de transición desconocidas. La principal contribución de esta tesis es el desarrollo de un novedoso algoritmo de aprendizaje por refuerzo distribuido de vanguardia dentro del marco Actor-Crítico de máxima entropía. Este algoritmo, denominado "Cramér-based Soft Distributional Soft Actor-Critic" (C-DSAC), demuestra un rendimiento superior en comparación con otros algoritmos de aprendizaje por refuerzo, especialmente en entornos con espacios de alta dimensionalidad y dinámicas complejas. Su rendimiento se debe en parte a un fenómeno que surge en el Aprendizaje por Refuerzo Distribucional basado en la métrica de Cramér, denominado "actualizaciones del modelo impulsadas por la confianza". Este mecanismo garantiza que el aproximador de la función de valor se actualice de manera más conservadora cuando la confianza en sus estimaciones es baja. Se presentan

justificaciones teóricas para el algoritmo, demostrando su convergencia en el escenario de evaluación de políticas y, bajo suposiciones ampliamente aceptadas, también en el escenario de control. Más allá de la investigación algorítmica fundamental, esta tesis contribuye a la aplicación práctica del aprendizaje por refuerzo en la robótica. Dada la importancia crucial de los sistemas robóticos multiarticulados en la tecnología de producción moderna, se diseña un meta-algoritmo de aprendizaje por refuerzo denominado "Reinforcement Learning - Inverse Kinematics" (RL-IK). Este enfoque mejora la aplicabilidad del aprendizaje por refuerzo en tareas de control robótico al acelerar significativamente la convergencia hacia políticas cercanas a óptimas en comparación con los métodos estándar de aprendizaje por refuerzo. Un requisito esencial para las aplicaciones del aprendizaje por refuerzo en sistemas de control del mundo real es la percepción de la máquina para la identificación de estados. Para abordar los desafíos en este campo, esta tesis explora nuevos enfoques de Aprendizaje Supervisado (Supervised Learning, SL), validados en tareas de clasificación de imágenes, con un enfoque en estrategias de aprendizaje por conjuntos (ensemble learning). Para proporcionar una comprensión teórica más profunda, los métodos se analizan desde la perspectiva de la optimización clásica y se comparan con enfoques heurísticos. Esta investigación ofrece conocimientos valiosos sobre las dificultades de encontrar conjuntos de clasificadores óptimos a nivel global.

Capítulo 1

El primer capítulo establece el marco de referencia para esta tesis. Introduce el concepto de aprendizaje automático junto con varios términos esenciales y conceptos clave necesarios para comprender los temas más avanzados que se exploran en los capítulos siguientes. El capítulo incluye una breve digresión técnica sobre uno de los principales impulsores del éxito de las herramientas contemporáneas de aprendizaje automático: los poderosos aproximadores de funciones, en particular las redes neuronales artificiales, inspiradas en los principios de funcionamiento de las neuronas biológicas. Finalmente, este capítulo expone las principales motivaciones y objetivos de la investigación presentada en esta tesis, a saber, contribuir a la comprensión del aprendizaje por refuerzo distribuido y desarrollar un nuevo algoritmo de aprendizaje por refuerzo distribuido que idealmente supere a los algoritmos de vanguardia actuales. Además, se investigan métodos de aprendizaje por conjuntos como un enfoque complementario para tareas de aprendizaje supervisado. Esta sección detalla los objetivos específicos que guían la investigación desarrollada en los capítulos siguientes. La principal métrica para la

mejora con respecto a los algoritmos existentes de aprendizaje por refuerzo se define como la convergencia a niveles deseados de desempeño en menos iteraciones que los algoritmos de vanguardia actuales.

Capítulo 2

Este capítulo introduce al lector a la importancia del "Reinforcement Learning" (RL) y contextualiza la investigación presentada en la tesis. Comienza con una breve presentacíon de la historia de la investigación en RL. Se presentan los Procesos de Decisión de Markov con dinámicas de transición desconocidas y grandes espacios de solución como el marco en el que se pueden describir los problemas de RL. Tras una descripción intuitiva del problema, se define formalmente y se establecen algunas suposiciones ampliamente utilizadas y poco restrictivas sobre el mismo. En este capítulo se revisan diversas estructuras dentro del RL: enfoques basados únicamente en críticos, aprendizaje por diferencia temporal (TD Learning), búsqueda directa de políticas, arquitecturas actor-crítico, función de ventaja y estimador de ventaja generalizada, algoritmos on-policy y off-policy, así como el aprendizaje por refuerzo multiagente. A continuación, se detallan diversos desafíos inherentes al campo del RL, incluyendo el dilema de exploración-explotación, el sesgo de sobreestimación en métodos basados en Q, la complejidad temporal en entornos de gran dimensión y la brecha sim-to-real. Dado su papel crucial, la ingeniería de recompensas se aborda por separado en la Sección 2.1.4 y se describe en detalle. En particular, se advierte al lector sobre cómo señales de recompensa aparentemente intuitivas pueden inducir comportamientos no deseados en los agentes, y se analiza este problema en el contexto de agentes hipotéticamente superinteligentes. Además, en la Sección 2.1.4.1, se expone el desafío paradigmático en RL y se introduce la noción de esparsidad en la función de recompensa. La densidad de la recompensa influye de manera significativa en la eficiencia del aprendizaje: aunque aprender a partir de recompensas escasas es el escenario ideal, a menudo conduce a una convergencia lenta o fallida. El diseño de recompensas densas puede acelerar el entrenamiento, pero requiere una considerable intervención humana, asemejándolo al aprendizaje supervisado y potencialmente entrando en conflicto con los principios de aprendizaje autónomo. Asimismo, se explice el concepto de alineación de valores, un aspecto clave para generar modelos de control que respeten principios éticos humanos, en el contexto del modelado de recompensas. En la Sección 2.1.5, se presentan algunos de los principales meta-algoritmos de RL que han demostrado reducir el sesgo de sobreestimación, mejorar la estabilidad numérica en el entrenamiento de modelos y priorizar

