Contributions to the Adoption of Deep Reinforcement Learning Algorithms in Real-World Robotics

Sven Gronauer

TUM School of Computation, Information and Technology
Technische Universität München

Contributions to the Adoption of Deep Reinforcement Learning Algorithms in Real-World Robotics

Sven G. Gronauer

Vollständiger Abdruck der von der TUM School of Computation, Information and Technology der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Ingenieurwissenschaften (Dr.-Ing.)

genehmigten Dissertation.

Vorsitz: Prof. Dr.-Ing. Jörg Ott

Prüfende der Dissertation:

1. Prof. Dr.-Ing. Klaus Diepold
2. Prof. Jan Tommy Gravdahl
3. Prof. Dr.-Ing. Martin Buss

Die Dissertation wurde am 29.02.2024 bei der Technischen Universität München eingereicht und durch die TUM School of Computation, Information and Technology am 25.07.2024 angenommen.

Sven Gronauer. Contributions to the Adoption of Deep Reinforcement Learning Algorithms in Real-World Robotics, 2024.

Chair of Data Processing, Technical Universtiy of Munich, 80333 Munich, Germany,
https://www.ce.cit.tum.de/ldv.

Acknowledgements

"What I cannot create, I do not understand."

— Richard Feynman

First and foremost, I want to thank my supervisor, Professor Klaus Diepold, who made it possible for me to pursue this journey as a doctoral candidate. He leads by example to explore the topics that are dear to your heart. I am more than grateful for enjoying the freedom to choose the research directions I am most excited about. I thank Hao Shen for being my mentor over the last few years and for our fruitful discussions, especially at the beginning of my doctorate.

I want to highlight *Project Phoenix*, which originated from a fun idea and turned into a fruitful research project. Many thanks to Matthias Emde for his efforts and contributions over the last few years to both the Software Campus and the drone project. Special thanks belong to Mathias Korte, who has supported my research work as a student assistant over the last three years.

I am extremely grateful to the *Chair of Data Processing* (or LDV for short) and all colleagues who have been – or still are – part of this institution. I cannot imagine a better working environment and collegial friendship that goes far beyond academic topics. Particularly, I want to thank Martin Gottwald, Manuel Lengl, Luca Sacchetto, and Matthias Kissel not only for the plentiful discussions, which were seldom related to academic topics, but also for the priceless evenings and excursions we spent together.

Last but by no means least, I would like to express my sincere gratitude to my friends and family. To my mother and father, thank you for the never-ending support throughout my educational life, for keeping my back free from any source of concern, and for being my safe haven whenever the sea is stormy. My deepest appreciation belongs to my Spatzl Sofie and her four-legged companions Otto and Peter, who share with me the most important things in life: love and friendship. Without her tireless endeavor to push me harder, I wouldn't be at the point where I am now.

Abstract

Learning-based methods such as reinforcement learning promise to synthesize control policies from data, eliminating the need for manual controller design. The field has witnessed rapid advances since the seminal breakthrough of deep reinforcement learning: Algorithms have emerged from simulation environments and have been successfully adopted in real-world robotics. However, applying control policies to physical robots is intricate, and various challenges must be overcome to render reinforcement learning a reliable option for robot control.

This dissertation addresses the utilization of reinforcement learning algorithms in real-world robotics. The emphasis lies on transfer learning, where control policies trained in simulation are deployed to a physical robot. To this end, this thesis examines three crucial topics for successful sim-to-real transfer.

First, the performance of control policies is examined with respect to the algorithm configuration and the design of the control problem. The results evidence that a handful of carefully selected algorithm components can compete with state-of-the-art algorithm implementations on common simulation benchmark tasks. Furthermore, the representation of the policy and the level of abstraction in the action space are studied on a real quadrotor robot. The experiments show that both have a positive impact on performance.

Second, the robustness of control policies is investigated toward the model mismatch between the simulation and the real robot. To this end, policies were directly transferred from the simulation to a quadrotor robot, and the robustness was assessed based on the abstraction level of the action space and the learning representation of the control policy. As the experimental results indicate, higher abstraction levels in the action space reduce the required simulation fidelity and increase robustness. Furthermore, recurrent neural networks combined with domain randomization facilitate an adaptive control behavior, which is beneficial in partially observable systems.

The third and last topic covered in this thesis is the safety certification of a reinforcement learner. A learning-based model predictive method is used to correct potentially unsafe actions of a control policy through planning. Since the reliable enforcement of safety constraints requires accurate modeling of the robot, two approaches to improve safety are investigated. The results indicate that prior knowledge of the dynamics, derived from first principles, enhances the quality of the learning-based safety certification. Additionally, the results show that an ensemble of neural networks reduces the number of safety constraint violations at the expense of rendering the closed-loop control system more conservative.

In summary, this thesis substantiates the recent success stories of deep reinforcement learning and shows that data-driven paradigms can be used to control robots in the real world. However, a successful deployment requires careful engineering of the algorithm

components and proper design of the control problem. This finding refutes the idealistic notion that reinforcement learning is applicable without prior knowledge about the real-world system and the structure of the environment. Nevertheless, the recent advancements are promising, and the interest in adopting reinforcement learning methods in robotics is increasing.

Contents

Acronyms

CBF	control barrier function.
CMDP	constrained Markov decision process.
CPO	constrained policy optimization.
CTCE	centralized training centralized execution.
CTDE	centralized training decentralized execution.
DDPG	deep deterministic policy gradient.
DR	domain randomization.
DTDE	distributed training decentralized execution.
EKF	extended Kalman filter.
FC	fully-connected.
FNN	feed-forward neural network.
GAE	generalized advantage estimation.
GP	Gaussian process.
GRU	gated recurrent unit.
IWPG	importance-weighted policy gradient.
KL	Kullback-Leibler.
Lag-TRPO	Lagrangian trust-region policy optimization.
LBPO	Lyapunov barrier policy optimization.
LQR	linear quadratic regulator.
LSTM	long short-term memory.
MADRL	multi-agent deep reinforcement learning.
MARL	multi-agent reinforcement learning.
MBPO	model-based policy optimization.
MDP	Markov decision process.
MLP	multi-layer perceptron.
MPC	model predictive control.
MPSC	model predictive safety certification.

NN	neural network.
NPG	natural policy gradient.
PID	proportional-integral-derivative.
POMDP	partially observable Markov decision process.
PPO	proximal policy optimization.
PWM	pulse-width modulated.
RL	reinforcement learning.
RMSProp	root mean squared propagation.
RNN	recurrent neural network.
RQ	research question.
SAC	soft actor-critic.
SGD	stochastic gradient descent.
SL	safety layer.
SQRL	safety Q-functions for reinforcement learning.
TD	temporal difference.
TRPO	trust-region policy optimization.
X-MPSC	ensemble model predictive safety certification.

Chapter 1

Introduction

Modern industrial progress is driven by a continuously increasing degree of automation [1, 2]. Automation in technical processes can reduce or even eliminate human supervision by assigning the decision-making to an entity called a controller or agent, which executes concrete instructions in specific situations. The conventional engineering approach for implementing automation is to develop a system description with appropriate inputs and outputs by deriving a system formulation from physical principles [3]. Then, a feedback loop is integrated such that the system becomes a closed loop, and a controller can drive the system to a desired state [4]. Closed-loop control traditionally requires a wide range of domain knowledge for system formulation, controller design, and high-level algorithms for planning and decision-making. For that reason, controller design becomes more intricate with an increasing degree of system complexity and is challenging for systems with only partially or completely unknown dynamics. Moreover, controllers are mainly implemented once and remain unchanged after deployment [5], rendering an a priori specification of controllers a notoriously difficult task for complex systems.

Learning-based methods are a promising direction to address the automated design of control policies. The showcase example is deep reinforcement learning (RL), a data-driven paradigm for learning controllers that eludes the necessity of a manual controller design [6, 7, 8]. Deep RL utilizes neural networks (NNs) as parameterized function approximators to find low-dimensional representations of high-dimensional data [9]. The representations are trained end-to-end by adjusting the parameters of the learning representation based on data obtained from trial-and-error interaction with the system. Decisions are inferred from the learning representation in such a way as to optimize the expected control performance. What renders the framework of deep RL so powerful is that an agent does not need explicit instructions in each time step but can find meaningful relationships and patterns in data, even from sparse feedback. Moreover, an agent improves itself by learning from experience, even when starting to learn tabula rasa [10], and can adapt online by learning continually from unknown situations and reusing past interaction data [11].

The rise of deep learning methods in the early 2010s has sparked renewed interest in learning-based control. Groundbreaking scientific advances in representation learning [12, 13] and algorithm implementation [14, 15] have paved the way for the rapid advances of the research field. Furthermore, the adoption of RL is driven by the steadily increasing power of computing resources and communication networks, the mounting availability of data, and improved physics simulations.

1.1 Challenges and Motivation

Although deep RL has demonstrated its capabilities as the superior control framework in several simulation and board-game environments [16, 17, 18], the deployment to robots is challenging for a number of reasons [19]:

1. *Curse of Dimensionality.* RL relies on the interaction with the environment to learn from the consequences of the actions taken. With an increasing state and action space dimensionality, the number of required data samples rises to develop accurate representations. In fact, a discrete state-action space grows exponentially with the dimension and the number of entities in multi-agent environments [20].

2. *Generalization.* When agents are trained on a single or a set of similar but related tasks, the ability to learn abstract concepts and reuse skills on new tasks and in unseen environments is crucial for learning in the real world. Generalization describes the ability to learn from a limited amount of data and develop meaningful representations to master novel deployment conditions, preventing overfitting to the training environment [21].

3. *Non-Markovian Systems.* The underlying mathematical framework of RL relies on the idealistic assumption that the environment state is fully observable and that the operating conditions are stationary. Both assumptions are generally not applicable to robotic hardware since changes in the dynamics caused by wear, delays in actuation and sensing, or immeasurable state information render the control problem non-Markovian [22, 23].

4. *Sample Efficiency.* Deep RL algorithms are known to be sample inefficient [24]. Large amounts of data are required to find expressive representations and learn policies with high control performance. Generating large amounts of data is typically not an issue in simulation but imposes a severe limitation on real-world robot systems, on which time and monetary constraints exist [25].

5. *Safety.* Agents require exploration to learn from unseen regions of the state space and novel situations. At the same time, the safety of the robot's behavior must be enforced at all stages of the learning process to avoid failures and damage to the hardware or its environment. Thus, an unconstrained exploration is neither tolerable nor affordable for most robot systems [26, 27].

6. *Robustness.* Control policies deployed in real-world systems require a certain degree of robustness to ensure reliable performance. In particular, policies are required to withstand uncertainties resulting from measurement noise and state estimation, structural changes, and non-stationarity of the environment. Robustness is generally required in settings where the deployment conditions differ from those during training [28, 29, 30].

The last half-decade has seen remarkable advances in the adoption of RL to real-world applications. Examples in the domain of robotics include locomotion with legged robots

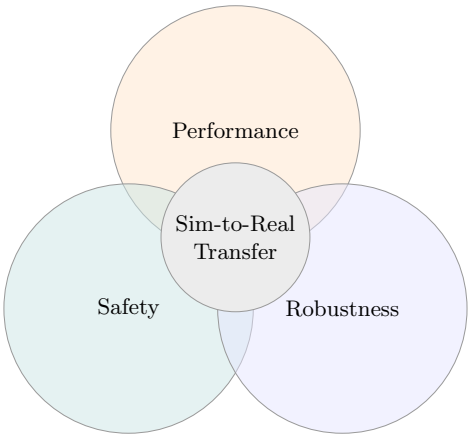


Figure 1.1: The scope of this thesis is centered around sim-to-real transfer learning. In this context, three topics are addressed: performance, robustness, and safety.

[31, 32], flying quadrotors through cluttered environments and first-person view racing [33, 34], and solving manipulation tasks with robotic hands or arms [35, 36, 37]. Beyond robotics, the first successful implementation of RL to control plasma for nuclear fusion was demonstrated [38], as well as the first use of deep learning and RL for laser welding [39]. RL likewise shows potential for solving abstract problems such as sorting and hashing [40], matrix multiplication [41], electronic chip layout [42], and chemical drug design [43].

Despite the tremendous progress, RL is not the mainstream methodology for controlling robots [5]. Engineers favor methods from control theory due to their safety and robustness guarantees under known operating conditions. In order to render RL a reliable option for robot control, several challenges must be addressed.

1.2 Scope of Research Work

The theme of this thesis is the adoption of deep RL algorithms in real-world robotic applications. In most parts, the emphasis lies on zero-shot transfer learning, where a control policy is exclusively trained in simulation and then deployed on a physical robot. An apparent advantage of a sim-to-real approach is that a theoretically infinite amount of data can be leveraged for the training. However, the quantity of simulated data comes at the cost of the so-called reality gap, a bias imposed on the control policy since the simulation model cannot perfectly fit reality. The model mismatch can result from an inaccurate simulation that does not well-characterize the real world, as well as stochasticity and non-stationarity of the environment. This thesis centers on the sim-to-real context and addresses three topics, as depicted in Figure 1.1.

First, the performance of a control policy is investigated. It is a well-known issue that experimental results underlie high variance since several high-level and low-level implementation decisions affect learning [25, 44, 45]. Hence, algorithm components and control design choices are evaluated based on their impact on policy performance.

Second, the robustness of policies toward the reality gap is studied. It is vital for successful sim-to-real transfers that policies are insensitive toward the reality gap, espe-

cially in zero-shot settings, where policies are trained exclusively in simulation, and no real-world data is used for fine-tuning the policies. Robustness is examined with respect to the level of abstraction in the action space and the learning representation of the control policy.

Third, the safety of a learning-based policy during the data collection phase is examined. Since good exploration strategies are highly stochastic and novelty-seeking, a naive deployment of an RL algorithm is not realizable on a robot, where physical safety is imperative. Thus, safety mechanisms must be embedded into the RL loop such that actions with irreversible consequences are averted and control behavior enforces the given constraints at deployment. To this end, the learning-based safety certification of potentially unsafe actions is investigated, as well as how to improve it.

1.3 Thesis Outline

This thesis is a cumulative dissertation based on Papers I–V that were published previously. Papers I–V are peer-reviewed and included as appendices to this thesis. The contributions of the research work can be divided into two parts. Part A addresses the single-agent domain and is dedicated to the topics of performance, robustness, and safety. Part B extends the scope to multiple agents and complements the six challenges introduced in Section 1.1 by reviewing challenges that arise exclusively in the multi-agent domain. The remainder of this thesis is structured as follows.

Chapter 2 covers the terminology and underlying concepts used throughout this thesis. To this end, the formal problem statement is given, followed by an introduction to the two computational frameworks used in this thesis.

Chapter 3 formulates three research questions (RQs), which are answered accordingly by verifying or falsifying hypotheses. Each RQ is associated with two or three hypotheses.

Chapter 4 consolidates the methodology used to evaluate the research hypotheses and briefly outlines the methods proposed in Papers I–V. Algorithm components and control design choices are introduced in Section 4.1. After that, the methods used to improve the robustness of control policies for sim-to-real transfer learning are covered in Section 4.2. Section 4.3 addresses the topic of safe exploration and introduces constrained RL and predictive safety filters.

Chapter 5 summarizes the context and contributions of Papers I–V. While Section 5.1 covers selected aspects of the single-agent RL problem, Section 5.2 surveys the landscape of the recent developments in multi-agent reinforcement learning (MARL).

An extended discussion about the three topics covered in this thesis is given in Chapter 6. Besides the reflection on the research work conducted in Papers I–V, a substantial amount of literature is included in this discussion. Most importantly, RQs 1–3 are addressed by providing answers based on the evaluation of the associated hypotheses.

Finally, Chapter 7 concludes this thesis by summarizing the main findings and giving a brief outlook on the adoption of RL in real-world robotics.

Chapter 2

Background

The primary purpose of this background chapter is to introduce the terminology, concepts, and definitions that are used in the subsequent chapters of this thesis. At first, the formal problem statement is given. After that, the two computational frameworks utilized in this thesis work are presented: data-driven RL and knowledge-driven model predictive control (MPC). The last part of this background chapter presents MARL and briefly introduces two major challenges of the multi-agent domain.