experiencias más valiosas para acelerar la convergencia. También se introducen diversas técnicas de inicialización de modelos para establecer parámetros iniciales bien definidos, y se detalla la atenuación progresiva de la tasa de aprendizaje (learning rate annealing), que contribuye a la estabilidad en las actualizaciones del modelo. Para ilustrar los conceptos expuestos en secciones anteriores, la Sección 2.1.6 introduce varios algoritmos de RL, cada uno destacando distintos aspectos fundamentales de este campo. Algunos de estos algoritmos también sirven como base para métodos más avanzados. La Sección 2.1.6.1 explica el algoritmo clásico de Q-Learning en su versión tabular, introduciendo el concepto clave de funciones de valor Q, sobre el cual se basa la principal contribución de esta tesis. En la Sección 2.1.6.3, se presenta REINFORCE, un algoritmo de RL basado en políticas directas que incorpora la función de ventaja. Las siguientes dos secciones introducen PPO y DDPG, dos algoritmos representativos del paradigma actor-crítico en configuraciones on-policy y off-policy, respectivamente. Posteriormente, el capítulo explora el concepto de aprendizaje por refuerzo con máxima entropía. Los conceptos novedosos presentados en esta tesis tienen su fundamento en este marco, reconocido por su robustez y alto rendimiento en tareas complejas y de alta dimensionalidad. La notación utilizada en esta sección refleja las bases teóricas del esquema actor-crítico. La formulación del objetivo con entropía aumentada se detalla en la Sección 2.2.1, seguida de la presentación del marco soft actor-critic y sus justificaciones teóricas. El capítulo concluye con una exposición sobre el "Distributional Reinforcement Learning" (DistRL). A diferencia de los métodos tradicionales basados en el valor esperado, DistRL se centra en modelar la distribución completa de los retornos. Esta aproximación, relativamente reciente, ha demostrado en múltiples ocasiones un rendimiento superior en comparación con sus variantes base y, por lo tanto, es de gran interés en la comunidad de investigación en RL. En la sección dedicada a DistRL, se presentan demostraciones rigurosas sobre la convergencia en los escenarios de evaluación y control bajo la métrica de Wasserstein. La elección del modelo de distribución en DistRL tiene un impacto significativo en el rendimiento, por lo que se exploran diversas estrategias. Se introduce el concepto de distribuciones de momentos libres, que no imponen restricciones sobre sus momentos y permiten una adaptación arbitraria de los mismos a costa de un alto costo computacional. En contraste, las distribuciones de momentos fijos están completamente caracterizadas por sus dos primeros momentos, lo que las hace computacionalmente más eficientes. Se llevan a cabo experimentos exhaustivos utilizando modelos Gaussianos y Gaussianos Mixtos, lo que lleva a la selección de núcleos Gaussianos individuales como el modelo de distribución preferido para las principales contribuciones de esta tesis, las cuales se detallan en el siguiente capítulo.

Capítulo 3

El Capítulo 3 describe el trabajo principal de esta tesis, el desarrollo de un novedoso algoritmo de "Reinforcement Learning" (RL) de última generación en el marco de Actor-Crítico de máxima entropía denominado "Cramér-based Distributional Soft Actor-Critic" (C-DSAC).

Basado en la evaluación de políticas utilizando la métrica de Cramér, se deriva una función de pérdida regularizada que mejora la eficiencia del aprendizaje y la robustez del algoritmo. Este algoritmo representa un avance respecto al estado del arte en aprendizaje por refuerzo y cuenta con un marco teórico riguroso que garantiza la convergencia y la estabilidad de las actualizaciones, incluso en presencia de un alto nivel de ruido ambiental.

En la Sección 3.2, se explora un nuevo enfoque distribucional para RL y se presenta un método para actualizar adaptativamente los modelos de valores-Q distribucionales utilizando la distancia de Cramér regularizada. Sobre estas bases, y en combinación con el aprendizaje por refuerzo de máxima entropía, se propone el marco Cramér-based Distributional Soft Actor-Critic (C-DSAC). Además, se derivan las fórmulas correspondientes para su implementación con redes neuronales y se presentan datos experimentales que demuestran un rendimiento de última generación en los entornos evaluados.

Para explicar la eficiencia del nuevo enfoque, se realiza un análisis exhaustivo de la dinámica del algoritmo. Se pone énfasis en el comportamiento del algoritmo en presencia de errores de aproximación y ruido del sistema, y se derivan ecuaciones para los gradientes del valor. Se demuestra que la adaptación a los valores-Q es más lenta en pares de estado-acción donde la varianza del modelo distribucional es grande, es decir, cuando la confianza en su estimación es baja. Además, se enfatiza cómo estas ecuaciones revelan el mecanismo inherente del algoritmo para mitigar el sesgo de sobreestimación.

Los teoremas en esta sección proporcionan una base teórica para el enfoque distribucional basado en Cramér para SAC. La Sección 3.2.1 define la distancia de Cramér regularizada e introduce la noción de una métrica ideal. Se concluye que, debido a tres propiedades— invarianza ante sumas, sensibilidad a la escala y muestras de gradiente no sesgadas—, la distancia de Cramér regularizada constituye una métrica ideal.

En la Sección 3.2.2, se define el objetivo distribucional de máxima entropía. La recompensa se modela explícitamente como una variable aleatoria, lo que permite el uso de una función de valor distribucional. Similar al algoritmo SAC (Sección 2.2.3), la entropía de la política se mide mediante la probabilidad logarítmica negativa.

La evaluación de políticas en C-DSAC se demuestra tener un punto fijo único en la Sección 3.2.3. La demostración utiliza las propiedades de Cramér para mostrar que la función de valor distribucional predice eventualmente el retorno de una política arbitraria. La mejora de políticas se describe en la Sección 3.2.4. Se sigue un razonamiento similar al de la mejora de políticas en SAC, con la adición de que la media de la función de valor distribucional es considerada en el paso de mejora. Posteriormente, se demuestra que la iteración de políticas para C-DSAC converge a la distribución de valores y política óptimos cuando los pasos de evaluación y mejora se alternan.

En la Sección 3.3, los resultados teóricos de las secciones anteriores se utilizan para derivar funciones objetivo parametrizadas adecuadas para la implementación. Para manejar los grandes espacios de estados y acciones en tareas complejas, se emplean redes neuronales como aproximadores de funciones paramétricas con el objetivo de optimizar sus parámetros. Se asume que la variable de valor sigue una distribución gaussiana, es decir, C-DSAC se implementa como un algoritmo de momento fijo, como se describe en la Sección 2.3.5.1. Los detalles de implementación incluyen estrategias para la evaluación y mejora de políticas. Basado en estos resultados, el algoritmo completo de C-DSAC se presenta en el Algoritmo 3.1.

El análisis en la Sección 3.4 muestra el efecto de la distancia de Cramér regularizada en la dinámica del algoritmo y tiene como objetivo proporcionar una posible explicación de su rendimiento superior observado. Además, se demuestra que C-DSAC posee un mecanismo inherente para contrarrestar el sesgo de sobreestimación.

El análisis concluye que la actualización de la red neuronal de distribución de valores mediante gradientes de la función de pérdida de Cramér regularizada $J_Z(\theta)$ mejora el modelo especialmente en aquellas regiones donde la varianza—medida de la incertidumbre—es pequeña. Se argumenta que la actualización del modelo en presencia de alto ruido (baja confianza) es más propensa a la sobreestimación. Sin embargo, dado que el gradiente tiene un impacto reducido en los parámetros del modelo en estos casos, este efecto mitiga naturalmente la sobreestimación.

La sección experimental (3.5) evalúa C-DSAC en entornos de referencia de diversa complejidad, cuantificando su rendimiento para sustentar las afirmaciones de la tesis respecto a su superioridad sobre enfoques alternativos.

Los experimentos se realizan con los mismos valores de hiperparámetros que en [30], excepto por los específicos de C-DSAC (por ejemplo, límites en la desviación estándar, etc.). Se reconoce que detalles auxiliares al algoritmo principal y la "optimización a nivel de código" pueden impactar significativamente el rendimiento de un algoritmo [84]. Se asume que la implementación original utilizada para los resultados base en

[30] no aplicó técnicas de optimización adicionales como recorte de la función de valor, inicialización ortogonal, escalado de capas, ajuste de la tasa de aprendizaje, recorte de observaciones y recompensas, o normalización de observaciones y recompensas. Para ser coherente con estos supuestos, en los experimentos no se utilizan estas técnicas de optimización a nivel de código.