As for the mathematical notation, $\mathbb{R}$ denotes the set of real numbers, $\mathbb{R}^n$ is the set of n-dimensional vectors, and $\mathbb{R}^{m\times n}$ denotes the set of real-valued $m \times n$ matrices. Small Latin alphabet generally denotes vectors (*e.g.*, x, u) and functions (*e.g.*, f, r), capitalized letters denote matrices (*e.g.*, A, B) and integer variables (*e.g.*, N, H), blackboard bold letters are used for sets (*e.g.*, $\mathbb{X}, \mathbb{U}$) and Greek letters denote hyperparameters (*e.g.*, α, γ). Some deviations may occur so that the notation better aligns with the existing literature: π denotes a policy, θ and ϕ are NN parameter vectors, and V and Q denote value functions. Calligraphic font describes geometric objects like ellipsoids $\mathcal{E}$. The symbol $\oplus$ denotes the Minkowski sum of two sets and square brackets $[1, N]$ describe the set of integer numbers from 1 to N. Finally, the p-norm of a vector is given by $\|\cdot\|_p$.

This thesis borrows the terminology from the RL community but applies the notation from control theory and dynamic programming [46], *e.g.*, transition tuples are described as (x, u, r, x') instead of (s, a, r, s') used in the RL literature [8]. For the sake of readability, the time index t is dropped when the time dependency becomes apparent from the context, *i.e.*, (x, u, r, x') is used instead of (x_t, u_t, r_t, x_{t+1}).

2.1 Formal Problem Statement

The formal problem is to make a sequence of optimal decisions under uncertainty. Optimality is determined through a numerical measure called *reward*, which provides information about the success or failure of the decisions being made. The decision maker, referred to as the *agent* in the remainder of this thesis, stores information about the system in *states* and manipulates the state of the system through *actions*. In this way, the agent is able to steer the future evolution of the system and drive the system to a desired state.

System Description. More formally, dynamical systems are considered in discrete-time form

$$x_{t+1} = f(x_t, u_t, w_t) \tag{2.1}$$

with states $x_t \in \mathbb{X}$, actions $u_t \in \mathbb{U}$, and disturbances $w_t \in \mathbb{W}$ at the time step t. The behavior of the system and its evolution over time is described by the system dynamics f and can be manipulated through actions. Unless otherwise stated, the system dynamics are assumed to be unknown, which prompts the agent to learn about the input-output behavior of the system through trial and error.

Constraints. In practice, most robot systems impose restrictions in terms of actions $\mathbb{U} \subset \mathbb{R}^{n_u}$ (*e.g.*, actuator limits) and states $\mathbb{X} \subset \mathbb{R}^{n_x}$ (*e.g.*, joint position and speed constraints), where n_u and n_x describe the dimension of the respective space. The system is assumed to underlie polytopic constraints in the states $x \in \mathbb{X} = \{x \in \mathbb{R}^{n_x} \mid H_x x \leq d_x\}$ and actions $u \in \mathbb{U} = \{u \in \mathbb{R}^{n_u} \mid H_u u \leq d_u\}$. The matrices H_x and H_u are of appropriate size, and the dimensions of the vectors d_x and d_u correspond to the number of constraints present in the system.

Noise and Uncertainties. Real-world systems are subject to uncertainty, which results from process noise, corrupted sensor measurements, and inaccurate state estimation. Process noise arises from the stochastic nature of the dynamical system itself. Furthermore, robot systems exhibit noise in the actuation and sensing. Non-stationarity is another source of uncertainty caused by changes in system parameters during operation (*e.g.*, due to wear and tear) but is typically neglected in RL for practical reasons. The system description in (2.1) accounts for all sources of uncertainty with the disturbance vector w.

Objective. To make optimal decisions, the agent manipulates the future evolution of system states $(x_t, x_{t+1}, \dots)$ through a sequence of actions $(u_t, u_{t+1}, \dots)$ in such a way as to maximize the cumulative reward

$$J(u_t, \dots, u_{t+N-1}) = r_f(x_{t+N}) + \sum_{k=0}^{N-1} r(x_{t+k}, u_{t+k}) \tag{2.2}$$

over the horizon N. The reward function $r : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$ outputs a scalar signal in each time step t that contains information about the effectiveness of the agent's control behavior, while the scalar function $r_f : \mathbb{X} \to \mathbb{R}$ denotes the terminal reward. The choice of N is a subtlety of the computational framework: modern RL algorithms mainly apply the infinite horizon case $N \to \infty$, whereas MPC methods utilize a finite N over a receding horizon.

2.2 Model Predictive Control

In MPC, the control problem is set to a finite horizon N and optimizes the objective

$$\begin{aligned} \underset{u_t,\dots,u_{t+N-1}}{\text{maximize}} \quad & J(u_t, \dots, u_{t+N-1}) \\ \text{subject to} \quad & z_0 = x_t, \\ & z_{k+1} = f_{\text{prior}}(z_k, u_{t+k}) \quad \forall\, k = [0, N-1], \\ & z_k \in \mathbb{X} \quad \forall\, k = [0, N], \\ & u_{t+k} \in \mathbb{U} \quad \forall\, k = [0, N-1], \\ & z_N \in \mathbb{X}_{\text{term}}. \end{aligned} \tag{2.3}$$

At each time step t, the control problem is solved online with a receding horizon [47]. MPC assumes access to a prior model f_{prior}, which is usually derived from first principles and may be calibrated with the help of expert knowledge. The initial condition of the optimization problem is the latest measurement or estimate x_t of the actual system (2.1), and $\mathbb{X}_{\text{term}}$ is the terminal set that must be reached within N steps. The subscript k is used to denote the planning stage, where a predicted state z_k is k stages ahead of x_t. By solving the mathematical program (2.3), the open-loop action sequence $(u_t, \dots, u_{t+N-1})$ is obtained. However, only u_t is applied to the system, and (2.3) is resolved at the next time step with a shifted horizon.

MPC plans trajectories that, at the same time, optimize the objective given in (2.2) and satisfy the given constraints. This is a big benefit of MPC since constraints can be directly enforced in the optimization process.

2.2.1 Robust Model Predictive Control

When the robot is subject to uncertainties, feedback control is superior to open-loop control [48]. While conventional MPC (2.3) finds an action sequence as a solution to the open-loop control problem, robust MPC outputs a sequence of policies. Although several approaches to robust MPC exist, this thesis focuses on tube-based MPC [48].

In tube-based MPC, the model f_{prior} is used to plan a nominal state trajectory $(z_0, \dots, z_N)$ produced by the action sequence $(v_0, \dots, v_{N-1})$. In the presence of uncertainty, it is assumed that the tube contains all possible trajectories of the actual system (2.1), where each trajectory implements a particular realization of the disturbance sequence. Since tubes can grow large under uncertainty, the affine feedback

$$u_{t+k} = v_k + K(x_{t+k} - z_k) \tag{2.4}$$

is applied to track the state trajectory $(x_t, x_{t+1}, \dots)$ of the actual system toward the nominal state trajectory $(z_0, z_1, \dots)$. The matrix $K \in \mathbb{R}^{n_u \times n_x}$ implements linear feedback and is chosen such that the error system $e_k = x_{t+k} - z_k$ becomes stable.

2.2.2 Model Predictive Safety Certification

Model predictive safety certification (MPSC) implements a mechanism that prevents learning-based systems from executing unsafe actions. A closed-loop system is considered

to be safe when predefined *safety constraints* are satisfied during both the learning and deployment phases, *i.e.*, $x \in \mathbb{X}$ and $u \in \mathbb{U}$. By predicting the future state trajectory based on a nominal model, MPSC tries to find an action sequence that complies with the state and action constraints. Any RL-based control loop can be extended with MPSC so that potentially unsafe actions are adjusted to safe ones.

More formally, nominal MPSC [49] aims to find a control input v_0 that modifies the action u_t as minimally as possible by solving the objective

$$\begin{aligned} \underset{v_0,\dots,v_{N-1}}{\text{minimize}} \quad & \|u_t - v_0\|_2^2 \\ \text{subject to} \quad & z_0 = x_t, \\ & z_{k+1} = f_{\text{prior}}(z_k, v_k) \quad \forall\, k = [0, N-1], \\ & z_k \in \mathbb{X} \quad \forall\, k = [0, N], \\ & v_k \in \mathbb{U} \quad \forall\, k = [0, N-1], \\ & z_N \in \mathbb{X}_{\text{term}}. \end{aligned} \tag{2.5}$$

The MPSC objective is set to a finite horizon N and outputs the nominal state-action sequence $(z_0, v_0, \dots, v_{N-1}, z_N)$ that satisfies the constraints in states, actions, and the terminal set. The time index t is used for state measurements and actions from and applied to the actual system (2.1), whereas k denotes states and actions used for planning with f_{prior}. Thus, the predicted states z_k are k stages ahead of the time step t. MPSC was introduced for linear dynamical systems [50] and extended to nonlinear systems in later works [51, 52].

2.3 Reinforcement Learning

In RL terminology, learning describes a closed-loop process wherein the agent interacts with its environment sequentially and learns from past data, as illustrated in Figure 2.1. Since the system dynamics of the environment are inherently assumed to be unknown, the agent can only learn about the system's behavior through trial and error. At each time step, the agent observes a snapshot of the system state and selects an action based on its control policy (or just the *policy*). The agent learns about the consequences of the actions taken through rewards that indicate the agent's capability to complete a control task. It aims to maximize the cumulative reward by adapting the policy accordingly. The remainder of this chapter introduces RL as a computational framework to find good policies through data.

2.3.1 Elements of Reinforcement Learning

The Markov decision process (MDP) is the most common approach for modeling sequential decision-making with RL [53]. Its name is based on the Markov property that requires a state to capture all relevant information about the system's history, decoupling the future evolution of the system from its past.

Markov Decision Process. An MDP is defined by the tuple $(\mathbb{X}, \mathbb{U}, p, r, \mathbb{X}_0)$, where $\mathbb{X}$ and $\mathbb{U}$ are the state and action spaces, respectively [54]. The dynamical system f

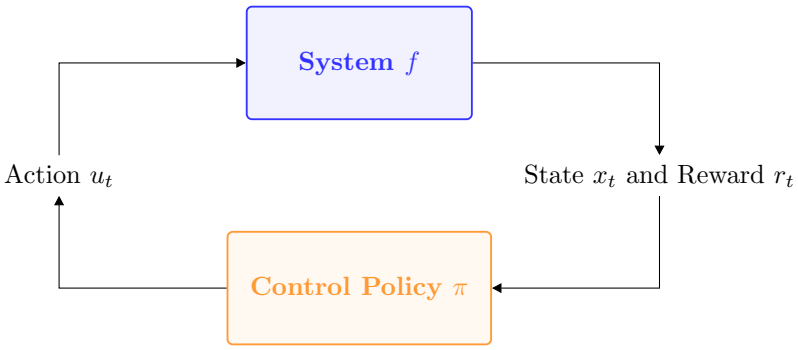


Figure 2.1: Closed-loop control with RL.

underlies a random disturbance w_t, which can be equivalently expressed by the state transition probability kernel $p : \mathbb{X} \times \mathbb{U} \to P(\mathbb{U})$ that assigns each state-action pair (x, u) a probability measure over $\mathbb{X}$. The notation $p(\cdot|x, u)$ is rather standard in the RL community to describe state transitions and is just an alternative formulation to (2.1). The reward function is $r : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$ and $\mathbb{X}_0$ denotes the initial state distribution.

Policy. The agent's behavior is encoded in the policy $\pi : \mathbb{X} \to P(\mathbb{U})$ that maps each state to a probability distribution over the action space, making the action selection stochastic, *i.e.*, $u \sim \pi(\cdot|x)$. Through an optimization process that applies incremental adjustments, the policy is updated such that the cumulative reward (2.2) is maximized. This thesis considers only policies that are stationary.

Trajectories. A trajectory, also called *episode* or *rollout* in RL terms, describes the state-action sequence $\tau = (x_0, u_0, x_1, u_1, \dots)$. In the remainder of this thesis, the notation $\tau \sim \pi$ is used as a shortcut for the trajectories produced under the policy π subject to $x_{t+1} \sim p(\cdot|x_t, u_t)$, $u_t \sim \pi(\cdot|x_t)$, and $x_0 \sim \mathbb{X}_0$.

Return. For the cumulative reward, also called *trajectory return* or just *return*, two formulations exist: finite and infinite. The finite return is given by the sum of rewards

$$R(\tau) = r_f(x_N) + \sum_{t=0}^{N-1} r(x_t, u_t) \tag{2.6}$$

over the horizon N. In the infinite horizon, the return becomes

$$R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(x_t, u_t). \tag{2.7}$$

An additional discount factor $\gamma \in (0, 1)$ is introduced such that the infinite sum converges, assuming that the reward function r is bounded. Furthermore, the discount factor can be seen as a trade-off between immediate rewards and long-term task completion. Note the slight difference between (2.2) and (2.6): MPC considers the cumulative reward over a receding horizon whereas RL takes the sum of rewards over the complete trajectory.

Optimization Problem. The agent aims to maximize the expected *policy performance*

$$J(\pi) = \mathop{\mathbb{E}}_{\tau\sim\pi}\left[R(\tau)\right], \tag{2.8}$$

by adapting the policy through small incremental steps. Since the policy performance can only be sampled when the system dynamics are unknown, Monte Carlo methods are used to obtain numerical estimates. The policy search problem to be solved through stochastic optimization [55, 56] has the form

$$\underset{\pi}{\text{maximize}} \quad J(\pi),$$

where the optimal policy π^* is sought over the space of policies Π.

Value Functions. In addition to the policy performance, which describes the expected performance of a policy, the utility of being in a particular state is described by a value function. The *state-value function* $V_\pi : \mathbb{X} \to \mathbb{R}$ under the policy π describes the utility when being in state x and taking actions according to π thereafter, *i.e.*,

$$V_\pi(x) = \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(x_t,u_t) \;\middle|\; x_0 = x\right].$$

In a similar way, the *action-value function* $Q_\pi : \mathbb{X}\times\mathbb{U} \to \mathbb{R}$ described by

$$Q_\pi(x,u) = \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t r(x_t,u_t) \;\middle|\; x_0 = x, u_0 = u\right]$$

expresses the utility of being in state x, performing action u, and following the policy π thereafter.

Bellman Equation. The value function underlying the policy π is known to satisfy the Bellman equation

$$V_\pi(x) = \mathop{\mathbb{E}}_{x'\sim p,u\sim\pi}\left[r(x,u) + \gamma V_\pi(x')\right] \tag{2.9}$$

for all $x \in \mathbb{X}$ [54]. Analogously, the action-value function under π satisfies the system of equations

$$Q_\pi(x,u) = \mathop{\mathbb{E}}_{x'\sim p,u'\sim\pi}\left[r(x,u) + \gamma Q_\pi(x',u')\right] \tag{2.10}$$

for all $(x,u) \in \mathbb{X}\times\mathbb{U}$. Except for a restricted class of problems, where linear algebraic recursion can be used [6], value functions are not known a priori and hence must be found iteratively. Temporal difference (TD) methods can be applied for the iterative computation of value functions by using bootstrapped values. The updates

$$V_{k+1}(x) \leftarrow V_k(x) + \alpha_k\delta,$$

are applied iteratively for all $x \in \mathbb{X}$ until convergence with a sufficiently small step size $\alpha_k > 0$, resulting in the fix point $\lim_{k\to\infty} V_k = V_\pi$ [7]. The one-step TD error is defined by

$$\delta = r(x,u) + \gamma V_k(x') - V_k(x), \tag{2.11}$$

where $V_k(x)$ and $V_k(x')$ are bootstrapped values at the k-th iteration of the update process.

2.3.2 Taxonomy of Reinforcement Learning Algorithms

RL algorithms can be categorized along several dimensions. Possible dimensions are the data generation process and the type of data used for policy improvement. Another dimension is whether the data collection is coupled with the learning process, *i.e.*, whether the data generation and policy improvement are an online process or training is performed on a fixed data batch.

Model-Free and Model-Based. The most fundamental distinction of RL algorithms is whether a dynamics model of the environment is leveraged for training. *Model-free* methods learn without dynamics knowledge and thus solely rely on interaction data. In contrast, *model-based* methods utilize a dynamics model, which can be specified a priori through domain knowledge [15, 57] or learned from data [58, 59, 60]. In general, model-based methods are more sample-efficient than their model-free counterparts. However, the superior sample efficiency typically comes at the cost of reduced asymptotic performance when utilizing a learned dynamics model [61].

Online and Offline. The classical RL viewpoint is that learning is an *online* process where the agent directly interacts with the environment and adapts its behavior to maximize performance. Data generation and policy improvement depict an alternating procedure in which the agent first collects data and then updates its policy. The ability to learn online from experience renders RL algorithms a versatile approach but also poses a significant challenge. The trial-and-error data collection makes the adoption of RL to robotic systems demanding due to the high sample complexity. Another line of research concentrates on *offline* RL, where data is collected once and before the training [62]. The data are kept fixed throughout the training so that the policy does not actively interact with the environment. Offline RL is a valuable approach in settings where online interaction is ineligible, either due to safety concerns or the expense of data [63].

On-Policy and Off-Policy. Another classification of RL algorithms can be seen in the policy used for the data collection. *On-policy* algorithms utilize the most recent policy iteration for creating trajectories and then update the policy consequently on this data batch. In contrast, *off-policy* algorithms can incorporate data generated by any controller in the update procedure. Data samples from former policy iterations are often reused in the updates of the current policy iteration, rendering off-policy algorithms, in general, more sample efficient than their on-policy equivalents [25, 64, 65].

2.4 Deep Reinforcement Learning

Even in a finite state-action space, the enumeration of all state-action pairs can easily exceed the available memory of a computer system. Thus, the visitation of all pairs in a discretized state-action space through sampling can become infeasible in practice. For instance, the discretization into ten bins per dimension in a continuous state space of size $\mathbb{R}^{11}$ and an action space in $\mathbb{R}^4$ results in $10^{11} \cdot 10^4 = 10^{15}$ state-action pairs. The so-called

curse of dimensionality [66] makes available data samples become sparse and necessitates the use of function approximators to estimate value functions and interpolate the space between samples.

2.4.1 Neural Networks

Deep RL copes with high-dimensional spaces by utilizing NNs as universal function approximators to represent value functions and/or policies. In this thesis, two specific types of NNs are considered: feed-forward neural networks (FNNs) and recurrent neural networks (RNNs).

Feed-Forward Neural Networks. The multi-layer perceptron (MLP) belongs to the class of FNNs since no recursive loops for memorization are used. Information flows unidirectional from input to output. MLPs are the most common form of FNNs and thus are adopted in a wide range of applications [67]. The MLP is composed of a series of layers, where each layer i computes the affine combination

$$y_i = h_i(y_{i-1}, W_i, b_i) = a\big(W_i y_{i-1} + b_i\big),$$

of the input vector y_{i-1}, the weight matrix W_i, and the bias vector b_i, followed by the nonlinear transformation a, which is known as activation function. A complete forward pass through an MLP network follows a sequence of mappings h_i over L layers [68], *i.e.*,

$$h(x, \theta) = h_L(\cdot, W_L, b_L) \circ \cdots \circ h_2(\cdot, W_2, b_2) \circ h_1(x, W_1, b_1).$$

The input to layer i is the output y_{i-1} of the previous layer, while the input to the first layer is $y_0 = x$. For compactness, the parameter vector of an NN is written as a column vector

$$\theta = \left[w_1^T, b_1^T, \ldots, w_L^T, b_L^T\right]^T,$$

which concatenates the NN parameters by stacking the flattened weights $w_i = \mathrm{vec}(W_i)$ and biases b_i. Note that the terms MLP and FNN are used interchangeably in this thesis.

Recurrent Neural Networks. The second class of NNs considered in this thesis are RNNs, which extend MLPs with feedback loops that retain their outputs for the next forward pass. RNNs are designed to process sequential data and are adopted in applications where temporal relationships must be inferred from data [69, 70]. In this thesis, only those RNN architectures that can be organized in layers are considered. Similar to the MLP, each layer implements the nonlinear transformation

$$y_i^{(t)} = h_i\left(y_{i-1}^{(t)}, y_i^{(t-1)}, W_i, b_i\right)$$

of inputs, weights, and biases, followed by an activation function at time step t. The time index $t-1$ is used to denote the outputs of the previous time step, while $i-1$ denotes the outputs from the preceding layer. Commonly selected recurrent layer units include long short-term memory (LSTM) [71] and gated recurrent units (GRUs) [72].

2.4.2 Actor and Critic

NNs play a central role in the recent success stories of RL, which can be explained by their expressiveness and the clever utilization of these function approximators [73]. This section briefly explains the role of NNs in deep RL.

Policy Representation. Deep RL generally addresses problems in continuous action spaces with policies represented by NNs, so-called *actors*. The actor π_θ is parameterized by the vector θ holding the weights and biases of the NN. In order to reinforce the exploration of the state-action space, stochasticity is induced in the selection of actions, commonly through the use of a multivariate Gaussian distribution

$$\pi_\theta(\cdot|x) = \mathcal{N}\left(h(x,\theta), S\right) = h(x,\theta) + \mathcal{N}(0,S), \tag{2.12}$$

where h parametrizes the mean of the Gaussian and S denotes the covariance matrix.

Value Function Approximation. The value functions

$$V_\phi \approx V_\pi \quad \text{and} \quad Q_\phi \approx Q_\pi$$

are approximated by NNs with parameters $\phi = \left[w_1^T, b_1^T, \ldots, w_L^T, b_L^T\right]^T$. The RL community also refers to the functions V_ϕ and Q_ϕ as *critics*. Although the Bellman equation does not hold exactly when using NNs, the system of equations (2.9) can still be solved approximately by optimizing the mean-squared TD error, *i.e.*,

$$\underset{\phi}{\text{minimize}} \quad L(\phi) = \mathop{\mathbb{E}}_{\tau\sim\pi}\left[\left(r(x,u) + \gamma V_\phi(x') - V_\phi(x)\right)^2\right].$$

Value estimates $V(x')$ of the next states are bootstrapped from data that are generated through the interaction with the environment. As new data samples are collected, the approximation V_ϕ can be refined. Since V_ϕ is a nonlinear map, the solution cannot be found in closed form but must be computed iteratively. The easiest and predominantly chosen approach for minimization is taking a step along the descending direction, *i.e.*, by going into the direction of the negative gradient $-\nabla_\phi L(\phi)$. The critic parameters are then successively updated by following the update rule

$$\phi_{k+1} = \phi_k - \alpha_k \nabla_\phi L(\phi),$$

where k describes the current iteration, and α_k is the step size.

Actor-Critic Methods. A very effective method to address continuous control problems with deep RL is the combination of an NN-based value function and an NN-based policy, both trained interchangeably [9]. An *actor-critic* method alternates between a policy evaluation step to improve the critic estimates and a policy improvement step to increase the trajectory return. For this purpose, the actor is updated along an approximate gradient direction based on the critic estimates. Actor-critics promise improved learning stability and better convergence behavior than actor-only and critic-only methods [74].

2.4.3 Policy Gradient Algorithms

A policy gradient method optimizes a policy by taking the gradient of the policy performance with respect to the policy parameters and going along the direction of the gradient [75, 76]. Since the gradients of the return (2.8) can exhibit high variance due to the accumulation of noisy one-step rewards, critics are used to guide the learning process [77]. Critics provide long-term value predictions with low variance instead of immediate reward signals. However, the reduced variance comes at the cost of inducing a bias to the gradient computation. Recently, the intersection of actor-critics and policy gradients has gained enormous popularity due to their success in various problems [78, 79, 80]. Hence, this combination is used as the method of choice throughout this thesis.

Policy gradient methods update the policy parameters θ iteratively by going into the direction p_k with the step size α_k, *i.e.*,

$$\theta_{k+1} = \theta_k + \alpha_k p_k.$$

Two common methods exist for determining the direction: line search and trust-region methods [81]. The direction is given by

$$p_k = B_k^{-1} g_k,$$

where B_k is an approximation to the Hessian and $g_k = \nabla_\theta J(\pi_\theta) \mid_{\theta=\theta_k}$ denotes the policy gradient. Since arbitrarily large policy update steps can lead to degenerative policy performance during the optimization procedure [77, 82], updates must be chosen carefully. A common measure used in policy gradient methods is the Kullback-Leibler (KL) divergence

$$D_{\mathrm{KL}}(\pi \parallel \mu) = \mathop{\mathbb{E}}_{\tau\sim\mu}\left[\int \pi(u|x) \log\left(\frac{\pi(u|x)}{\mu(u|x)}\right) du\right],$$

which evaluates the displacement in the action selection between the policies π and μ.

Stochastic Policy Gradients. Stochastic policy gradients utilize policies (2.12) that parameterize probability distributions over the action space. Based on the policy performance (2.8) in the infinite horizon case, the stochastic policy gradient is given by

$$\begin{aligned}
\nabla_\theta J(\pi_\theta) &= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\Big[\nabla_\theta R(\tau)\Big] \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\Big[R(\tau)\nabla_\theta \log p(\tau|\pi_\theta)\Big] \\
&= \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty} R(\tau)\nabla_\theta \log \pi_\theta(u_t|x_t)\right].
\end{aligned}$$

From the first to the second line, the likelihood ratio derivative $\nabla_\theta \pi_\theta = \pi_\theta \nabla_\theta \log \pi_\theta$ is applied [83]. From the second to the third equation, the likelihood function of a trajectory

$$p(\tau|\pi_\theta) = p(x_0) \prod_{t=0}^{\infty} p(x_{t+1}|x_t, u_t)\pi_\theta(u_t|x_t)$$

under π is differentiated with respect to the policy parameters and inserted, *i.e.*,

$$\nabla_\theta \log p(\tau|\pi_\theta) = \sum_{t=0}^{\infty} \nabla_\theta \log \pi_\theta(u_t|x_t).$$

As the policy gradient estimate is independent of the state transition probability function and the initial state density function $p(x_0)$, only the sum of the gradient log-likelihood of the policy remains. In a more general context, the policy gradient can be formulated as

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathbb{E}}_{\tau\sim\pi_\theta}\left[\sum_{t=0}^{\infty} \Psi_t(\tau)\nabla_\theta \log \pi_\theta(u_t|x_t)\right], \tag{2.13}$$

where Ψ_t can take any of the following expressions [84]:

$$\Psi_t(\tau) = \begin{cases} R(\tau) & \text{trajectory return (2.7),} \\ \delta_t = r_t + \gamma V_\phi(x_{t+1}) - V_\phi(x_t) & \text{TD residual (2.11),} \\ Q_\phi(x_t, u_t) & \text{action-value function (2.10),} \\ A_\phi(x_t, u_t) = Q_\phi(x_t, u_t) - V_\phi(x_t) & \text{advantage function.} \end{cases}$$

Ψ_t is commonly set to A_ϕ for on-policy and to Q_ϕ for off-policy policy gradient methods. A drawback of the stochastic policy gradient formulation in (2.13) is that the objective requires data resampling because the data-generating distribution changes with each policy update step. This flaw can be made obvious when switching the viewpoint of the objective to the performance difference between two policies defined by

$$J(\pi) = J(\mu) + \int_x d^\pi(x) \int_u \pi(u|x) A_\mu(x, u)\, du\, dx,$$

where $d^\pi(x) = (1-\gamma)\,\mathbb{E}_{\tau\sim\pi}\left[\sum_{t=0}^{\infty} \gamma^t p(x_t = x) \mid x_0 = x\right]$ denotes the normalized weighted state distribution under policy π and A_μ is the advantage function under μ [76, 77, 85]. Since J is difficult to optimize due to its complex dependency on d^π, the local approximation

$$\hat{J}_\mu(\pi) = J(\mu) + \int_x d^\mu(x) \int_u \pi(u|x) A_\mu(x, u)\, du\, dx$$

is used instead to disconnect the optimization variable π from the data distribution d^μ [86]. The local approximation matches the original function to the first order [85], *i.e.*,

$$\hat{J}_{\pi_{\theta_k}}(\pi_{\theta_k}) = J(\pi_{\theta_k}) \quad \text{and} \quad \nabla_\theta \hat{J}_{\pi_{\theta_k}}(\pi_\theta)\,|_{\theta=\theta_k} = \nabla_\theta J(\pi_\theta)\,|_{\theta=\theta_k}\,.$$

In order to reuse generated trajectory data for multiple update steps, importance sampling is applied to perform updates based on the *surrogate objective*

$$\hat{J}_\mu(\pi_\theta) = J(\mu) + \mathop{\mathbb{E}}_{\tau\sim\mu}\left[\frac{\pi_\theta(u_t|x_t)}{\mu(u_t|x_t)} A_\mu(\tau)\right] = J(\mu) + \mathop{\mathbb{E}}_{\tau\sim\mu}\left[\lambda_t A_\mu(\tau)\right]. \tag{2.14}$$

The scalar $\lambda_t = \frac{\pi(u_t|x_t)}{\mu(u_t|x_t)}$ denotes the importance weighting of the samples reused for updates. Finally, by taking the derivative of the surrogate objective (2.14) with respect to θ, the importance-weighted policy gradient (IWPG) is obtained in its general form as

$$\nabla_\theta \hat{J}_\mu(\pi_\theta) = \mathop{\mathbb{E}}_{\tau\sim\mu}\left[\sum_{t=0}^{\infty} \lambda_t \Psi_t(\tau)\, \nabla_\theta \log \pi_\theta(u_t|x_t)\right]. \tag{2.15}$$

Policy gradient algorithms perform multiple update iterations based on the same data batch before generating a new one. Although the transition samples become off-policy after the first policy parameter update, the data distribution of the updated policy is still close to the policy that generated the data samples. The RL community refers to this modus operandi as on-policy learning, notwithstanding that samples become slightly off-policy after the first update.

Deterministic Policy Gradients. Different from the previously introduced stochastic class, *deterministic policy gradients*

$$\nabla_\theta J(\pi_\theta) = \nabla_\theta \mathop{\mathbb{E}}_{\tau\sim\mu}\left[Q_\phi\big(x, h(x,\theta)\big)\right], \tag{2.16}$$

use the deterministic policy output $h(x,\theta)$ and no likelihoods for computing the gradient. Since Q_ϕ is differentiable with respect to the actions, the policy parameters can be updated along the gradient of objective (2.13) by setting $\Psi_t = Q_\phi$. The benefit is that long-term critic estimates guide the gradient steps of the actor, and data samples from any generating policy μ can be leveraged for the policy updates. Updates are thus not required to incorporate the latest transition data produced by π_θ, which renders deterministic policy gradients a popular choice for off-policy algorithms [87, 88, 89].

2.4.4 Algorithm Overview

In this section, popular algorithms are introduced, and the inner workings of their policy update computation are outlined. The emphasis lies on the combination of an actor-critic with a policy gradient method, where updates are calculated iteratively based on the gradient estimate g_k, the step size α_k, and the approximation B_k to the Hessian. The latter two are chosen by the optimization method (or just the *optimizer*) in most RL algorithms, *e.g.*, Adam [90] and root mean squared propagation (RMSProp) [91] use adaptive schemes to approximate the Hessian and automatically tune the step size whereas stochastic gradient descent (SGD) uses $B_k = I$ and a fixed step size. However, the gradient computation differs between algorithms:

- IWPG resembles the starting point for all of the on-policy algorithms presented in this thesis. The policy gradient is given by $g_k = \nabla_\theta \hat{J}_\mu(\pi_\theta)\,|_{\theta=\theta_k}$ based on (2.15).
- Proximal policy optimization (PPO) [92] is an on-policy algorithm that utilizes clipped importance weighting to maintain the step size in the parameter space. The policy gradient is described by

$$\nabla_\theta \hat{J}_\mu^{\mathrm{PPO}}(\pi_\theta) = \nabla_\theta \mathop{\mathbb{E}}_{\tau\sim\mu}\left[\min\left(\lambda_t \Psi_t(\tau),\, \mathrm{clip}(\lambda_t,\, 1-\epsilon,\, 1+\epsilon)\Psi_t(\tau)\right)\right], \tag{2.17}$$

where λ_t is the importance weighting factor and ϵ is a hyperparameter. The clip operator projects the values of a variable x onto the set $[x_{\min}, x_{\max}]$.

- Deep deterministic policy gradient (DDPG) is an off-policy algorithm that applies deep Q-networks to continuous state-action spaces [88]. The policy gradient is given by $g_k = \nabla_\theta J(\pi_\theta) \mid_{\theta=\theta_k}$ (2.16), where the long-term prediction capabilities of the Q-function are exploited. The action-value function is iteratively updated with respect to the mean squared Bellman error based on (2.10). Additionally, DDPG utilizes target networks to stabilize the learning process of the actor and critic networks.

- Soft actor-critic (SAC) is an off-policy policy gradient algorithm [87, 93] that aims to solve the constrained optimization problem

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & J^{\text{SAC}}(\pi_\theta) = \mathop{\mathbb{E}}_{\tau\sim\mu}\left[\min_{i=1,2} Q_\phi^{(i)}(x, h(x,\theta))\right] \\ \text{subject to} \quad & \mathop{\mathbb{E}}_{\tau\sim\mu}\left[-\log \pi_\theta(u|x)\right] \geq \nu, \end{aligned}$$

  while regularizing the entropy $m(p) = \mathbb{E}_{x\sim p}\left[-\log p(x)\right]$ to be greater or equal to ν. Through Lagrangian relaxation, the constrained objective is converted into an unconstrained one by adding a Lagrangian penalty term to the one-step reward, *i.e.*, $r(x_t, u_t) + \beta m(\pi_\theta(u_t|x_t))$ with the dual variable β. SAC is based on the DDPG framework but uses double Q-networks as an important implementation enhancement to reduce value function over-estimation [89]. Finally, the policy gradient is given by

$$g_k = \nabla_\theta \mathop{\mathbb{E}}_{\tau\sim\mathbb{D}}\left[\min_{i=1,2} Q_\phi^{(i)}(x, h(x,\theta)) + \beta m(\pi_\theta(u|x))\right],$$

  where the minimization over the two action-value functions implements the double Q-network trick. In order to stabilize the training, target networks are utilized.