El agente es entrenado en los entornos de OpenAI: Hopper-v4 $(\mathcal{S} \times \mathcal{A}) \in \mathbb{R}^{11} \times \mathbb{R}^{3}$, Ant-v4 $(\mathcal{S} \times \mathcal{A}) \in \mathbb{R}^{111} \times \mathbb{R}^{8}$, Humanoid-v4 $(\mathcal{S} \times \mathcal{A}) \in \mathbb{R}^{376} \times \mathbb{R}^{17}$, HalfCheetah-v4 $(\mathcal{S} \times \mathcal{A}) \in \mathbb{R}^{17} \times \mathbb{R}^{6}$ yWalker2d-v4 $(\mathcal{S} \times \mathcal{A}) \in \mathbb{R}^{17} \times \mathbb{R}^{6}$. Siguiendo las recomendaciones de [86], se reporta la acción media de la política objetivo cada 1.000 iteraciones, con una duración de episodio de 1.000 pasos. El agente es entrenado por un millón de iteraciones en cada entorno y los resultados se promedian sobre cinco semillas aleatorias. Se muestra que C-DSAC supera a SAC en 4 de los 5 entornos evaluados. En particular, en el entorno más complejo, Humanoid-v4, C-DSAC supera claramente a SAC y alcanza una recompensa media de aproximadamente 5730, valor que SAC solo logra después de aproximadamente cuatro millones de iteraciones. Se concluye que C-DSAC es superior a SAC.

Capítulo 4

Este capítulo explora la aplicación de algoritmos de aprendizaje por refuerzo (RL) en el campo de la robótica y presenta un enfoque novedoso diseñado específicamente para tareas de manipulación robótica. El algoritmo propuesto, denominado Reinforcement Learning - Inverse Kinematics (RL-IK) [87], integra el conocimiento de la cinemática del robot y puede interpretarse como un algoritmo de RL basado en modelos (model-based RL). Esta integración permite un aprendizaje más eficiente al aprovechar el modelo físico subyacente del robot, particularmente porque el movimiento del robot y la planificación de la tarea se separan y asignan a módulos distintos. El enfoque de este capítulo está centrado en robots de múltiples articulaciones en un espacio tridimensional $(3D)$, controlados mediante políticas aprendidas dentro del marco actor-crítico (actor-critic, AC). El objetivo del algoritmo es mejorar el rendimiento de los robots en tareas que implican manipulación y posicionamiento, combinando cinemática inversa con aprendizaje por refuerzo para la planificación y toma de decisiones en un nivel de abstracción superior. El capítulo comienza con una introducción detallada a la cinemática directa, estableciendo el estándar Denavit-Hartenberg para la descripción de matrices de transformación. A continuación, se definen las transformaciones rotacionales y traslacionales, las cuales se utilizan para especificar la posición y orientación

del efector final (end-effector) del robot en el espacio. Se introduce el problema de la cinemática inversa y se exponen los diversos métodos numéricos empleados para calcular la configuración articular del robot a partir de un estado objetivo. Tras analizar diferentes enfoques, se establece que el método de Levenberg-Marquardt es una solución numérica suficientemente eficiente para abordar este problema. En la Sección 4.3, se describe la idea central de RL-IK: en el contexto del aprendizaje por refuerzo aplicado a la robótica, resulta natural distinguir entre la planificación del movimiento y la ejecución física de los desplazamientos del robot. Esta separación permite una arquitectura de control más modular e interpretable. RL-IK se basa explícitamente en esta segmentación de tareas: en cada paso de decisión, el agente de RL genera una posición y orientación objetivo para el efector final en función de su política. Las ventajas de RL-IK se exponen en la misma sección. En primer lugar, reduce el espacio de acciones. Independientemente del número de grados de libertad (Degrees of Freedom, DoF) del robot, el espacio de acción se mantiene fijo y definido únicamente por las posiciones y orientaciones del efector final. En segundo lugar, los movimientos del robot resultan más suaves. En tercer lugar, se evitan singularidades en la configuración del robot. Por último, la eficiencia en la utilización de muestras (sample efficiency) mejora de manera drástica en la tarea gracias a RL-IK. Para el algoritmo propuesto, el diseño de la función de recompensa desempeña un papel crucial. Se plantea un esquema de recompensas que incentiva al agente a resolver la tarea de manera eficiente sin infringir restricciones de seguridad ni condiciones operativas. Finalmente, en la Sección 4.5, RL-IK se compara con variantes básicas de RL en la resolución de una tarea robótica. Los resultados obtenidos evidencian que RL-IK acelera significativamente el proceso de aprendizaje.

Capítulo 5

Este capítulo marca un cambio en el enfoque de esta tesis, pasando del Aprendizaje por Refuerzo (RL) al Aprendizaje Supervisado (SL), y sirve como introducción a los capítulos posteriores. Aunque el núcleo principal de este trabajo ha sido el aprendizaje por refuerzo, debido a su aplicabilidad en el desarrollo de modelos avanzados para el control, es fundamental reconocer el papel complementario del aprendizaje supervisado en la resolución de ciertos desafíos específicos, particularmente en la percepción automática y la identificación de estados dentro de un entorno dinámico. El capítulo comienza explicando que el aprendizaje supervisado es un paradigma fundamental dentro del aprendizaje automático. En este enfoque, los modelos aprenden

a partir de datos etiquetados—entendidos como señales de supervisión, generalmente anotadas por humanos—para realizar predicciones precisas. El objetivo central del aprendizaje supervisado es la construcción de un modelo predictivo que pueda generalizar adecuadamente a instancias no vistas, identificando patrones relevantes a partir de ejemplos etiquetados. En términos formales, la meta del aprendizaje supervisado es desarrollar un modelo estadístico capaz de aproximar con precisión una función de mapeo que asocie las características de entrada con sus etiquetas de salida correspondientes. Dentro de este paradigma, los modelos basados en redes neuronales desempeñan un papel fundamental debido a su capacidad para aproximar funciones altamente generalizables, es decir, funciones que pueden desempeñarse eficazmente en datos distintos de los utilizados durante el entrenamiento. Por esta razón, la parte de esta tesis dedicada al aprendizaje supervisado se centra específicamente en el contexto de las redes neuronales. En la Sección 5.2, se introduce el Aprendizaje en Conjunto (Ensemble Learning) como el eje central de este trabajo en aprendizaje supervisado. Inspirado en el principio de la "Sabiduría de las Multitudes" (Wisdom of Crowds), el aprendizaje en conjunto busca construir modelos robustos mediante la combinación de múltiples modelos base (Base Learners, BLs) débiles pero diversos, con el objetivo de gestionar el equilibrio entre sesgo y varianza. Posteriormente, en las Secciones 5.2.1, 5.2.2 y 5.2.3, se profundiza en las estrategias fundamentales del aprendizaje en conjunto: "Bagging" (Bootstrap Aggregating), "Boosting" y "Majority Voting", analizando sus principios, ventajas y aplicaciones en distintos contextos de aprendizaje automático