The algorithm of choice for most of the conducted research work in this thesis is PPO, primarily due to its learning stability, ease of implementation, and resilience toward the choice of hyperparameters. PPO has been found to be a ubiquitous on-policy algorithm that yields strong results across various problems [80, 94]. Although on-policy methods can require an order of magnitude more data samples than off-policy algorithms in robotics [25], the benefits outweigh the sample efficiency aspect when training in simulation.

2.5 Multi-Agent Reinforcement Learning

Major success stories with RL have been recorded with a single-agent learning framework that was applied to a multi-agent problem [95], treating agents as independent learners, *e.g.*, in board games [10, 17] or video games [18, 96, 97]. However, the multi-agent domain offers its own extensive framework with algorithms and tools to address high-dimensional and complex problems, as demonstrated in [16]. The content of this section is a brief summary of the background material presented in Paper V.

2.5.1 Framework

The Markov game, together with its partially observable derivatives, serves as the main framework for addressing multi-agent problems in the RL community. Markov games generalize MDPs to multiple decision-makers that interact simultaneously within a shared environment and possibly with each other [98].

Markov Game. The Markov game extends the MDP and is formalized by the tuple $(\mathbb{X}, \{\mathbb{U}_i\}, \mathbb{N}, p, \{r_i\}, \gamma)$, where $\mathbb{N} = [1, N]$ denotes the set of $N > 1$ interacting agents. $\mathbb{X}$ is the set of states observed by all agents. The joint action space is denoted by $\mathbb{U} = \mathbb{U}_1 \times \cdots \times \mathbb{U}_N$, which is the collection of individual action spaces $\mathbb{U}_i$ of agents $i \in \mathbb{N}$, and the transition probability kernel is given by $p : \mathbb{X} \times \mathbb{U} \to P(\mathbb{X})$. Each agent owns an associated reward function $r_i : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$, and $\gamma \in (0, 1)$ describes the discount factor.

Value Functions. Unlike the single-agent case, the value function $V_{\pi_i, \pi_{-i}} : \mathbb{X} \to \mathbb{R}$ depends on the actions taken under π_i and the actions selected by the other agents π_{-i}. Thus, the value function of agent i is given by

$$V_{\pi_i, \pi_{-i}}(x) = \mathop{\mathbb{E}}_{\tau \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t r_i(x_t, u_t) \;\middle|\; x_0 = x \right],$$

where all agents follow the joint policy $\pi : \mathbb{X} \to P(\mathbb{U})$ that is the collection of all individual policies $\pi = \{\pi_1, \ldots, \pi_N\}$. Further, the convention is used that $-i$ denotes all agents except agent i, resulting in the collection $\pi_{-i} = \{\pi_1, \ldots, \pi_{i-1}, \pi_{i+1}, \ldots, \pi_N\}$.

Optimality. Optimal behavior in Markov games is not solely determined by a single policy but the joint policy. Assuming for a moment that π_{-i} is fixed, agent i can only find the *best response* π_i^* to the other agents when maximizing

$$V_{\pi_i^*, \pi_{-i}}(x) \geq V_{\pi_i, \pi_{-i}}(x)$$

over all states $x \in \mathbb{X}$ and policies $\pi_i \in \Pi_i$. A solution where each π_i^* is the best response to π_{-i}^* is called *Nash equilibrium*, *i.e.*, the following inequality

$$V_{\pi_i^*, \pi_{-i}^*}(x) \;\geq\; V_{\pi_i, \pi_{-i}^*}(x)$$

holds true for all states $x \in \mathbb{X}$, policies $\pi_i \in \Pi_i$, and agents $i \in \mathbb{N}$. Intuitively spoken, a Nash equilibrium is a solution where agents cannot improve their own utility as long as the other agents' policies are stationary.

2.5.2 Additional Challenges with Multi-Agents

In the single-agent domain, the agent is the only entity that manipulates the environment state. Thus, state transitions can be unambiguously attributed to the agent, while all other processes causing state changes are regarded as part of the environment dynamics. Although the system is subject to aleatoric uncertainty, the underlying learning problem

remains stationary. On the contrary, the fundamental challenge in the multi-agent domain is that agents learn alongside each other and adapt their control policies simultaneously, resulting in the environment dynamics appearing non-stationary from a single agent's perspective. The Markov assumption no longer holds, and agents face, without further treatment, a moving target problem [20]. In the following, two important challenges that arise in MARL are presented.

Non-stationarity. From the perspective of a single agent, a moving target problem emerges when the environment dynamics

$$p\big(x'|x, \pi(x)\big) \neq p\big(x'|x, \tilde{\pi}(x)\big)$$

change due to the co-adaption of agents, *i.e.*, $\exists i \in \mathbb{N}$ such that $\pi_i \neq \tilde{\pi}_i$.

Partial Observations. Outside an idealized world, information is distributed among agents, which renders the environment not fully observable since agents cannot instantaneously access the internal knowledge state of other agents. Agents thus receive only a partial observation, which is modeled with individual observation spaces $\mathbb{O}_i$. In order to comprise relevant information about the environment and its history, the agents are commonly equipped with history-dependent policies $\pi_i : \mathbb{O}_i \times \cdots \times \mathbb{O}_i \to P(\mathbb{U}_i)$ that map from a history of observations to a distribution over the action space [99, 100]. The partially observable Markov game is formalized by the tuple $(\mathbb{X}, \mathbb{U}, \mathbb{N}, \mathbb{O}, p, \{r_i\}, \gamma)$, where $\mathbb{X}$ is the set of global but unobserved system states, and $\mathbb{O} = \mathbb{O}_1 \times \cdots \times \mathbb{O}_N$ is the collection of individual observation spaces.

Chapter 3

Research Questions and Hypotheses

This thesis covers the application of RL algorithms to real-world robotics. In this context, three RQs are formulated and answered accordingly by evaluating research hypotheses. Each RQ is dedicated to one of the three aspects: performance, robustness, or safety.

Research Question 1

Although most algorithms are conceptually straightforward to implement, a well-known issue in the RL community is that experimental results are difficult to reproduce [45]. Seemingly minor adjustments to the algorithm implementation can have a stark influence on the learning and the final policy performance [44]. As these findings are solely based on studies in simulation, it can be presumed that the sensitivity of RL algorithms is exacerbated in a sim-to-real transfer learning scenario, where the target domain differs from the domain on which the agent was trained. This gives rise to the following question:

RQ 1. *How do algorithm implementation and problem design choices affect policy performance in a zero-shot sim-to-real context?*

The first RQ is addressed by evaluating three hypotheses, which are related to the chosen algorithm components, the level of abstraction in the action space, and the policy representation.

Hypothesis 1.1. *State-of-the-art policy performance can be achieved by deploying a handful of carefully selected algorithm components.*

As recent algorithm improvements can be largely attributed to implementation details [44, 101], such improvements could be explained alternatively by an increased complexity in the algorithm design. The conjecture behind this hypothesis is that algorithms could be tailored to contemporary benchmark tasks, eventually overfitting to the benchmark specifications. High complexity in the algorithm design is undesirable for two reasons. On the one hand, overfitting reduces the generalization capabilities and thus could lead to diminishing policy performance on novel tasks. On the other hand, fewer algorithm components imply less hyperparameter tuning. Because sim-to-real transfer experiments usually require considerable testing in both simulation and the real world, a reduced algorithm design could accelerate experimentation cycles. Hypothesis 1.1 can be associated

with the minimum description length as an interpretation of the principle of Occam's Razor, which states the trade-off between the quality of data fitting and the complexity of the deployed algorithm model [102]. In that regard, the configuration with the least number of algorithm components that achieves good control performance is preferable.

Hypothesis 1.2. *Low-level control structures achieve superior policy performance compared to higher levels of control when the reality gap is sufficiently small.*

The intuition behind the performance superiority of low-level controllers is that low-level actions show the best reactivity to new situations by being able to directly drive actuators. In contrast, high-level actions represent abstract commands that guide subsequent controllers, thus being less reactive since motor commands must still be computed in the lower levels of the control loop. For instance, cascaded control based on proportional-integral-derivative (PID) controllers can be utilized on a robotic manipulator to transform an action from the task space into the respective motor instruction: an end-effector position displacement command is converted from the task space into torques in the joint space. If Hypothesis 1.2 is true, then low-level controllers will show worse zero-shot policy performance than high-level control structures when the reality gap is increased. The larger the model mismatch, the fewer real-world samples could resemble the data known from the simulation, resulting in a performance loss or transfer failure.

Hypothesis 1.3. *Recurrent actor-critics improve the policy performance upon feed-forward architectures when trained with domain randomization.*

A popular technique to improve the sim-to-real policy transfer is domain randomization (DR), which randomizes simulation parameters during training [31, 103]. Rather than training on a single system realization, the agent is exposed to a parameter distribution, which forces the agent to learn from a more diverse set of simulated data. FNN-based policies can exhibit conservative control behavior under DR since the mean trajectory return for all parameter realizations is optimized. This conservatism results in diminished policy performance compared to the training without DR [104] and the use of RNNs [105]. The superiority of RNNs can be explained by the encoding of environment characteristics in the latent memory [35, 106], which could also improve the transfer performance in a zero-shot setting. If Hypothesis 1.3 is correct, then RNNs will be superior to FNN architectures due to the capability to infer the robot's system realization from temporal data and adapt the control behavior accordingly, thus yielding improved policy performance.

Research Question 2

The reality gap is inherently inevitable since a simulation is just a simplified replica of the real world. Because the reality gap reduces the performance once the policy is deployed on the real robot, a desideratum is to synthesize policies that are robust toward the domain shift elicited by the transfer [107]. Discrepancies between the training and the deployment environment are caused, among other things, by an inaccurate simulation, partial knowledge about the environment, and phenomena that do not occur in simulation. Robustness is thus a vital requirement for successful sim-to-real policy transfers.

Although robustness has been investigated extensively in simulation [28, 108, 109, 110], studies in the sim-to-real context are sparse [111, 112]. This observation leads to the following question:

RQ 2. *What design choices impact the robustness of policies in zero-shot sim-to-real transfer learning?*

Although a plethora of factors could influence the robustness of a policy, RQ 2 is investigated from two perspectives: the implemented control structure and the NN representation.

Hypothesis 2.1. *High-level control structures are more robust toward the reality gap than low-level controllers.*

The first hypothesis addresses the level of abstraction in the action space. Low-level control requires the agent to learn the mapping from states to motor commands in an end-to-end fashion. Hence, low-level controllers must implicitly learn the underlying robot dynamics, while high-level control encourages the agent to develop a conceptual understanding of the robot and the learning task. High-level control structures could show improved robustness since the model mismatch between the simulation and the reality gap is concealed from the agent through fast stabilizing feedback in the low-level control loops, as argued in [113]. If the hypothesis is true, then high-level control structures will show a higher transfer success rate than low-level controllers when the simulation fidelity is reduced.

Hypothesis 2.2. *Policies based on recurrent neural networks are more robust than feed-forward architectures when trained with domain randomization.*

When trained with DR in simulation, recurrent actor-critic architectures learn to encode valuable information about the system's realization in the latent memory [35, 106]. It is hypothesized that RNN-based policies infer a parameter realization similar to one known from training from temporal data and adapt the policy to the characteristics of the real-world robot at deployment. However, it was found that RNNs are prone to overfit the simulation without DR [106]. If Hypothesis 2.2 is correct, then RNN-based policies trained without DR will produce worse sim-to-real transfer results than FNNs under enforced domain shifts due to overfitting.

Research Question 3

Agents explore the state space in a trial-and-error fashion, ideally with a highly stochastic and novelty-seeking strategy to discover large parts of the state space [114, 115]. However, the major issue of learning-based methods is that, without any further precautions, agents strive for unsafe actions in order to learn from failure [30]. While unsafe trajectories can be unhesitatingly generated in simulation without risking real consequences, safety mechanisms must be integrated into the learning loop of physical systems. The actions taken by a robot are supposed to satisfy the safety requirements at all times to avoid damage to the hardware or the robot's environment.

Despite the wide range of approaches that address safe exploration with RL-based systems [26, 27], RQ 3 is answered by utilizing a learning-based MPSC method that aims for safe closed-loop control. To this end, the method proposed in Paper IV, called ensemble model predictive safety certification (X-MPSC), is taken to investigate learning-based safety certification exemplarily. X-MPSC acts as a predictive filter that checks the safety of the agent's action in each time step through planning. Since learning-based MPSC cannot guarantee hard safety constraint satisfaction without strict assumptions, the following question arises:

RQ 3. *How can safety be improved in learning-based model predictive safety certification?*

Although learning-based MPSC methods exist, the expressiveness of the learning models applied in related work is limited, *e.g.*, linear regression [51] or linear Bayesian regression [52] on hand-selected nonlinear system features. On the contrary, X-MPSC deploys deep NNs as expressive learning representations. Eventually, RQ 3 is addressed by testing two hypotheses.

Hypothesis 3.1. *An ensemble of dynamics models decreases the number of safety constraint violations.*

A common problem with model-based RL is that NN-based dynamics models exhibit prediction inaccuracies. These inaccuracies grow over the predictive horizon, which limits the applicability of NNs to short rollouts and prevents their use for long-term planning [116]. Ensembles of NN models have been shown to address this issue [58, 117]. If Hypothesis 3.1 is correct, then the model reliability will increase with the number of models in the ensemble, which will reduce the total number of safety constraint violations that occur throughout training.

Hypothesis 3.2. *Adding a crude prior model to the learning representation lowers the number of safety constraint violations.*

Learning a dynamics model from scratch is possible, but it does not offer a rigorous safety certification during training. The use of a prior dynamics model is mandatory when dealing with strict safety certificates [118, 119]. In lieu of learning a model solely from interaction data, prior knowledge can be incorporated into the model. Most robot systems can be formulated by a system description that is derived from the first principles of physics. Although such a nominal model does not perfectly describe the real-world robot, the learning model only needs to capture the difference between the nominal model and the real system. If Hypothesis 3.2 is correct, then a nominal model will guide the training process of X-MPSC and will result in safer closed-loop control behavior than starting to learn tabula rasa.

Chapter 4

Methods

This chapter consolidates the methodology used to evaluate the research hypotheses and answer the RQs. Furthermore, this chapter also summarizes the methods applied in the research work of Papers I–V.

The methodology is divided into three sections, each linked to one RQ, respectively. Section 4.1 addresses RQ 1 and introduces the approach used to quantify the impact of algorithm components and the control problem design on policy performance. Section 4.2 examines RQ 2 by studying factors that could affect the robustness of policies in sim-to-real transfer learning. Third and last, RQ 3 is investigated based on an extended version of MPSC in Section 4.3. The improvement of learning-based safety certification is addressed by adding prior dynamics knowledge to the learning model and by using an ensemble of dynamics models. An overview of the methodology is depicted in Figure 4.1.

4.1 Algorithm and Control Problem Design

Despite the conceptually straightforward implementation of most algorithms, it is a widely known issue in the RL community that experiments are difficult to reproduce. Studies revealed that different code bases, although describing the same algorithm, produce divergent experimental results [45, 120]. This inconsistency suggests that subtleties in the implementation are decisive. In fact, it was found that seemingly minor implementation adjustments can be made accountable for performance improvements rather than algorithmic innovations [44, 101].

The policy performance eventuates from a complex entanglement between algorithm components and hyperparameters [45, 121]. Moreover, several decisions in the control problem design can have a significant impact on the learning, *e.g.*, the action space representation or the chosen features of the state space [122]. The study of the RL algorithms is further complicated by an abundance of tunable hyperparameters and evaluation protocols that are not fixed across the literature [123].

These observations raise the question of which underlying algorithm mechanisms and design choices drive the learning performance of RL algorithms. In order to address RQ 1, the methodology is divided into two parts. First, the impact of algorithm components is assessed. Therefore, the intricate complexity of on-policy policy gradient algorithms is disentangled by splitting the analysis into three stages: algorithm core components,

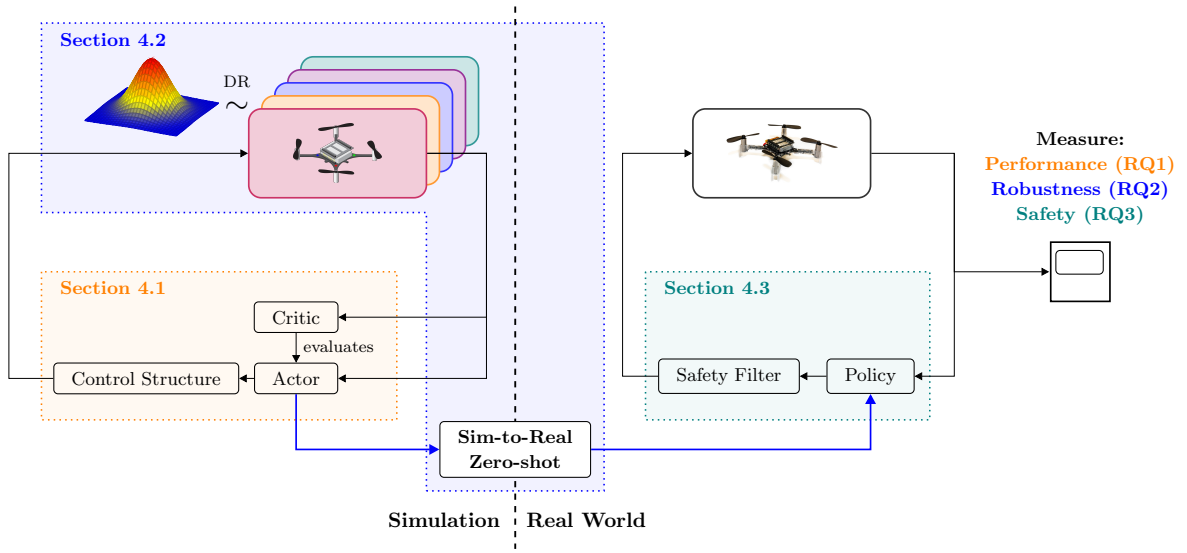


Figure 4.1: Overview of methodology. Section 4.1 explains the methods used to assess the impact of algorithm components and control design choices on policy performance. Section 4.2 addresses methods that aim to improve the robustness of policies in the zero-shot transfer learning context. Last but not least, Section 4.3 covers the approach used to assess safety with learning-based MPSC.

code-level enhancements, and structural learning components. Each stage is evaluated separately, and the analysis is conducted chronologically such that the influence of individual components becomes verifiable. The second part addresses RQ 1 by examining the impact of the control problem design on the policy performance based on different action space representations and policy architectures.

4.1.1 Algorithm Core Components

Algorithm core components lie at the heart of new algorithm proposals and aim to improve learning stability, speed, and convergence behavior. Common innovations in policy gradient algorithms are the ones that reduce the variance of the gradient estimation and improve the quality of the update direction. In this section, the following two components are examined due to their widespread adoption: trust-region enforcement and variance reduction of critic estimates. While described precisely from the theoretical point of view, the practical implementation often only becomes apparent from studying the provided source code.

Trust-Region Enforcement. Poorly chosen steps in the parameter space can deteriorate policy performance, which can eventually lead to policy collapse [77, 82]. Instabilities during training can be alleviated by constraining the distance between successive policy iterations through the enforcement of a trust region, which describes a careful selection of the step size α_k and the Hessian approximation B_k. The following four methods for constraining the policy update directions are studied:

1. IWPG, with an *early stopping criterion*, measures the KL divergence between consecutive policy iterations and terminates the update procedure as soon as a distance criterion δ is exceeded. More formally, the policy parameters are updated iteratively $\theta_{k+1}^{(j)} = \theta_{k+1}^{(j-1)} + \alpha_k p_k^{(j)}$ for $j \in [1, N_{\text{updates}}]$ as long as the distance criterion $D_{\text{KL}}\big(\pi_{\theta_{k+1}^{(j)}} \parallel \pi_{\theta_k}\big) < \delta$ given $\pi_{\theta_{k+1}^{(0)}} = \pi_{\theta_k}$ is satisfied. The direction $p_k^{(j)}$ is based on $B_k^{(j)}$ and $\alpha_k^{(j)}$, and g_k is given by (2.15).

2. Natural policy gradient (NPG) regards the *curvature* of the policy parameter space by approximating the Fisher information matrix $H_k = \mathbb{E}_{\tau\sim\pi}\left[\psi\psi^T\right]$ using $\psi = \nabla_\theta \log \pi_\theta(u_t|x_t)$ [124]. The update direction is given by $p_k = B_k^{-1} g_k$, where the policy gradient $g_k = \nabla_\theta \hat{J}_{\pi_\theta}(\pi_\theta)\,|_{\theta=\theta_k}$ is based on (2.15) and $B_k = H_k$. Since curvature information is included in the update direction, the step size is set to $\alpha = 1$.

3. Trust-region policy optimization (TRPO) [86] optimizes the surrogate objective (2.14) and extends the NPG algorithm with a *trust region*. In contrast to NPG, TRPO explicitly regards the KL divergence between policy iterations by evaluating the objective function and running a backtracking line search. For this purpose, the step size $\alpha \in (0, 1)$ is exponentiated with $j \in [1, N_{\text{steps}}]$, and the objective is optimized until the criterion $D_{\text{KL}}\big(\pi_{\theta_{k+1}^{(j)}} \parallel \pi_{\theta_k}\big) < \delta$ with $\theta_{k+1}^{(j)} = \theta_k + \alpha^j B_k^{-1} g_k$ is fulfilled. The inverse of the approximate Hessian is calculated by $B_k^{-1} = \sqrt{\frac{2\delta}{g_k^T H_k^{-1} g_k}} H_k^{-1}$, where H_k denotes the Fisher information matrix and g_k is the policy gradient. The size of the trust region is determined by the hyperparameter δ.

4. PPO [92] utilizes *clipped probability ratios* to maintain the step size in the parameter space. The policy gradient (2.17) is applied iteratively for stochastic gradient ascent. Different from TRPO, the step size α_k and the Hessian approximation B_k are chosen by an optimizer like Adam or RMSProp.

Variance Reduction of Critic Estimates. Low variance gradient estimates are critical for good policy performance [77, 125]. Thus, three techniques that are used to reduce the variance of policy gradients are compared with each other:

1. *Plain advantage estimation* reduces the variance of the gradient estimation by utilizing bootstrapping. The variance is reduced by estimating the advantage

$$A_t = r(x_t, u_t) + \gamma V_\phi(x_{t+1}) - V_\phi(x_t) = \delta_t,$$

   by subtracting the baseline V_ϕ from the bootstrapped action value, which equals the TD residual (2.11). Plain advantage estimation is a one-step method, which is used as a performance baseline for the following n-step methods.

2. *Generalized advantage estimation (GAE)* [84] aims to reduce the variance of gradient estimates at the cost of introducing bias. The n-step advantage

$$A_t^{(n)} = -V_\phi(x_t) + r_t + \gamma r_{t+1} + \cdots + \gamma^{n-1} r_{t+n-1} + \gamma^n V_\phi(x_{t+n}) = \sum_{k=0}^{n-1} \gamma^k \delta_{t+k}$$

is exponentially weighted by a scalar factor κ and taken as the sum over the infinite horizon, resulting in the generalized advantage estimator

$$A_t^{\mathrm{GAE}} = (1-\kappa)\Big(A_t^{(1)} + \kappa A_t^{(2)} + \kappa^2 A_t^{(3)} + \dots\Big) = \sum_{k=0}^{\infty} (\gamma\kappa)^k \delta_{t+k}$$

at time step t. Note that the notation r_t is used instead of $r(x_t, u_t)$ to improve readability. The parameters γ and κ contribute a trade-off between variance and bias in the estimation. The discount factor γ reduces the variance but induces bias in the policy gradient estimation, while κ regulates only the induced bias.

3. *V-trace* [126] accounts for off-policy critic updates in distributed architectures and compensates for policy lags. To reduce variance in gradient estimates, advantages

$$A_t^{\mathrm{VTR}} = r_t + \gamma \tilde{V}_\phi(x_{t+1}) - V_\phi(x_t)$$

are computed based on

$$\tilde{V}_\phi(x_t) = V_\phi(x_t) + \sum_{k=0}^{n-1} \gamma^k \left(\prod_{i=0}^{k-1} c_{t+i}\right) \rho_{t+k}\delta_{t+k},$$

where δ_{t+k} is the TD residual. The scalar terms $\rho_{t+k} = \min\left(\tilde{\rho}, \frac{\pi(u_{t+k}|x_{t+k})}{\mu(u_{t+k}|x_{t+k})}\right)$ and $c_{t+i} = \min\left(\tilde{c}, \frac{\pi(u_{t+i}|x_{t+i})}{\mu(u_{t+i}|x_{t+i})}\right)$ are truncated importance sampling weights, where the hyperparameters $\tilde{\rho}$ and $\tilde{c}$ are chosen such that $\tilde{\rho} \geq \tilde{c}$.

4.1.2 Code-Level Enhancements

The recent progress in policy gradient algorithms has been backed up by effective implementation enhancements such as double Q-networks and target networks in the off-policy setting [89, 127]. Similarly, with on-policy gradients, studies reported that a major share of performance increments can be attributed to implementation decisions rather than innovative algorithmic properties [44, 101].

This section is dedicated to the investigation of such decisions in the code implementation. Code-level enhancements denote augmentations to the algorithmic core that aim to improve learning quality through better stability or faster learning speed. As indicated by the name, code-level enhancements are typically not mentioned in the main paper but are specified in the code implementation or provided as detail in the supplemental materials. Over the last few years, many of the enhancements described in this section have been established as good practices in the RL community and are, thus, implemented without further consideration. Moreover, code-level enhancements are typically not included in the hyperparameter search, obscuring their effect on performance. The following code-level enhancements are examined for on-policy policy gradients as they are applied most frequently [44, 121].

Standardization. Standardization is commonly applied to observations and advantage estimates. The quantity of interest y is made mean-free and re-scaled to unit variance by subtracting the mean value first and then dividing by the standard deviation, *i.e.*,

$$\tilde{y} = \frac{y - \text{mean}(y)}{\text{std}(y)}.$$

While observation standardization typically considers all collected samples since the start of training by computing running statistics, it can be beneficial to standardize the advantage estimates only based on the latest data batch [121]. The latter can be seen as an instance of batch normalization [128].

Scaling. The received rewards are divided by the standard deviation of a running discounted sum of rewards, resulting in return-scaled rewards

$$\tilde{r}_t = \frac{r(x_t, u_t)}{\text{std}(R(\tau))}.$$

Rewards are re-scaled but not shifted to avoid the creation of a moving target problem.

Entropy Bonus Term. Actions with high stochasticity promote the exploration of the state space. An effective technique to encourage exploration is adding an entropy bonus term m to the optimization objective [129], *i.e.*,

$$\hat{J}(\pi_\theta) + \beta m(\pi_\theta(u_t|x_t)).$$

The hyperparameter β is either fixed or can be tuned automatically through the optimization of a Lagrangian objective and dual gradient descent to ensure a minimum expected entropy throughout the training [93].

Annealing. When using an annealing strategy, the hyperparameter of interest is decreased throughout the training. In this thesis, only linear annealing schedules are considered and applied to two different hyperparameters. Learning rate annealing is used to decay the learning rate of the policy optimizer toward zero over the training. The linear decay

$$\alpha_k = \left(1 - \frac{k-1}{N_{\text{epochs}}}\right) \alpha_0$$

is applied with α_0 being the initial learning rate and $k \in [1, N_{\text{epochs}}]$ being the training epoch. The second investigated strategy is exploration noise annealing. Good exploration is risk-seeking and prefers novelty over experience, especially during the early phase of the training. In order to have high exploration at the beginning and low exploration toward the end of the training, the entries of the diagonal covariance matrix $S = \epsilon_k \,\text{diag}(s_1, \ldots, s_{n_u})$ are decreased for a stochastic policy (2.12). The diagonal entries $s_1, \ldots, s_{n_u}$ denote the initial exploration noise factor for each action dimension and are changed according to the linear decay schedule

$$\epsilon_k = \left(1 - \frac{k-1}{N_{\text{epochs}}}\right)$$

over $k \in [1, N_{\text{epochs}}]$ epochs.

Gradient Clipping. As pointed out in Section 4.1.1, poorly chosen policy parameter steps can deteriorate the policy performance. An effective technique to control the step size is limiting the maximum norm of the policy gradient. The gradient is re-scaled to length ζ if exceeding a threshold, *i.e.*,

$$\nabla_\theta J(\pi_\theta) = \begin{cases} \zeta \frac{\nabla_\theta J(\pi_\theta)}{\|\nabla J(\pi_\theta)\|_2} & \text{if } \|\nabla_\theta J(\pi_\theta)\|_2 > \zeta, \\ \nabla_\theta J(\pi_\theta) & \text{else.} \end{cases}$$

Through clipping, the magnitude of the gradients can be maintained, which can eventually help to avoid policy collapse.

4.1.3 Structural Learning Components

Structural learning components refer to design choices and hyperparameters affecting the NN-based policy architecture and the optimization procedure. The following three components are examined since their implementation can vary between different software frameworks and often lack a detailed description of the experimentation.

Optimizer. The optimizer describes the algorithm that selects the parameter update direction for optimization. The various flavors of an optimizer are noticeable in how α_k and B_k are chosen. Three optimizers are considered: Adam [90], RMSProp [91], and SGD without momentum. Besides the actual choice of the optimizer, the impact of an additive denominator term called *Adam epsilon*, which is added to the denominator of the update step to improve the numerical stability of the optimizer, is investigated.

Parameter Sharing. When actor and critic networks are realized with the same NN architecture, layers of the learning representation can be shared between both. The intention of parameter sharing is to accelerate the learning since features in the latent representation used for estimating the long-term value of a state might also be meaningful for selecting an action in this state.

Parameter Initialization. An essential procedure prior to the training is the initialization of the policy parameters, as it affects the exploration during the initial phase of learning [121]. The following three initialization schemes are examined: Kaiming uniform [130], Glorot [131], and orthogonal [132]. At initialization, only the NN parameters of the weight matrices are set randomly. The bias vectors are not affected by the initialization scheme and are set to zero values.

4.1.4 Control Structures

Related works demonstrated that decision-making in high-dimensional state-action spaces requires good representations and abstraction levels [133, 134]. Although NNs are expressive models that learn features from relationships in data automatically, the representations of the state and action space are engineered prior to the training. These

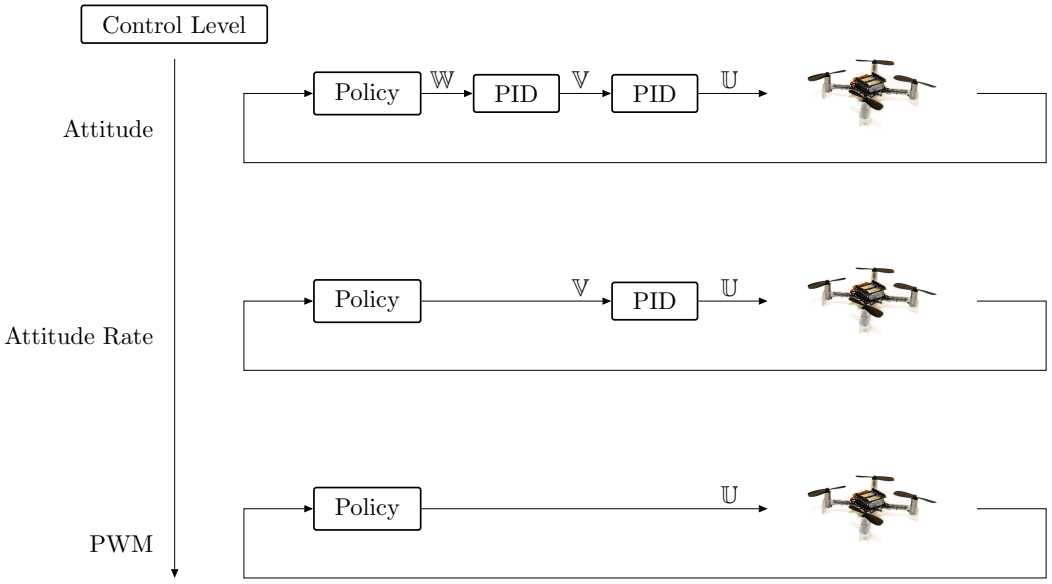


Figure 4.2: Control structures applied to a quadrotor robot.

design decisions can affect both the learning speed and the resulting policy performance [122, 135, 136, 137]. This section introduces different levels of abstraction in the action space, so-called *control structures*.