Capítulo 6

Este capítulo presenta un novedoso algoritmo de boosting que integra principios de generación de columnas, una técnica bien establecida en optimización clásica, con el aprendizaje supervisado basado en redes neuronales. El enfoque propuesto, denominado algoritmo de aprendizaje en conjunto con reducción de datos (DDA, por sus siglas en inglés), optimiza un conjunto de modelos resolviendo un problema de programación lineal (LP) definido por las predicciones en el conjunto de entrenamiento. El algoritmo construye una combinación convexa óptima a partir de un conjunto de modelos base, aprovechando el concepto de margen suave, generalmente utilizado en máquinas de soporte vectorial. A través de esta optimización, el conjunto busca maximizar el rendimiento predictivo en los datos de entrenamiento mientras mantiene su capacidad de generalización. El conjunto de modelos base se amplía de manera iterativa, y la generación de nuevos modelos base está guiada por el desempeño del conjunto en puntos de

datos específicos del conjunto de entrenamiento. Para ello, se emplean aproximadores basados en redes neuronales y técnicas de aprendizaje supervisado con el fin de ajustar nuevos modelos base diversos que corrijan los errores previos del conjunto. Este trabajo busca tender un puente metodológico entre las técnicas tradicionales de optimización y los marcos modernos de aprendizaje basados en redes neuronales. La Sección 6.2 introduce LPBoost [117] y la Generación de Columnas como los principales fundamentos del algoritmo propuesto. Se centra en tareas de clasificación binaria de imágenes y en la construcción lineal de un modelo en conjunto. El problema de aprendizaje supervisado se define formalmente en la Sección 6.2.1, donde se proporciona una justificación teórica para un enfoque de generación de columnas basado en conjuntos de redes neuronales. En la Sección 6.2.2 se define el problema de combinación convexa lineal y se introduce una función de pérdida sustituta, el margen suave, para abordar el error de generalización. Con base en este problema, se formula un problema dual para determinar si un nuevo modelo base contribuye a aumentar el margen suave. La Sección 6.2.3 introduce el concepto de dispersión en un problema de programación lineal y reporta que la solución del problema maestro dual es típicamente un vector disperso. Esta observación resulta clave para implementar el algoritmo DDA de manera eficiente. Un enfoque interesante para estimar el rendimiento del modelo en conjunto es analizar la diversidad de sus modelos base en combinación con su precisión. La Sección 6.2.4 define la métrica de diversidad y la monitorea como un indicador adicional de la capacidad de generalización del conjunto. Basándose en la teoría establecida, la Sección 6.3 describe el algoritmo DDA, el cual se define mediante dos fases: una fase de generación basada en el principio de Bagging y una fase de refinamiento que opera sobre los principios del algoritmo LPBoost. La fase de generación proporciona las columnas iniciales necesarias para la formulación del programa lineal. Bajo supuestos relativamente fuertes, se demuestra que el algoritmo converge hacia el margen óptimo suave. El capítulo continúa con la presentación de experimentos numéricos en la Sección 6.4. Se evalúa la mejora en la precisión tras la ejecución del algoritmo y se presentan resultados intermedios de la fase de generación para analizar la contribución de cada una de las fases. Finalmente, el capítulo concluye con la observación de que, si bien DDA genera modelos en conjunto que superan en precisión a los modelos base individuales, su costo computacional no justifica las leves mejoras de precisión obtenidas. DDA permanece como un ejercicio académico que arroja luz sobre la intersección entre el aprendizaje automático y la optimización clásica y, aunque no resulta útil en la práctica, explora conceptos teóricos relevantes para el análisis y la comprensión de los principios de funcionamiento de los métodos de aprendizaje supervisado.

Capítulo 7

Este capítulo presenta una técnica innovadora para la generación de conjuntos no lineales que aprovecha las propiedades de las redes neuronales. Concretamente, el enfoque propuesto explota el hecho de que las redes neuronales, como funciones compuestas, transforman los datos de entrada a través de múltiples capas de abstracción.

En tareas de clasificación, la última capa de la red codifica la distribución sobre las predicciones de clases, representando el nivel más alto de abstracción. Sin embargo, las activaciones de las capas ocultas —frecuentemente denominadas variables latentes— encapsulan representaciones ricas en niveles inferiores de abstracción. En este contexto, este capítulo introduce un enfoque innovador que construye conjuntos no a partir de la capa de salida final, sino a partir de estas variables latentes.

La metodología propuesta parte de la premisa de que las variables latentes, cuando se consideran en su conjunto dentro de una misma capa, proporcionan una representación significativamente más amplia de la distribución de los datos. Así, en lugar de fusionar las predicciones finales de múltiples modelos, la técnica construye el conjunto directamente sobre estas representaciones intermedias. Para potenciar aún más la capacidad de generalización del conjunto, se incorpora un discriminador entrenado sobre las variables latentes.

Además, con el fin de mejorar la precisión del conjunto, la función de pérdida del entrenamiento se complementa con un término adicional de diversidad. Este término cuantifica explícitamente la diferencia entre los parámetros de cada aprendiz base y los parámetros agregados del conjunto. El objetivo de este mecanismo es doble: por un lado, incentivar que los aprendices base realicen predicciones precisas y, por otro, garantizar que sus configuraciones de parámetros sean lo suficientemente distintas entre sí. En términos ideales, cada aprendiz base debería ocupar una dirección ortogonal en el espacio funcional, maximizando así la independencia y minimizando la redundancia entre los modelos que conforman el conjunto.

Después de una breve introducción al problema del aprendizaje supervisado, la Sección 7.2 formaliza los conceptos relevantes. Se definen las representaciones latentes de los datos de entrada, así como los módulos de clasificación y abstracción. Se introduce el concepto de transformador, cuyo espacio imagen se denomina espacio latente. Dentro de esta arquitectura, un clasificador toma como entrada las transformaciones generadas por los aprendices base y, en conjunto, configuran la nueva estructura del conjunto. Adicionalmente, la función de pérdida se enriquece con un término que mide la distancia en norma $L2$ entre los parámetros promedio de los aprendices base en la

iteración previa y los parámetros del aprendiz base en la iteración actual del boosting.

Las conclusiones de este capítulo son similares a las del capítulo anterior. Aunque la leve mejora en la precisión del conjunto no justifica el costo computacional adicional del enfoque, los resultados empíricos obtenidos ofrecen hallazgos interesantes respecto a la diversidad de representaciones y las estrategias de conjunto basadas en grafos. Estos aspectos proporcionan nuevas perspectivas para el desarrollo de métodos más eficientes y diversificados en el aprendizaje supervisado.