When the action space representation is implemented for low-level control, then actions directly drive the actuators of a robot. Low-level controllers learn to map from states to motor commands in an end-to-end fashion. In contrast, a high-level control structure maps from states to an intermediate action space. Subsequent controllers transform the abstract action from the intermediate action space into actual motor commands. For the remainder of this thesis, a quadrotor robot is used as a running example and hardware platform to implement control structures. The following three control structures, ordered from low-level to high-level, are investigated:

1. In *pulse-width modulated (PWM)* control, policies output motor thrust commands and map from states to distributions over the action space, *i.e.*, $\pi : \mathbb{X} \to P(\mathbb{U})$.

2. *Attitude-rate* control produces actions that lie in the intermediate action space $\mathbb{V} \subseteq \mathbb{R}^4$, which consists of the mass-normalized collective thrust and the desired body angle rate. The transformation from $\mathbb{V}$ to $\mathbb{U}$ is carried out by a PID controller that calculates the thrust commands $u \in \mathbb{U}$ based on local feedback from the measured linear acceleration and the error between the actual and the desired body angle rate. Agents learn the mapping $\pi : \mathbb{X} \to P(\mathbb{V})$.

3. *Attitude control* requires policies to learn the map $\pi : \mathbb{X} \to P(\mathbb{W})$, where the policy output space $\mathbb{W} \subseteq \mathbb{R}^4$ includes the mass-normalized collective thrust and the desired body angle. Thereafter, the sequence of mappings $\mathbb{W} \to \mathbb{V} \to \mathbb{U}$ is handled by cascaded PID controllers.

The three control structures are illustrated in Figure 4.2.

4.1.5 Policy Architecture

On real hardware systems, the Markov property is often violated because neither the environment is fully observable nor the complete state of the robot can be measured directly. State estimators are a common approach for computing an estimate of the current robot state based on a dynamics model and sensor fusion. Besides designing a state estimator, partial observability can be tackled by selecting a policy architecture capable of reconstructing missing state information from past interaction data [69]. Two implementations for history-dependent policies are considered in this thesis: (i) the stacking of the most recent observations into one vector and feeding into an MLP and (ii) the utilization of an RNN that processes observations sequentially.

4.1.6 Addressing the Performance Hypotheses

The assessment of Hypothesis 1.1 is challenging for two reasons. First, algorithm components can cross-correlate such that a component unfolds its effectiveness only when combined with another algorithm component. Cross-correlations can be alleviated by adding components randomly in each training run so that the impact is assessed based on the average performance difference, whether the component was added or excluded. Second, the combinatorial complexity grows exponentially with the number of algorithm components. The combinatorial complexity can be addressed by splitting the experimental evaluation into individual stages that are examined chronologically.

Hypothesis 1.1 was tested empirically in Paper I. Based on IWPG (2.15), code-level enhancements (Section 4.1.2) were added randomly to each training run with a probability of 50 % to remedy cross-correlations between different components. After that, the experimental evaluation succeeded chronologically by endowing the optimization objective with algorithm core components (Section 4.1.1). A grid search over the learning rates and the number of policy iterations was conducted to determine the performance impact of the best component configuration. In the third and last stage, structural learning components were assessed by applying a hyperparameter grid search and measuring the final policy performance (Section 4.1.3). Based on the three-staged analysis, the components with the largest performance gains were selected and assembled to a minimal configuration that was compared to the algorithm implementations of TRPO and PPO [138]. The comparison was carried out in simulation environments, which consisted of five locomotion and three robotic manipulation tasks. These eight tasks are common benchmark representatives for continuous control problems [139].

The evaluation of Hypothesis 1.2 is based on the comparison of the three control structures introduced in Section 4.1.4. The experimental evaluation was conducted in Paper II through real-world experiments on a CrazyFlie drone, where the zero-shot performance was measured in terms of the mean flight time of the drone. In order to render the reality gap as small as possible, the simulation parameters were tuned with simulation optimization.

Hypothesis 1.3 was tested through zero-shot transfers to a CrazyFlie drone in Paper III. Similar to the previous hypothesis, the performance was evaluated in terms of the mean flight time and the success rate, which is defined as the number of trials without failure

relative to the number of total flights. The history-dependency was realized with the two policy architectures from Section 4.1.5 that were compared to each other.

4.2 Zero-Shot Policy Transfer and Robustness

Transfer learning describes the idea of reusing experience collected in learning to accomplish one task in a related but different task [140]. Since sample complexity is a major concern in RL, sim-to-real transfer learning is a common methodological approach to alleviate the cost of real-world data collection when dealing with robotics [107]. Control policies are trained in a simulation that runs multiple times faster than real-time to gather large amounts of training data. After training on simulated data, the policies are deployed on the physical robot. Zero-shot transfer methods are an approach to elude the real-world sampling issue completely since policies are trained entirely on simulated data, and no real-world data is used for fine-tuning the controller.

Although the simulation shares similarities with the real-world system in the underlying dynamics and the downstream tasks, the actual realization and the data distributions differ due to the model mismatch. In the remainder of this section, two approaches that are used in the literature to mitigate the reality gap are introduced, namely DR and simulation optimization. To address RQ 2, the evaluation protocol and methods, which are used to assess the robustness of policies toward the reality gap, are described.

4.2.1 Domain Randomization

The idea behind DR is to randomize the simulation parameters during the training. By exposing the control policy to a parameter distribution rather than a single system realization, the policy is expected to generalize across the realizations and, thus, become more robust toward the reality gap. DR can be seen as a data augmentation technique in which the diversity of the simulated data is enhanced. Possible targets of the randomization are parameters affecting the visual appearance of the scene [103, 141] and the robot dynamics [105, 142]. Throughout this thesis, the latter is applied, and the terms domain and dynamics randomization are used interchangeably.

More formally, the system parameters ξ are sampled from a uniform distribution Ξ at the beginning of each episode. This results in the state transition probability being parametrized by the system parameters under DR, *i.e.*, $x_{t+1} \sim p(\cdot|x_t, u_t, \xi)$. Thus, policy search aims to optimize the objective

$$\underset{\theta}{\text{maximize}} \quad J(\pi_\theta) = \mathbb{E}_{\xi\sim\Xi,\tau\sim\pi_\theta}\left[R(\tau)\right]$$

over the dynamics induced by the distribution of simulation parameters. A larger number of randomized parameters causes less cumulative reward during the training but increases the success rate of the transfers to the real robot [35, 105].

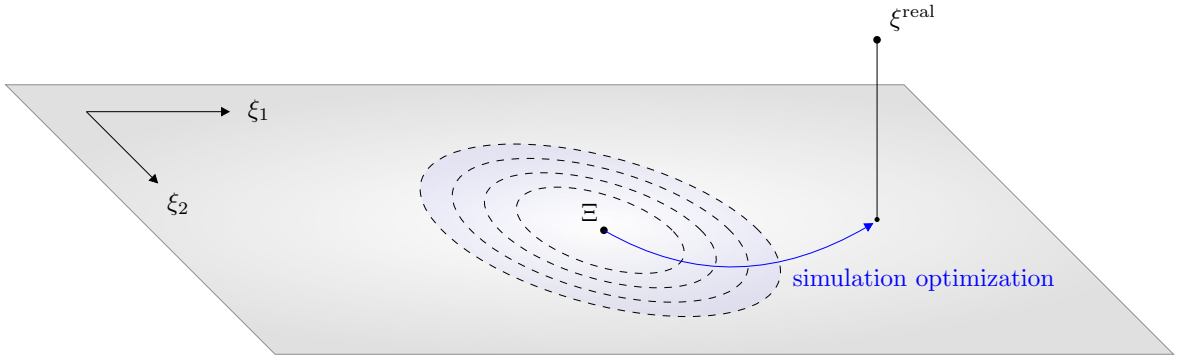


Figure 4.3: Abstract illustration of the simulation optimization approach. Since simulations are simplified replicas of the real world, the simulation parameter distribution Ξ (illustrated as light blue disk) is embedded in a plane while the realization of the robot ξ^{real} lies in the three-dimensional space. The goal of simulation optimization is to minimize the distance between the system parameter distribution induced by DR and the realization of the real-world system.

4.2.2 Simulation Optimization

Another approach to mitigate the reality gap is simulation optimization, which uses real-world data to reduce the gap between the simulation and the physical robot [143]. A great benefit of this technique is that the simulation parameters are tuned data-driven, which requires less engineering when offline data are available or when data can be collected effortlessly from a real-world system, *e.g.*, by a human operator or a controller with low task performance. An illustration of simulation optimization and DR is shown in Figure 4.3.

Given a real-world trajectory $\tau = (x_0, u_0, \ldots, u_{T-1}, x_T)$, a rollout can be simulated with respect to the action sequence $(u_0, \ldots, u_{T-1})$ and the dynamics model

$$z_{t+1} = f_{\text{sim}}(z_t, u_t, \xi) \quad \forall t \in [0, T-1],$$

where the initial state of the simulation is set to the first measurement of the real-world data, *i.e.*, $z_0 = x_0$. The simulated trajectories $\tilde{\tau} = (z_0, u_0, \ldots, u_{T-1}, z_T)$ depend on ξ, which enables the optimization over the parameter space Ξ. In order to align the simulation trajectories with the measured data of the real system, the objective function is formulated as a weighted sum of state differences, *i.e.*,

$$\underset{\xi}{\text{minimize}} \; \underset{\tilde{\tau} \sim f_{\text{sim}}}{\mathbb{E}} \left[\sum_{t=0}^{T} \eta^t \Big(\big\| C\big(z_t - x_t\big) \big\|_1 + \big\| C\big(z_t - x_t\big) \big\|_2 \Big) \;\Big|\; z_0 = x_0 \right]. \tag{4.1}$$

With a slight abuse of notation, $\tilde{\tau} \sim f_{\text{sim}}$ denotes the trajectories generated according to the dynamics model f_{sim} and the action sequence $(u_0, \ldots, u_{T-1})$ over trajectories of length T. The hyperparameter $\eta \in (0, 1]$ regulates the accuracy trade-off between one-step and long-term model prediction error. A small η discounts later stages of the rollout, in which the discrepancies between the simulation and the real-world trajectory grow. State differences $z_t - x_t$ are weighted according to matrix C to include preferences, *e.g.*,

emphasize model accuracy in the angular speed rather than linear velocity. Similar to Chebotar et al. [144], the objective (4.1) uses the sum of L1 and L2 norm.

The simulation optimization problem can have long evaluation times and may not be differentiable depending on the physics simulator and the randomized simulation parameters. In Paper II, *Bayesian optimization* was used to minimize (4.1) as it is particularly suited for objective functions with long evaluation times. Moreover, Bayesian optimization is regarded as a good option for optimization in continuous domains with less than 20 dimensions and is robust against stochastic function evaluations [145].

4.2.3 Evaluation of the Robustness Hypotheses

The evaluation of Hypotheses 2.1 and 2.2 is based on the assumption that the simulation parameters are optimized and a change of simulation parameters results in a magnification of the reality gap. The robustness of different policies was tested in Papers II and III, where the experiments were carried out on a CrazyFlie drone robot. The robustness of the policies was tested by manipulating actuator parameters such as the motor lag and latency. Changing parameter values in simulation enforces a distributional shift through an increased reality gap. The zero-shot performance on the real robot was measured by two metrics: (i) the cumulative reward that is identical for simulated and real-world tasks and (ii) the success rate that relates the number of failure-free trials (*e.g.*, crashing the drone or violating body angle constraints) to the total number of flights.

Hypothesis 2.1 states that high-level control structures are more robust toward the reality gap than low-level control structures. Paper II examines this hypothesis by enforcing a distributional shift until the policy transfer fails. The policy performance is recorded based on the mean flight time.

Hypothesis 2.2 claims that RNNs are more robust toward the reality gap than FNNs with observation-history inputs. The hypothesis is approached in Paper III by removing the latency modeling from the training and measuring the transfer success rate.

4.3 Safe Exploration

In the artificial intelligence community, different concepts exist for defining the safety of a learning-based system [30, 146, 147]. A tailored definition for RL describes safe learning as the process of being able to optimize a performance measure while satisfying constraints during training and at deployment [27]. Constraints in the state space are typically introduced to prevent so-called error states, which are irreversible points in the state space from which the original state of the system cannot be restored. For instance, a quadrotor robot that lies bottom-side up on the ground cannot recover an upright position by spinning its rotors. Error states are closely linked with the idea of ergodicity in MDPs [148] and forward invariant sets [149]. This thesis adopts the safety definition with respect to constraint satisfaction and refers to task-specific and system-imposed restrictions in the state and action space as safety constraints.

This section explains two algorithm classes that address safe exploration. First, *constrained RL* algorithms, which achieve excellent control performance, are introduced.

However, good empirical performance comes at the expense of only being able to satisfy the safety constraints upon algorithm convergence [150]. This algorithm class can be applied to high-dimensional state-action spaces when safety violations are tolerable during training and when the enforcement of safety constraints is required only at the end of the training [151]. Second, predictive safety filters are presented that exhibit more rigorous safety satisfaction than constrained RL algorithms. In particular, X-MPSC is introduced as a safety filter method that combines robust MPC with model-based deep RL to correct potentially unsafe actions taken by a learning agent. X-MPSC is the main subject for answering RQ 3, while constrained RL algorithms are used as a comparison baseline in the experiments.

4.3.1 Constrained Reinforcement Learning

The standard formalism for constrained RL is the constrained Markov decision process (CMDP), which incentivizes policies to maximize task performance while producing expected costs less or equal to a predefined safety threshold [152]. Costs are commonly measured by the accumulation of state constraint violations along a trajectory. As a result, constrained RL algorithms merely encourage safety by aiming for constraint satisfaction at the end of training and regretting constraint violations during training [26].

The CMDP adds an auxiliary cost function $c : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$ to an MDP [153]. Analogous to the expected return (2.8), the cumulative cost

$$H(\pi_\theta) = \mathop{\mathbb{E}}_{\tau \sim \pi_\theta} \left[\sum_{t=0}^{N} c(x_t, u_t) \right]$$

depends on the policy π_θ and accounts for long-term constraint satisfaction. Therefore, the objective function in CMDPs is given by

$$\begin{aligned} \underset{\theta}{\text{maximize}} \quad & J(\pi_\theta) \\ \text{subject to} \quad & H(\pi_\theta) \leq d, \end{aligned} \tag{4.2}$$

where $d \in \mathbb{R}$ is a task-specific cost limit that must be satisfied to provide the desired safety level. Policy parameter vectors are thus restricted to the set of feasible policies $\{\pi_\theta \in \Pi_\theta \mid H(\pi_\theta) \leq d\}$.

4.3.2 Ensemble Model Predictive Safety Certification

Ensemble model predictive safety certification (X-MPSC) belongs to the class of *predictive safety filters*. Originating from the control theory community, predictive safety filters leverage a prior dynamics model to predict the future evolution of the actual system over a receding horizon [154]. Through optimization, an action sequence is found that generates a nominal trajectory satisfying the safety constraints and leading the system back to a terminal control invariant set. From the terminal set, the system can be kept safe for all future time steps. Predictive safety filters can be integrated into any learning-based control system, as depicted in Figure 4.1.

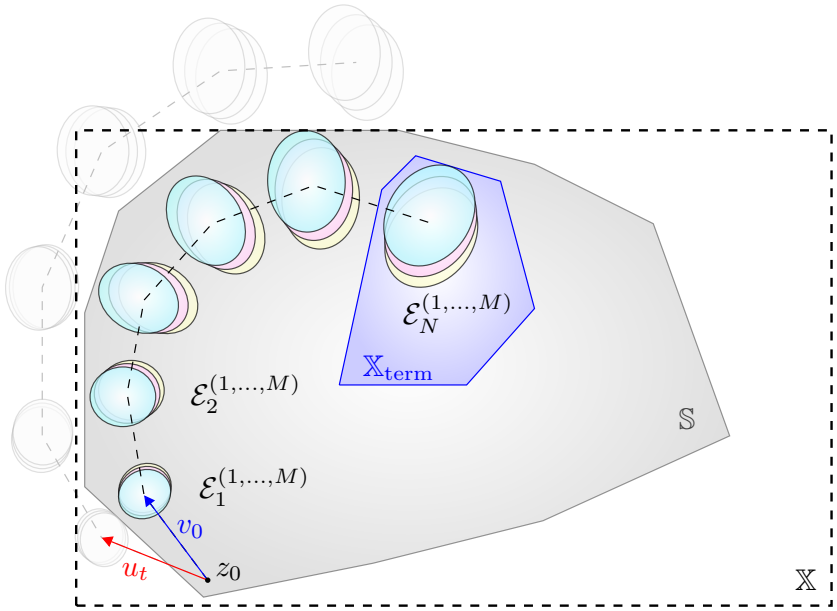


Figure 4.4: X-MPSC utilizes multiple NNs and multi-step planning to propagate ellipsoidal uncertainty estimates forward. An action is certified when all predictive tubes lie within the state space constraints $\mathbb{X}$ and end in the terminal set $\mathbb{X}_{\text{term}}$. Otherwise, the unsafe action u_t (red) is corrected to v_0 (blue), which keeps the system within the specified safety constraints.

Nominal MPSC (2.5) prevents a learning-based system from entering unsafe regions in the state space by modifying input actions in a minimally invasive fashion while enforcing safety constraints. However, the MPSC framework has the demerit that a nominal model is required a priori, which has to be sufficiently calibrated in order to provide accurate predictions. To account for the deviations between the nominal model and the actual system, robust MPC is integrated into MPSC to capture the uncertainty through tube-base predictions [50]. The expansion of MPSC with a data-driven model follows seamlessly, either by learning a model from scratch or by learning an additive component to the nominal model [51, 52].

In this section, X-MPSC is introduced as an extension to MPSC, which leverages an ensemble of probabilistic NNs trained on past interaction data. Each member of the ensemble is used for multi-step look-ahead rollouts, where the predicted uncertainty tube of each NN enforces the optimization constraints. The methodological approach of X-MPSC is illustrated in Figure 4.4.

Model Ensemble. The ensemble $\tilde{f}_\phi = (f_{\phi_1}, \dots, f_{\phi_M})$ is composed of M models, where each model is represented by a probabilistic NN with parameters ϕ_i. Each dynamics model $f_{\phi_i} : \mathbb{X} \times \mathbb{U} \to P(\mathbb{X})$ parameterizes a Gaussian probability distribution function

$$f_{\phi_i}(x_t, u_t) = \mathcal{N}\big(m_{\phi_i}(x, u), S_{\phi_i}(x, u)\big)$$

with the mean m_{ϕ_i} and the diagonal covariance matrix S_{ϕ_i}. The ensemble of dynamics models aims to approximate the actual system (2.1) and is trained by optimizing the

maximum likelihood over sampled trajectory data. The parametrization with a Gaussian allows the alternative formulation of the uncertainty as a level set expressed as an ellipsoid

$$\mathcal{E}(c, S) = \left\{ x \mid (x - c)^T S^{-1} (x - c) \leq 1 \right\}.$$

Ellipsoids are propagated over multiple time steps and are used to describe the uncertainty of the NN predictions. The propagated ellipsoids are assumed to construct an uncertainty tube that captures all realizations of the actual system.

Uncertainty Propagation. The ellipsoidal uncertainty predictions are used for multi-step look-ahead planning. The nominal trajectory $\left(z_0^{(i)}, \ldots, z_N^{(i)}\right)$ is based on the action sequence $(v_0, \ldots, v_{N-1})$, where the nominal states $z_{k+1}^{(i)} = m_{\phi_i}\left(z_k^{(i)}, v_k\right)$ are computed for each model i and for all stages $k \in [0, N-1]$.

The forward propagation of the uncertainty sets is implemented with a one-step error prediction. By using the first-order Taylor-series expansion, the next state can be approximated by

$$x_{t+k+1} \approx m_{\phi_i}\left(z_k^{(i)}, v_k\right) + A_k\left(x_{t+k} - z_k^{(i)}\right) + B_k(u_{t+k} - v_k)$$

around the fixed point $\left(z_k^{(i)}, v_k\right)$, where the Jacobians $A_k = \nabla_x m_{\phi_i}(x, u)^T \Big|_{x=z_k^{(i)}, u=v_k}$ and $B_k = \nabla_u m_{\phi_i}(x, u)^T \Big|_{x=z_k^{(i)}, u=v_k}$ are used. The prediction error is given by the expression $e_k^{(i)} = x_{t+k} - z_k^{(i)}$ and approximately satisfies the error difference equation

$$\begin{aligned} e_{k+1}^{(i)} &\approx A_k\left(x_{t+k} - z_k^{(i)}\right) + B_k(u_{t+k} - v_k) \\ &= A_k\left(x_{t+k} - z_k^{(i)}\right) + B_k K\left(x_{t+k} - z_k^{(i)}\right) \\ &= (A_k + B_k K)\left(x_{t+k} - z_k^{(i)}\right) \\ &= F_k e_k^{(i)} \end{aligned}$$

to the first order. The matrix $F_k = A_k + B_k K$ describes the closed-loop error system induced by the local feedback $u_{t+k} = v_k + K\left(x_{t+k} - z_k^{(i)}\right)$ based on (2.4). Finally, the one-step ellipsoidal uncertainty propagation for model i is given by

$$\mathcal{E}_{k+1}^{(i)} = g_{\phi_i}\left(\mathcal{E}_k^{(i)}, v_k\right),$$

which is propagated forward at each stage $k \in [0, N-1]$ based on the nonlinear map

$$g_{\phi_i}\left(\mathcal{E}_k^{(i)}, v_k\right) = \mathcal{E}\left(m_{\phi_i}\left(z_k^{(i)}, v_k\right), F_k S_k F_k^T\right) \oplus \mathcal{E}\left(0, S_{\phi_i}\left(z_k^{(i)}, v_k\right)\right).$$

The evolution of each ellipsoid only depends on the action sequence $(v_0, \ldots, v_{N-1})$ and the NN parameters ϕ_i.

Optimization Problem. The aim of X-MPSC is to certify the action u_t of a learning agent in each time step t of the RL loop by solving the optimization problem

$$\begin{aligned}
\underset{v_0,\dots,v_{N-1}}{\text{minimize}} \quad & \|u_t - v_0\|_2^2 \\
\text{subject to} \quad & \mathcal{E}_0^{(i)} = \mathcal{E}(x_t, 0) && \forall\, i, \\
& \mathcal{E}_{k+1}^{(i)} = g_{\phi_i}\big(\mathcal{E}_k^{(i)}, v_k\big) && \forall\, i, k \in [0, N-1], \\
& \mathcal{E}_k^{(i)} \subseteq \mathbb{X} && \forall\, i, k \in [0, N], \\
& v_k \in \tilde{\mathbb{U}}\big(\mathcal{E}_k^{(i)}\big) && \forall\, i, k \in [0, N-1], \\
& \mathcal{E}_N^{(i)} \subseteq \mathbb{X}_{\text{term}} && \forall\, i.
\end{aligned} \tag{4.3}$$

The objective is solved in a receding horizon fashion and v_0 is a safe action that is as close as possible to the agent's original action u_t. In order to maintain recursive feasibility and safety in future time steps, the final ellipsoid of each tube is required to lie in the terminal set $\mathbb{X}_{\text{term}}$. The action space shrinks $v_k \in \tilde{\mathbb{U}}\big(\mathcal{E}_k^{(i)}\big) = \mathbb{U} \ominus \mathcal{E}\big(0, KS_k^{(i)}K^T\big)$ depending on the uncertainty tubes. For the solver to find a solution, the tubes described by the propagated ellipsoids must be contained in the polytopic state space $\mathbb{X}$. By solving (4.3), not only a safe action is obtained but also a sequence of feedback policies $(\pi_t, \pi_{t+1}, \dots, \pi_{t+N-1})$, where the local controllers $\pi_{t+k} = v_k + K(x_{t+k} - z_k)$ track the actual system toward the nominal trajectory.

4.3.3 Approach to Evaluate the Safety Hypotheses

RQ 3 is investigated using X-MPSC as a predictive safety filter method. The chosen metric for measuring safety is the total number of safety constraint violations that occur throughout the training. Once the learning agent leaves the polytopic state space, a constraint violation is recorded, and the episode terminates early.

Hypothesis 3.1 claims that planning with multiple models decreases the total number of constraint violations. To test this hypothesis, the ensemble size was varied and evaluated over a hyperparameter grid search in Paper IV. In order to avoid cherry-picking good configurations and risking a bias in the general impact of the selected hyperparameter, the average effect of a hyperparameter configuration was measured.

Hypothesis 3.2 states that an additive prior model reduces the total number of constraint violations during training. The validity of Hypothesis 3.2 was tested empirically in Paper IV by comparing the safety violations when training a dynamics model from scratch to the training using a prior dynamics model. The prior model was implemented as an additive component to each member of the ensemble, *i.e.*,

$$f_{\phi_i}(x_t, u_t) = f_{\text{prior}} + \mathcal{N}\big(m_{\phi_i}(x, u), S_{\phi_i}(x, u)\big).$$

The system parameters in f_{prior} were chosen with an error of 20 % compared to the ground-truth parameters of the actual system (2.1).

Constrained RL algorithms were used as baselines in the experimental evaluation because they provide an upper bound on the constrained policy performance (4.2) that can be reached with policy search methods. In particular, the X-MPSC results were compared with the following three algorithm classes: Lagrangian relaxation methods [152], constrained policy search [155, 156], and action projection methods [157, 158].

Chapter 5

Contributions

This chapter summarizes the contributions of Papers I–V included in this thesis. In order to draw the connections to the methodology described in Chapter 4, the context of each paper is presented, and the main findings are summarized. Furthermore, the author's contributions to each paper are outlined.

The contributions of this thesis are divided into two parts. Part A is devoted to the single-agent RL domain and comprises four conference publications. Part B covers the learning with multiple agents and provides a journal paper surveying the landscape and the challenges of the multi-agent domain.

5.1 Part A: Single-Agent Reinforcement Learning

The four papers described in this part address RQs 1–3. Paper I investigates the policy performance in simulation environments, while Papers II and III examine both the robustness and performance of policies in a sim-to-real transfer learning setting with a quadrotor robot. Paper IV addresses the learning-based safety certification of an RL agent.

Paper I †

S. Gronauer, M. Gottwald, and K. Diepold. "The Successful Ingredients of Policy Gradient Algorithms". In: *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*. 2021, pp. 2455–2461. DOI: 10.24963/ijcai.2021/338

Context. Despite the sublime advances in the field of RL, the reproducibility of experimental results is challenging. The reasons for bad reproducibility are manifold. First of all, numerical results are sensitive to the choice of hyperparameters and can vary between different experiment runs [123]. Different code implementations produce

† Core publication: a paper is considered as a core publication if (i) it has been accepted for publication or published as a full paper in an internationally distributed publication organ with peer review process and (ii) the author is the lead author of the publication and has contributed at least 50% of the content.

inconsistent results, despite describing the same algorithm [45, 120]. Besides that, studies showed that a major share of the performance increments cannot be attributed to new algorithmic properties but to minor implementation choices [44, 101]. Other works demonstrated that simple learning representations like linear maps or radial basis functions show comparable performance to more expressive NNs on locomotion benchmark tasks in simulation [160, 161]. These observations contest the common implementation practices of deep RL algorithms and raise the question of which underlying algorithm components and control design choices are responsible for good policy performance.

Summary. This work examines the impact of individual algorithm components and problem design choices on the control policy performance by disentangling the intricate complexity of current policy gradient methods in the on-policy setting. To this end, the three-staged analysis described in Sections 4.1.1–4.1.3 was applied in the experiments, and the impact of individual components on the policy performance was assessed within simulation environments. Based on the results of the three-staged analysis, a minimal configuration of algorithm components was assembled and benchmarked against baseline implementations of TRPO and PPO [138].

Own Contributions. I proposed the methodology and categorization into algorithm core components (Section 4.1.1), code-level enhancements (Section 4.1.2), and structural learning components (Section 4.1.3). Further, I designed the experimental setup and conducted the experiments.

Paper II †

S. Gronauer, M. Kissel, L. Sacchetto, M. Korte, and K. Diepold. "Using Simulation Optimization to Improve Zero-Shot Policy Transfer of Quadrotors". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 10170–10176. DOI: 10.1109/IROS47612.2022.9981229

Context. Many real-world systems cannot afford to learn policies from scratch due to the expense of data. In order to evade the necessity of real-world samples, simulations are commonly utilized to synthesize RL-based policies previous to the real-world deployment [107]. However, the success of a policy transfer is limited when the data at test time differ largely from those seen during the training [21]. The reality gap is, in general, inevitable but can be sufficiently narrowed when the actuator dynamics, noise distributions, and system delays are modeled accurately [31, 105].

Summary. In this paper, the reality gap is addressed from two perspectives: by reducing the reality gap through simulation optimization and by improving the robustness of policies. Overall, the contributions of this paper are threefold. First, data-driven simulation optimization from Section 4.2.2 was applied based on pre-collected real-world data, demonstrating the potential to narrow the reality gap with minimal manual tuning. The experiments confirm that zero-shot policy transfers from the simulation to a quadrotor

robot become feasible after proper simulation optimization. Second, the three control structures described in Section 4.1.4 were tested through real-world experiments with respect to their performance and robustness. Finally, this paper demonstrates that RL can be used for low-level decision-making, and zero-shot transfers to a real-world quadrotor can be successfully accomplished while using onboard sensing and computation only.

Own Contributions. I proposed the three control structures described in Section 4.1.4: PWM, attitude rate, and attitude. Further, I developed the idea of simulation optimization and tuned the simulation parameters based on real-world data, which were collected by a PID controller. Finally, the experiments were carried out under my supervision.

Paper III †

S. Gronauer, D. Stümke, and K. Diepold. "Comparing Quadrotor Control Policies for Zero-Shot Reinforcement Learning under Uncertainty and Partial Observability". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 7508–7514. DOI: 10.1109/IROS55552.2023.10341941

Context. Although Paper II demonstrates that RL-based policies can be zero-shot transferred to a real quadrotor robot, certain aspects remain sparsely explored in terms of practical research. For instance, partial observability is a ubiquitous challenge in real-world systems [19]. While the ground-truth state of the simulation is directly accessible, the state of a real-world system must be typically estimated with a filter based on several sensor readings and a dynamics model [23]. Furthermore, non-stationarity arises as hardware components are subject to wear, distributional shifts emerge due to sensor drifts, and system delays are present during actuation and sensing [164]. To address non-Markovian systems, history-dependent policies are commonly utilized in combination with DR [31, 113]. This practice is also supported by theoretical work proving the importance of history-dependent policies under randomized dynamics [165]. However, there is a lack of practical research about the benefits of using history-dependent policies with RL.

Summary. This work examines history-dependent policies for controlling a quadrotor robot in the context of zero-shot transfer learning. The performance and robustness of the policies are investigated with an emphasis on partial observability and delays in acting and sensing. Two policy architectures are compared with each other: FNNs with a stacked observation-action history and RNNs. Further, the paper contrasts two different observation representations as policy inputs with each other: state estimates provided by an extended Kalman filter (EKF) and raw sensory data from the onboard sensors. As the final contribution, the paper examines if an end-to-end learned representation can control a quadrotor based on raw sensory information only, showing limited yet encouraging results.

Own Contributions. I selected the policy architectures, namely FNNs with a stacked observation-action history and RNNs that process one observation at a time. In addition to that, I suggested two different observation representations, one using raw sensory data and another based on state estimates provided by an EKF. As the final contribution, I designed the experimental setup.

Paper IV †

S. Gronauer, T. Haider, F. Schmoeller da Roza, and K. Diepold. "Reinforcement Learning with Ensemble Model Predictive Safety Certification". In: *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS).* 2024

Context. The exploratory and novelty-seeking nature of RL algorithms requires additional precautions in applications where safety is imperative. Current methods for safe exploration suffer from an inherent trade-off between rigorous safety guarantees and scalability. On the one hand, methods derived from MPC theory can provide formal safety guarantees under known operating conditions by making strict problem assumptions, but their applicability is often limited to low-dimensional systems [118, 119, 167]. On the other hand, RL algorithms can scale to high-dimensional spaces but have shortcomings in terms of hard constraint satisfaction since safety is incentivized and not enforced during the training [168, 169, 170].

Summary. This paper presents X-MPSC as a method to address safe exploration in RL-based systems. X-MPSC aims to combine the best of both worlds by combining robust MPC with model-based RL. As explained in Section 4.3.2, X-MPSC leverages an ensemble of probabilistic NNs and tube-based planning through a multi-step look-ahead to correct potentially unsafe actions. An action is certificated as safe when all tube-based trajectories of the model ensemble enforce the safety constraints during planning. The ensemble of dynamics models is trained on sampled trajectory data from the actual system and is leveraged for both policy optimization and planning with uncertainty tubes. Uncertainty-aware ensemble models are motivated by the prediction inaccuracies that arise with a growing prediction horizon when using a single model only [116]. A great benefit of X-MPSC is that only safe offline data are required, compared to related methods that require mixed safe and unsafe data samples [157, 158, 171]. The experimental results show that X-MPSC can achieve significantly fewer safety constraint violations than comparable RL methods.