Capítulo 8

Este capítulo concluye la tesis. Se reitera en él los objetivos fundamentales del trabajo, se describen las metodologías empleadas en el desarrollo de la investigación y se presentan los hallazgos más relevantes en los ámbitos del aprendizaje por refuerzo (RL) y del aprendizaje supervisado (SL). La Sección 8.2 expone diversas líneas de investigación prometedoras que pueden ser exploradas para avanzar en el campo del aprendizaje por refuerzo, tanto en su forma convencional como en su variante distribucional (DistRL), y, de manera más general, en el aprendizaje automático. Se postula que el estudio de los aspectos fundamentales del aprendizaje automático no solo impulsa el progreso en este campo, sino que también amplía nuestra comprensión de la inteligencia humana. Entre los objetivos a corto y mediano plazo, se plantea la mejora algorítmica del método C-DSAC. En particular, se considera la integración de heurísticas comunes en RL descritas en el capítulo introductorio, así como la incorporación de la técnica de auto-entropía, cuya omisión en los experimentos realizados tuvo el propósito de garantizar una comparación justa con el algoritmo SAC. Asimismo, se espera que la representación más sofisticada de distribuciones a través de Modelos de Mezcla Gaussiana (GMM) permita un incremento significativo en el rendimiento de C-DSAC. Desde una perspectiva más amplia, el aprendizaje por refuerzo está llamado a desempeñar un papel central en la búsqueda de la Inteligencia Artificial General (AGI). En este contexto, se menciona el "Plan Alberta" de Richard Sutton, una estrategia de investigación que enfatiza la necesidad de desarrollar sistemas de RL continuos que operen en entornos del mundo real, teniendo en cuenta las limitaciones computacionales y las interacciones multiagente. Este plan introduce el concepto de un agente generalizado que amplía las arquitecturas tradicionales de RL al integrar módulos adicionales de percepción, transición, política y crítica, cada uno de los cuales optimiza objetivos distintos. Otra línea de investigación de gran interés es la integración del aprendizaje por refuerzo con los modelos fundacionales, tales como los grandes modelos de lenguaje (LLMs). Los mod-

elos fundacionales encapsulan vastos volúmenes de conocimiento sobre el mundo desde una perspectiva humana. Dicho conocimiento puede ser aprovechado para implementar mecanismos de arranque eficiente en problemas de toma de decisiones. A su vez, los algoritmos de RL están siendo empleados con creciente frecuencia en la adaptación y ajuste fino de estos modelos. Un ejemplo reciente de ello es la aplicación de un ciclo de RL puro en el modelo DeepSeek R1 para optimizar su comportamiento. Más allá de los avances algorítmicos, se argumenta que un cambio de paradigma en la arquitectura de los aproximadores de funciones podría impulsar el próximo gran salto en el rendimiento de los modelos. En particular, este capítulo enfatiza la importancia de la aceleración mediante hardware innovador, posiblemente inspirado en la biología neuronal o basado en principios cuánticos. Finalmente, la tesis concluye con una afirmación ambiciosa: se predice que el aprendizaje automático no solo impulsará el progreso tecnológico, sino que también contribuirá a una comprensión más profunda de la naturaleza fundamental de la inteligencia. Dadas las sorprendentes similitudes entre los sistemas de aprendizaje biológicos y artificiales—tanto en su capacidad de interactuar con el entorno como en su habilidad para formar abstracciones—se sostiene que los desarrollos futuros en este campo jugarán un papel clave en la exploración de los mecanismos subyacentes a la cognición, la creatividad y la inteligencia misma.

Bibliography

[1] Oxford English Dictionary. Oxford english dictionary, 2023.

[2] Sarker Iqbal H. Machine learning: Algorithms, real-world applications and research directions. *SN Computer Science*, 2021.

[3] Aureilien Gelron. *Hands-on machine learning with Scikit-Learn, Keras and TensorFlow: concepts, tools, and techniques to build intelligent systems.* O'Reilly, California, 2 edition, 2019.

[4] Vladimir Nasteski. An overview of the supervised machine learning methods. *Horizons*, 4:51–62, 2017.

[5] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction.* The MIT Press, second edition, 2018.

[6] Hebb Donald O. *The organization of behavior: A neuropsychological theory.* Wiley, New York, 1 edition, 1949.

[7] Warren Mcculloch and Walter Pitts. A logical calculus of ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:127–147, 1943.

[8] F. Rosenblatt. The perceptron - a perceiving and recognizing automaton. Technical Report 85-460-1, Cornell Aeronautical Laboratory, Ithaca, New York, January 1957.

[9] Marvin Minsky and Seymour A. Papert. *Perceptrons: An Introduction to Computational Geometry.* The MIT Press, 1969.

[10] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning, 2013.

[11] Yongshuai Liu, Avishai Halev, and Xin Liu. Policy learning with constraints in model-free reinforcement learning: A survey. In Zhi-Hua Zhou, editor, *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pages 4508–4515. International Joint Conferences on Artificial Intelligence Organization, 8 2021. Survey Track.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. Attention is all you need. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

BIBLIOGRAPHY

[13] T.P. Nguyen, S. Choi, and SJ. et al. Park. Inspecting method for defective casting products with convolutional neural network (cnn). *International Journal of Precision Engineering and Manufacturing-Green Technology*, 8:583–594, 2021.

[14] A. Azadeh, S.F. Ghaderi, and S. Sohrabkhani. Annual electricity consumption forecasting by neural network in high energy consuming industrial sectors. *Energy Conversion and Management*, 49(8):2272–2278, 2008.

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

[16] OpenAI. Dall-e: Creating images from text. Blog post, 2021. `https://openai.com/blog/dall-e/`, Accessed: 2022-02-10.

[17] Sercan Arik, Jitong Chen, Kainan Peng, Wei Ping, and Yanqi Zhou. Neural voice cloning with a few samples. In S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[18] Rainer Drath and Alexander Horch. Industrie 4.0: Hit or hype?[industry forum]. *IEEE industrial electronics magazine*, 8(2):56–58, 2014.

[19] Klaus Schwab. *The fourth industrial revolution*. Crown Currency, 2017.

[20] Marc G. Bellemare, Will Dabney, and Rémi Munos. A distributional perspective on reinforcement learning. In *Proceedings of the 34th International Conference on Machine Learning - Volume 70*, ICML'17, page 449–458. JMLR.org, 2017.

[21] Alhussein Fawzi, Matej Balog, Aja Huang, Thomas Hubert, Bernardino Romera-Paredes, Mohammadamin Barekatain, Alexander Novikov, Francisco J. R. Ruiz, Julian Schrittwieser, Grzegorz Swirszcz, David Silver, Demis Hassabis, and Pushmeet Kohli. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 610:47 – 53, 2022.

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

[23] Shixiang Shane Gu, Ethan Holly, Timothy P. Lillicrap, and Sergey Levine. Deep reinforcement learning for robotic manipulation with asynchronous off-policy updates. In *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, pages 3389–3396, Piscataway, NJ, USA, May 2017. IEEE.

[24] Cosmin Paduraru, Daniel Jaymin Mankowitz, Gabriel Dulac-Arnold, Jerry Li, Nir Levine, Sven Gowal, and Todd Hester. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning*, 110:2419 – 2468, 2021.

[25] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double Q-learning. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 2094–2100. AAAI Press, 2016.

[26] 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, 2019.

[27] John Schulman, Sergey Levine, Philipp Moritz, Michael I. Jordan, and Pieter Abbeel. Trust region policy optimization, 2017.

[28] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[29] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 1587–1596. PMLR, 2018.

[30] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In Jennifer G. Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML 2018, Stockholmsmässan, Stockholm, Sweden, July 10-15, 2018*, volume 80 of *Proceedings of Machine Learning Research*, pages 1856–1865. PMLR, 2018.

[31] Marc G. Bellemare, Will Dabney, and Mark Rowland. *Distributional Reinforcement Learning*. MIT Press, 2023. `http://www.distributional-rl.org`, Accessed: 2023-09-15.

[32] Timothy H. Muller, James L. Butler, Sebastijan Veselic, Bruno Miranda, Timothy Edward John Behrens, Zeb Kurth-Nelsfon, and Steven Wayne Kennerley. Distributional reinforcement learning in prefrontal cortex. *Nature Neuroscience*, 27:403 – 408, 2021.

[33] Martin L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming*. John Wiley & Sons, Inc., New York, NY, USA, 1st edition, 1994.

[34] Bob Givan and Ron Parr. An introduction to markov decision processes.

[35] Venkat N. Gudivada, Dhana Rao, and Vijay V. Raghavan. Chapter 9 - big data driven natural language processing research and applications. In Venu Govindaraju, Vijay V. Raghavan, and C.R. Rao, editors, *Big Data Analytics*, volume 33 of *Handbook of Statistics*, pages 203–238. Elsevier, 2015.

[36] Theresa Eimer, Marius Lindauer, and Roberta Raileanu. Hyperparameters in reinforcement learning and how to tune them. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 9104–9149. PMLR, 23–29 Jul 2023.

[37] Gerald Tesauro. Td-gammon: A self-teaching backgammon program. In Alan F. Murray, editor, *Applications of Neural Networks*, pages 267–285. Springer US, Boston, MA, 1995.

[38] Ivo Grondman, Lucian Busoniu, Gabriel A. D. Lopes, and Robert Babuska. A survey of actor-critic reinforcement learning: Standard and natural policy gradients. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 42(6):1291–1307, 2012.

[39] Chris Nota and Philip S. Thomas. Is the policy gradient a gradient? In *Adaptive Agents and Multi-Agent Systems*, 2019.

[40] Dimitri P. Bertsekas and John N. Tsitsiklis. Gradient convergence in gradient methods with errors. *SIAM J. Optim.*, 10:627–642, 1999.

[41] Thomas Degris, Martha White, and Richard S. Sutton. Off-policy actor-critic, 2013.

[42] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation, 2018.

[43] Evan Greensmith, Peter L. Bartlett, and Jonathan Baxter. Variance reduction techniques for gradient estimates in reinforcement learning. *Journal of machine learning research*, 5:1471–1530, 2004.

[44] Huy Xuan Pham, Hung Manh La, David Feil-Seifer, and Aria Nefian. Cooperative and distributed reinforcement learning of drones for field coverage, 2018.

[45] Martin A. Riedmiller, Andrew W. Moore, and Jeff G. Schneider. Reinforcement learning for cooperating and communicating reactive agents in electrical power grids. In Markus Hannebauer, Jan Wendler, and Enrico Pagello, editors, *Balancing Reactivity and Social Deliberation in Multi-Agent Systems, From RoboCup to Real-World Applications (selected papers from the ECAI 2000 Workshop and additional contributions)*, volume 2103 of *Lecture Notes in Computer Science*, pages 137–149. Springer, 2000.

[46] Pawel Ladosz, Lilian Weng, Minwoo Kim, and Hyondong Oh. Exploration in deep reinforcement learning: A survey. *Information Fusion*, 85:1–22, 2022.

[47] Sebastian Thrun and Anton Schwartz. Issues in using function approximation for reinforcement learning. In Michael Mozer, Paul Smolensky, David Touretzky, Jeffrey Elman, and Andreas Weigend, editors, *Proceedings of the 1993 Connectionist Models Summer School*, pages 255–263. Lawrence Erlbaum, 1993.

[48] R. A. Howard. *Dynamic Programming and Markov Processes*. MIT Press, Cambridge, MA, 1960.

[49] Bruno Scherrer. Improved and generalized upper bounds on the complexity of policy iteration. In C.J. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K.Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc., 2013.

[50] Yashraj Narang, Kier Storey, Iretiayo Akinola, Miles Macklin, Philipp Reist, Lukasz Wawrzyniak, Yunrong Guo, Adam Moravanszky, Gavriel State, Michelle Lu, Ankur Handa, and Dieter Fox. Factory: Fast contact for robotic assembly, 2022.

[51] Richard S. Sutton, Michael Bowling, and Patrick M. Pilarski. The Alberta plan for AI research, 2023.

[52] Aravind Singh, Abhishek Gupta, Aravind Rajeswaran, Vikash Kumar, and Sergey Levine. End-to-end robotic reinforcement learning without reward engineering. In *Robotics: Science and Systems*, 2019.

[53] Nuttapong Chentanez, Andrew Barto, and Satinder Singh. Intrinsically motivated reinforcement learning. In L. Saul, Y. Weiss, and L. Bottou, editors, *Advances in Neural Information Processing Systems*, volume 17. MIT Press, 2004.

[54] Iason Gabriel and Vafa Ghazavi. 336the challenge of value alignment: From fairer algorithms to AI safety. In *Oxford Handbook of Digital Ethics*. Oxford University Press, 12 2023.

[55] Thomas Arnold, Daniel Kasenberg, and Matthias Scheutz. Value alignment or misalignment–what will keep systems accountable? In *Workshops at the thirty-first AAAI conference on artificial intelligence*, 2017.

[56] Jan Leike and Ilya Sutskever. Introducing superalignment, 2023. Accessed: 2024-10-28.

[57] Nazri Mohd Nawi, Walid Hasen Atomi, and M.Z. Rehman. The effect of data pre-processing on optimized training of artificial neural networks. *Procedia Technology*, 11:32–39, 2013. 4th International Conference on Electrical Engineering and Informatics, ICEEI 2013.

[58] Ping Luo, Xinjiang Wang, Wenqi Shao, and Zhanglin Peng. Towards understanding regularization in batch normalization, 2019.

[59] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift, 2015.

[60] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay, 2016.

[61] Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In Yee Whye Teh and Mike Titterington, editors, *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, volume 9 of *Proceedings of Machine Learning Research*, pages 249–256, Chia Laguna Resort, Sardinia, Italy, 13–15 May 2010. PMLR.

[62] Eyal Even-Dar and Y. Mansour. Learning rates for q-learning. *J. Mach. Learn. Res.*, 5:1–25, 2004.

[63] Christopher Watkins and Peter Dayan. Q-learning. *Machine Learning*, 8(3):279–292, 1992.

[64] M. G. Bellemare, Y. Naddaf, J. Veness, and M. Bowling. The arcade learning environment: An evaluation platform for general agents. *Journal of Artificial Intelligence Research*, 47:253–279, June 2013.

[65] Ronald J. Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8:229–256, 2004.

[66] David Silver. Lecture 7: Policy gradient methods, 2015. Slides for lecture on reinforcement learning, Accessed: 2023-06-05.

[67] David Silver, Guy Lever, Nicolas Manfred Otto Heess, Thomas Degris, Daan Wierstra, and Martin A. Riedmiller. Deterministic policy gradient algorithms. In *International Conference on Machine Learning*, 2014.

[68] Antonin Raffin, Ashley Hill, Adam Gleave, Anssi Kanervisto, Maximilian Ernestus, and Noah Dormann. Stable-baselines3: Reliable reinforcement learning implementations. *Journal of Machine Learning Research*, 22(268):1–8, 2021.

[69] Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes, 2022.

[70] Kaiwen Wang, Kevin Zhou, Runzhe Wu, Nathan Kallus, and Wen Sun. The benefits of being distributional: Small-loss bounds for reinforcement learning, 2023.

[71] Kaiwen Wang, Owen Oertell, Alekh Agarwal, Nathan Kallus, and Wen Sun. More benefits of being distributional: Second-order bounds for reinforcement learning. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *Proceedings of Machine Learning Research*, pages 51192–51213. PMLR, 2024.

[72] Marc G. Bellemare, Ivo Danihelka, Will Dabney, Shakir Mohamed, Balaji Lakshminarayanan, Stephan Hoyer, and Rémi Munos. The Cramer distance as a solution to biased Wasserstein gradients, 2017.

[73] Will Dabney, Mark Rowland, Marc G. Bellemare, and Rémi Munos. Distributional reinforcement learning with quantile regression. In *AAAI Conference on Artificial Intelligence*, 2017.

[74] Will Dabney, Georg Ostrovski, David Silver, and Rémi Munos. Implicit quantile networks for distributional reinforcement learning, 2018.

[75] Mark Rowland, Marc G. Bellemare, Will Dabney, Rémi Munos, and Yee Whye Teh. An analysis of categorical distributional reinforcement learning. In *International Conference on Artificial Intelligence and Statistics*, 2018.

[76] Peter J. Huber. Robust Estimation of a Location Parameter. *The Annals of Mathematical Statistics*, 35(1):73 – 101, 1964.

[77] Karl Pearson. Contributions to the mathematical theory of evolution. *Philosophical Transactions of the Royal Society of London*, 185:71–110, 1894.

[78] Karl Cobbe, Oleg Klimov, Chris Hesse, Taehoon Kim, and John Schulman. Quantifying generalization in reinforcement learning. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 1282–1289. PMLR, 09–15 Jun 2019.

[79] Aviral Kumar, Rishabh Agarwal, Tengyu Ma, Aaron C. Courville, G. Tucker, and Sergey Levine. Dr3: Value-based deep reinforcement learning requires explicit regularization. *ICML*, abs/2112.04716, 2021.

[80] Jingliang Duan, Yang Guan, Shengbo Eben Li, Yangang Ren, Qi Sun, and BO CHENG. Distributional soft actor-critic: Off-policy reinforcement learning for addressing value estimation errors. *IEEE Transactions on Neural Networks and Learning Systems*, 33(11):6584–6598, nov 2022.

[81] Xiaoteng Ma, Li Xia, Zhengyuan Zhou, Jun Yang, and Qianchuan Zhao. DSAC: Distributional soft actor critic for risk-sensitive reinforcement learning, 2020.

[82] Alix Lhéritier and Nicolas Bondoux. A Cramér distance perspective on quantile regression based distributional reinforcement learning, 2022.

[83] V M Zolotarev. Metric distances in spaces of random variables and their distributions. *Mathematics of the USSR-Sbornik*, 30(3):373, apr 1976.

[84] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep policy gradients: A case study on PPO and TRPO, 2020.

[85] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diega, CA, USA, 2015.

[86] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *AAAI Conference on Artificial Intelligence*, 2017.

[87] Vanya Aziz. Robotersteuerung mit modellgestütztem Reinforcement Learning. Master's thesis, Technische Hochschule Köln, Germany, 2021.

[88] International Federation of Robotics. Standardisation: Definition of a robot, 2024. Accessed: 2024-12-12.

[89] Sami Haddadin, Sven Parusel, Lars Johannsmeier, Saskia Golz, Simon Gabl, Florian Walch, Mohamadreza Sabaghian, Christoph Jähne, Lukas Hausperger, and Simon Haddadin. The franka emika robot: A reference platform for robotics research and education. *IEEE Robotics & Automation Magazine*, 29(2):46–64, 2022.

[90] Francesco Cursi, Weibang Bai, Eric Yeatman, and Petar Kormushev. Globdesopt: A global optimization framework for optimal robot manipulator design. *IEEE Access*, PP:1–1, 01 2022.

[91] Kenneth Levenberg. A method for the solution of certain non – linear problems in least squares. *Quarterly of Applied Mathematics*, 2:164–168, 1944.

[92] Donald W. Marquardt. An algorithm for least-squares estimation of nonlinear parameters. *Journal of the Society for Industrial and Applied Mathematics*, 11(2):431–441, 1963.

[93] Kungurtsev Bergou, Diouane. Convergence and complexity analysis of a levenberg–marquardt algorithm for inverse problems. *Journal of Optimization Theory and Applications*, 185:927–944, 2020.

[94] James Surowiecki. *The Wisdom of Crowds*. Anchor, 2005.

[95] Lior Rokach. Ensemble-based classifiers. *Artificial intelligence review*, 33:1–39, 2010.

[96] Balaji Lakshminarayanan, Alexander Pritzel, and Charles Blundell. Simple and scalable predictive uncertainty estimation using deep ensembles. *Advances in Neural Information Processing Systems*, 30, 2017.

[97] Kohavi Bauer. An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine Learning*, 36:105–139, 1999.

[98] Yoav Freund and Robert E. Schapire. A desicion-theoretic generalization of on-line learning and an application to boosting. In Paul Vitányi, editor, *Computational Learning Theory*, pages 23–37, Berlin, Heidelberg, 1995. Springer Berlin Heidelberg.

[99] Robert E. Schapire. Explaining adaboost. In *Empirical Inference*, 2013.

[100] Jerome H Friedman. Greedy function approximation: a gradient boosting machine. *Annals of statistics*, pages 1189–1232, 2001.

[101] Marc Pfetsch and Sebastian Pokutta. Ipboost–non-convex boosting via integer programming. In *International Conference on Machine Learning*, pages 7663–7672. PMLR, 2020.

[102] Faraz Torabi, Garrett Warnell, and Peter Stone. Behavioral cloning from observation. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, IJCAI'18, page 4950–4957. AAAI Press, 2018.

[103] Thomas G. Dietterich. Ensemble methods in machine learning. In *Multiple Classifier Systems*, pages 1–15, Berlin, Heidelberg, 2000. Springer Berlin Heidelberg.

[104] Omer Sagi and Lior Rokach. Ensemble learning: A survey. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 8(4):e1249, 2018.

[105] Zhi-Hua Zhou. *Ensemble Methods: Foundations and Algorithms*. CRC Press, 2012.

[106] Andreas Aakerberg, Kamal Nasrollahi, and Thomas Heder. Improving a deep learning based rgb-d object recognition model by ensemble learning. In *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*, pages 1–6. IEEE, 2017.

[107] Yen-Yu Lin, Tyng-Luh Liu, and Chiou-Shann Fuh. Local ensemble kernel learning for object category recognition. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[108] Tomasz Malisiewicz, Abhinav Gupta, and Alexei A Efros. Ensemble of exemplar-svms for object detection and beyond. In *2011 International Conference on Computer Vision*, pages 89–96. IEEE, 2011.

[109] Jie Xu, Wei Wang, Hanyuan Wang, and Jinhong Guo. Multi-model ensemble with rich spatial information for object detection. *Pattern Recognition*, 99:107098, 2020.

[110] Hao Jiang, Xiaojie Qiu, Jing Chen, Xinyu Liu, Xiren Miao, and Shengbin Zhuang. Insulator fault detection in aerial images based on ensemble learning with multi-level perception. *IEEE Access*, 7:61797–61810, 2019.

[111] Yibin Li, Yan Song, Lei Jia, Shengyao Gao, Qiqiang Li, and Meikang Qiu. Intelligent fault diagnosis by fusing domain adversarial training and maximum mean discrepancy via ensemble learning. *IEEE Transactions on Industrial Informatics*, 17(4):2833–2841, 2020.

[112] Yueping Wang, Kun Zhu, Mengyun Sun, and Yueyu Deng. An ensemble learning approach for fault diagnosis in self-organizing heterogeneous networks. *IEEE Access*, 7:125662–125675, 2019.

BIBLIOGRAPHY

[113] Abolfazl Abdollahi, Biswajeet Pradhan, and Abdullah M Alamri. An ensemble architecture of deep convolutional segnet and unet networks for building semantic segmentation from high-resolution aerial images. *Geocarto International*, pages 1–16, 2020.

[114] Mohd Hanafi Ahmad Hijazi, Stefanus Kieu Tao Hwa, Abdullah Bade, Razali Yaakob, and Mohammad Saffree Jeffree. Ensemble deep learning for tuberculosis detection using chest x-ray and canny edge detected images. *IAES International Journal of Artificial Intelligence*, 8(4):429, 2019.

[115] Alexander Thebelt, Jan Kronqvist, Miten Mistry, Robert M Lee, Nathan Sudermann-Merx, and Ruth Misener. Entmoot: A framework for optimization over ensemble tree models. *Computers & Chemical Engineering*, 151:107343, 2021.

[116] Jan Kocoń, Igor Cichecki, Oliwier Kaszyca, Mateusz Kochanek, Dominika Szydło, Joanna Baran, Julita Bielaniewicz, Marcin Gruza, Arkadiusz Janz, Kamil Kanclerz, Anna Kocoń, Bartłomiej Koptyra, Wiktoria Mieleszczenko-Kowszewicz, Piotr Miłkowski, Marcin Oleksy, Maciej Piasecki, Łukasz Radliński, Konrad Wojtasik, Stanisław Woźniak, and Przemysław Kazienko. Chatgpt: Jack of all trades, master of none. *Information Fusion*, 99:101861, 2023.

[117] Ayhan Demiriz, Kristin P Bennett, and John Shawe-Taylor. Linear programming boosting via column generation. *Machine Learning*, 46(1):225–254, 2002.

[118] Pavlo Muts, Stefan Bruche, Ivo Nowak, Ouyang Wu, Eligius MT Hendrix, and George Tsatsaronis. A column generation algorithm for solving energy system planning problems. *Optimization and Engineering*, 24:317–351, 2023.

[119] Yijun Bian and Huanhuan Chen. When does diversity help generalization in classification ensembles? *IEEE Transactions on Cybernetics*, 52(9):9059–9075, 2022.

[120] V. M. Tikhomirov. *Convex Analysis*, pages 1–92. Springer Berlin Heidelberg, 1990.

[121] Sandra Benítez-Peña, Emilio Carrizosa, Vanesa Guerrero, M. Dolores Jiménez-Gamero, Belén Martín-Barragán, Cristina Molero-Río, Pepa Ramírez-Cobo, Dolores Romero Morales, and M. Remedios Sillero-Denamiel. On sparse ensemble methods: An application to short-term predictions of the evolution of covid-19. *European Journal of Operational Research*, 295(2):648–663, 2021.

[122] Göksu Tüysüzoğlu and Dokuz Birant. Enhanced bagging (ebagging): A novel approach for ensemble learning. *International Arab Journal of Information Technology*, 17(4):515–528, 2020.

[123] D.W. Ruck, S.K. Rogers, M. Kabrisky, and J.P. Mills. Target recognition: Conventional and neural network approaches. In *International 1989 Joint Conference on Neural Networks*, pages 608 vol.2–, 1989.

[124] K. Fukushima. Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. *Biological Cybernetics*, 36(4):193–202, 1980.

[125] Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

[126] Alex Krizhevsky. Learning multiple layers of features from tiny images. *University of Toronto*, 05 2012.

[127] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[128] Vanya Aziz, Ivo Nowak, Ouyang Wu, Eligius M. T. Hendrix, and Jan Kronqvist. On optimizing ensemble models using column generation. *J Optim Theory Appl*, 203(1):1794–1819, 2024.

[129] Alexander Kolesnikov, Lucas Beyer, Xiaohua Zhai, Joan Puigcerver, Jessica Yung, Sylvain Gelly, and Neil Houlsby. Big transfer (bit): General visual representation learning. In Andrea Vedaldi, Horst Bischof, Thomas Brox, and Jan-Michael Frahm, editors, *Computer Vision – ECCV 2020*, pages 491–507, Cham, 2020. Springer International Publishing.

[130] Linnan Wang, Saining Xie, Teng Li, Rodrigo Fonseca, and Yuandong Tian. Sample-efficient neural architecture search by learning actions for monte carlo tree search. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5503–5515, 2022.

[131] Bruno Antonio, Davide Moroni, and Massimo Martinelli. Efficient adaptive ensembling for image classification. *Expert Systems*, n/a(n/a), 2023.

[132] Hassam Sheikh, Mariano Phielipp, and Ladislau Boloni. Maximizing ensemble diversity in deep reinforcement learning. In *International Conference on Learning Representations*, 2022.

[133] Tuomas Haarnoja, Aurick Zhou, Kristian Hartikainen, George Tucker, Sehoon Ha, Jie Tan, Vikash Kumar, Henry Zhu, Abhishek Gupta, Pieter Abbeel, and Sergey Levine. Soft actor-critic algorithms and applications, 2019.

[134] DeepSeek-AI. Deepseek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. Technical report, deepseek, 2025.

[135] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan, editors, *Advances in Neural Information Processing Systems*, 2021.

[136] Scott Emmons, Benjamin Eysenbach, Ilya Kostrikov, and Sergey Levine. Rvs: What is essential for offline RL via supervised learning? In *International Conference on Learning Representations*, 2022.

[137] Vanya Aziz, Ivo Nowak, and Eligius M. T. Hendrix. Distributional RL Using the Cramér distance. *J. Mach. Learn. Res., submitted*, 2024.

[138] Vanya Aziz, Ivo Nowak, Ouyang Wu, Jan Kronqvist, and Eligius M. T. Hendrix. Boosting via Latent Representation. In *Proceedings OLA'2023 International Conference on Optimization and Learning*, pages 150–153, 2023.