Own Contributions. I proposed the framework of X-MPSC and the underlying algorithm for safety certification. This includes the uncertainty propagation based on ellipsoids and the construction of the optimization problem. Further, I led the experimental evaluation and carried out the experiments of the X-MPSC method.

5.2 Part B: Multi-Agent Reinforcement Learning

Many real-world applications naturally comprise multiple agents that make decisions concurrently and adapt control behavior alongside each other [172, 173]. Hence, studying multi-agent problems is fundamental when applying RL in the real world. To enhance the viewpoint of Part A, Paper V comprehensively reviews the recent advances of deep RL in the multi-agent domain and extends upon the challenges introduced in Section 1.1.

Paper V †

S. Gronauer and K. Diepold. "Multi-Agent Deep Reinforcement Learning: A Survey". In: *Artificial Intelligence Review* 55.2 (2022), pp. 895–943. DOI: 10.1007/s10462-021-09996-w

Context. Although being overshadowed by the single-agent domain in the early days of the deep RL success stories [14, 15], MARL has caught up by demonstrating impressive results in high-dimensional control problems [16, 18, 97, 113, 175]. Thus, new interest has ignited the MARL community, and a plethora of literature has been published recently [95, 176, 177]. Although multi-agent systems enjoy a long record of research and numerous seminal surveys have been published over the last two decades [20, 178, 179, 180], the research community lacks a survey paper that holistically overviews the landscape of multi-agent deep reinforcement learning (MADRL) literature.

Summary. This paper complements the challenges formulated in Section 1.1 and reviews the challenges that arise exclusively in the multi-agent domain. To this end, a comprehensive survey of the landscape in MADRL is provided, and the methods proposed to address these challenges are discussed. The focus is primarily on the recent literature that combines deep RL methods with a multi-agent scenario. The contents are divided into three main parts. First, different training schemes are presented, which describe the synthesis of control policies and the information flow. Second, the emergence of different agent behaviors based on the task specification is reviewed. Third, challenges in the multi-agent domain are analyzed, and solution methods, which are leveraged to address these challenges, are reviewed. The survey paper concludes with an extended discussion of research advances and identifies recent trends.

Own Contributions. I conducted the literature review and selected publications related to the survey. In addition to that, I proposed the taxonomy that was used to classify the literature into the analysis of the training schemes, emergent patterns of agent behavior, and challenges of the MADRL domain. As the final contribution, I identified trends that are frequently utilized to address MADRL challenges.

Chapter 6

Results and Discussion

In this chapter, Hypotheses 1.1–3.2 are evaluated through falsification or verification, and RQs 1–3 are addressed accordingly. To this end, the results of Papers I–V are briefly recapitulated, and the empirical findings are discussed based on insights from related works. Lastly, the perspectives on the use of simulations are argued.

6.1 Performance Impact of Algorithm and Control Problem Design

The discussion in this section is based on the results presented in Papers I–III and comprises a mixture of simulation and real-world experiments.

Hypothesis 1.1. *State-of-the-art policy performance can be achieved by deploying a handful of carefully selected algorithm components.*

Hypothesis 1.1 was tested empirically in Paper I by disentangling the algorithm complexity into three stages, where the effect of each component was assessed individually. Through a three-staged analysis, the components with the largest performance gains were selected and composed to a minimal algorithm configuration. The minimal configuration was then benchmarked against the state-of-the-art baseline implementations of TRPO and PPO [138]. In the following paragraph, the insights of the three-staged analysis are discussed.

In the first stage, algorithm core components were found to be crucial for reaching good control performance with on-policy algorithms. Trust-region enforcement methods boosted the policy performance and improved the robustness of the hyperparameter selection in all tested environments. The use of variance reduction techniques resulted in an even greater performance improvement than trust-region enforcement. These results are supported by the findings from Ilyas et al. [181], who reported that using a critic as a baseline function effectively reduces gradient variance, although to a smaller extent than using the true gradient. In the second stage, the impact of code-level enhancements was investigated. Observation standardization and learning rate annealing significantly enhanced the policy performance in at least six out of seven environments. These results are consistent with the ones reported in [44, 121]. Reward scaling showed, on average, improvements in the locomotion tasks but not in the manipulation tasks. This observation

coincides with the findings in [44], where the authors found reward scaling beneficial for learning locomotion tasks. The remaining code-level components showed mixed results, and their effectiveness depends on the control task. In the third and last stage, the impact of the structural learning components was examined. Parameter-sharing deteriorated learning, similar to the results reported in [121]. The choice of the optimizer and the weight initialization scheme had a positive impact on the performance, which contradicts the results in [121], where the authors did not recognize performance changes.

Based on the three-staged analysis, a minimal configuration of algorithm components was assembled and benchmarked against state-of-the-art algorithm implementations of TRPO and PPO [138]. The comparison was conducted on locomotion and manipulation tasks of the PyBullet physics simulation [139]. As the results of Paper I demonstrate, a subset of carefully chosen algorithm components is sufficient to achieve policy performance that is on par with the baseline implementations. This finding confirms Hypothesis 1.1.

Hypothesis 1.2. *Low-level control structures achieve superior policy performance compared to higher levels of control when the reality gap is sufficiently small.*

Paper II conducted zero-shot sim-to-real experiments with a quadrotor robot and compared the policy performance of the three control structures: PWM, attitude rate, and attitude. All three control structures yielded the best zero-shot performance when simulation parameters were set to the ones found through simulation optimization, which are assumed to be closest to the real-world parameters. However, when deviating from the parameters found by simulation optimization, low-level PWM control quickly dropped in performance, and the zero-shot transfer attempts failed. PWM control showed the worst performance compared to the other two control structures when system latency was not modeled in the simulation. In contrast, attitude-rate control showed better performance results than PWM control in settings where simulation parameters differed from the ones found by simulation optimization. Attitude control, although being the highest tested abstraction level, showed worse results than the other two control structures when the reality gap was small. Although PWM and attitude-rate controllers showed the best task performance in the experiments of Paper II, no control level surpassed another in terms of zero-shot performance.

Related works studied the impact of the action space design within simulation environments (*i.e.*, settings without a reality gap). In the majority of works, high-level action space representations showed the best final performance in simulation [135, 137, 182] compared to the literature reporting the best reward performance with low-level control [183]. There is also a broad consensus in the literature that the selection of a higher control abstraction generally results in improved learning speed, as shown across various robot platforms [135, 137, 182, 183]. This is backed up by recent results indicating that high-level control structures can offer a smoother optimization landscape and thus can accelerate learning [184]. In sim-to-real experiments, Kaufmann et al. [112] found attitude-rate control to be the best control approach. Low-level thrust control failed in their experiments when the communication delay was increased.

The results of Paper II do not indicate the superiority of PWM over attitude-rate control when the reality gap is narrow. Without the reality gap, simulation-based studies found higher control levels superior in terms of reward performance across various robot

systems, *e.g.*, in manipulation tasks [137] and locomotion tasks [135, 182]. Based on these results, Hypothesis 1.2 is rejected.

Hypothesis 1.3. *Recurrent actor-critics improve the policy performance upon feed-forward architectures when trained with domain randomization.*

The experimental evaluation of Paper III showed performance gains when using RNN-based policies instead of FNNs. Based on state estimates of an EKF, the RNN-based policies showed a marginally better cumulative reward than FNN architectures when all system parameters were randomized in simulation. Despite one exception, all policies were able to fly the quadrotor for at least 20 s, after which the trial was terminated manually. However, when using raw sensory data as observations, the trials with FNN-based policies led to unstable control behavior. Most of the trials with FNNs terminated early, resulting in a success rate of less than 7 %. In contrast, RNN-based policies trained with DR achieved a success rate of 66.7 % on raw sensory inputs. In further experiments, it was tested if an MLP classifier could predict the simulated system latency from latent memory states. The results indicate that RNNs extract and encode useful information about the latency from the sequence of partial observations in the latent state. Similar results were found in [106], who were able to recover relevant dynamics parameters from the state-action history.

Related works on sim-to-real transfer learning reported that RNNs are superior to FNNs when DR is applied during training. On a hand-in manipulation task, FNNs were compared with RNNs, demonstrating that LSTM-based actor-critics can considerably outperform their FNN counterparts [35]. Further, the authors observed that the latent state of the RNN encodes valuable information about the environment randomization. In another study about robotic manipulation, recurrent policies achieved a higher zero-shot success rate than FNN-based policies despite showing similar task performance in simulation [105]. The authors argue that LSTMs generalize better to the dynamics of the real-world robot. Similar to the finding of Paper III, FNN-based policies were found to be on par with RNNs when the reality gap is small and accurate robot state estimates are available [185].

The results of Paper III confirm Hypothesis 1.3. RNNs can infer the system parameters from temporal data and encode the realization information in the latent state when trained with DR. As a result, RNN-based policies are adaptive controllers due to their implicit system identification at deployment. The validity of the hypothesis is supported by several related works pointing out the superiority of RNNs when combined with DR [35, 105, 106].

RQ 1. *How do algorithm implementation and problem design choices affect policy performance in a zero-shot sim-to-real context?*

In summary, the policy performance obtained through RL is the product of an intricate interplay of various algorithm components and control problem design choices. In what follows, the merits and demerits of these choices are discussed.

Algorithm components impact the stability and speed of learning, the asymptotic policy performance, and the resilience toward hyperparameters. However, there are few studies

in the literature that consider the sim-to-real transfer performance based on algorithm components. Most findings are based on simulation experiments [44, 45, 121].

The policy representation can significantly impact learning performance. Changing the NN architecture, such as the depth or width, can degrade training performance [45]. In contrast, clever NN parameter initialization can foster exploration and improve learning speed [121]. Similarly, the regularization of the policy parameters can have a positive impact on control performance [186]. Representations based on RNNs address not only problems with incomplete information [38, 187] but also provide superior reward performance in zero-shot sim-to-real transfer learning due to the online adaption capabilities of RNNs when trained with DR [35, 105, 106, 188]. However, in settings with a small reality gap and accurate state estimates, FNNs can be on par with RNN-based policies [163, 185].

The design of the action space affects the learning speed and the final policy performance. Since reducing the reality gap requires considerable engineering effort, higher control structures are generally the preferred choice for deployment. This statement is substantiated by the majority of works utilizing high-level control structures [33, 34, 35, 36, 105, 113, 188, 189] compared to the works deploying low-level controllers [38, 183, 185, 190]. Low-level control structures can only be applied in control problems where a simulation with high fidelity is available. Besides the abstraction level, a change in the action space parameterization can foster the exploration of the state space and improve performance, *e.g.*, bang-bang controllers based on Bernoulli distributions can outperform Gaussian action distributions [136].

A thorough observation space design can enhance policy transfer performance. Experiments in simulation showed that a careful selection of observation space features can boost learning speed and performance [191]. In sim-to-real experiments on a quadrupedal robot, an observation space with reduced dimensionality was shown to result in lower performance in simulation but improved the controller performance on the real hardware [142]. The authors believe that with a small observation space, the policy is more likely to encounter data samples known from the simulation at deployment, resulting in a higher transfer success rate.

Other design choices include the reward function design, the initial state distribution, and the handling of episode resets. A diligent reward function design guides the learning process and can improve both sample complexity and policy performance [192]. Typically, domain knowledge is used for engineering the reward function to incentivize success-oriented behavior, *e.g.*, reward the proximity toward a task goal. The most known technique is reward shaping, which has been successfully applied in robotics [33, 35] and video games [193, 194] to tackle the credit assignment problem. Next to the reward function, small initial state distributions have been shown to boost the learning curve but fail to generalize in regions that have not been visited during training, whereas wide initial state distributions can hinder learning performance [122]. Finally, performance improvements are achieved when bootstrapping at the end of episodes in case of timeouts [195]. Timeouts cause the infinite horizon assumption to break and can diminish the prediction accuracy of the critic, which results in decreased policy performance if not handled carefully [196].

6.2 Robustness in Zero-Shot Policy Transfers

The experiments of Papers II and III tested the robustness of policies that were zero-shot transferred to a real quadrotor robot. In order to assess the robustness of a policy, the reality gap was increased by changing the values of critical simulation parameters to enforce a distributional shift.

Hypothesis 2.1. *High-level control structures are more robust toward the reality gap than low-level controllers.*

In the experiments of Paper II, three different control structures were compared: PWM, attitude rate, and attitude. Low-level PWM policies were only transferable when the dynamics of the actuator and the system latency were set close to the values found through simulation optimization. When the reality gap increased, the trials terminated early since the drone became unstable, resulting in failure in most cases. Attitude-rate control showed more robust behavior toward the change of simulation parameters than PWM-based controllers. Finally, attitude control showed, in settings where simulation parameters were close to the ones found by simulation optimization, worse reward performance than the other two control structures. However, when the reality gap was widened, attitude control showed better results than PWM control.

Related studies investigated the robustness of different action space representations toward the reality gap. Kaufmann et al. [112] tested the sensitivity toward delays in zero-shot transfers for drone control, showing that a higher action space abstraction is more robust toward latency. In a bipedal locomotion task, Haarnoja et al. [113] reported that the zero-shots failed with torque-based actions but were able to conduct successful transfers with position-based joint control. The authors argue that fast stabilizing and local feedback from the PID controllers renders high-level control more robust toward the reality gap.

The empirical results of Paper II and the evidence found in related works confirm Hypothesis 2.1. Low-level action representations require a higher simulation fidelity, which promotes the usage of high-level control structures in settings where the reality gap is difficult to minimize, *e.g.*, when robot dynamics are partially unknown, or system parameters are not accurately identified.

Hypothesis 2.2. *Policies based on recurrent neural networks are more robust than feed-forward architectures when trained with domain randomization.*

The experiments of Paper III compared RNN-based policies with FNN architectures regarding the robustness toward system delays. With raw sensory inputs, the success rate of both architectures dropped to 0 % when latency was not modeled during training in simulation. Using state estimates as policy inputs, RNNs dropped in terms of the success rate from 100 % to 20 % and FNNs from 100 % to 66.7 %. The results of Paper III provide no evidence that the use of RNNs offers robustness benefits over FNN architectures. However, the experiments of Paper III evidence that RNNs can overfit the simulation dynamics when DR is disabled during training. With raw sensory data as policy inputs, this overfitting led to a success rate of 0 % compared to 66.7 % when training with DR.

A similar observation was made in a bipedal locomotion task, where RNNs overfitted to the simulation dynamics without DR [106].

Related works demonstrated that RNN-based policies yield better zero-shot performance than FNNs on robotic manipulation tasks [35, 105]. However, improved policy performance was also observed during the training in simulation, where recurrent actors learned faster and achieved higher asymptotic performance. The superiority of RNNs in the zero-shot transfer context could be justified by a better training performance rather than superior robustness toward the reality gap.

Although related works evidence the positive performance effects of recurrent policy representations in various tasks, the performance gains over FNNs can also be attributed to the improved control performance in simulation rather than superior robustness. When exposed to an out-of-distribution test scenario, RNN-based policies fail in a similar way as their feed-forward counterparts, as the experiments of Paper III show. Based on these observations, Hypothesis 2.2 is rejected.

RQ 2. *What design choices impact the robustness of policies in zero-shot sim-to-real transfer learning?*

The evaluation of Hypotheses 2.1 and 2.2 underpins the relevance of using DR to improve the robustness of policies in a zero-shot context. This insight is supported by the vast majority of papers that address the sim-to-real transfer with DR during training in simulation [31, 34, 38, 103, 105, 113, 142, 189] compared to the works that train without DR [197, 198]. Moreover, ablation studies examined the impact of DR in manipulation tasks [35, 105, 199] and a quadrotor flight task [200], approving the importance of DR for successful policy transfers.

Besides DR, the level of abstraction impacts the robustness of policies. Abstraction in the observation space, where policies operate on a common intermediate representation, helps to bridge the reality gap, especially in high-dimensional input spaces. Domain adaptation was successfully used on pixel inputs [34, 36, 201] and feature levels [133, 189]. Furthermore, the right representation of the action space can alleviate the reality gap and can render policy transfers feasible [198]. High-level control structures require less simulation fidelity and are more robust toward the reality gap than low-level controllers [112, 113].

Recurrency in the policy representation helps in situations where systems are only partially observable [189] or the deployment conditions change over time. However, an improved robustness of RNNs over FNNs cannot be confirmed.

6.3 Safety Certification of Reinforcement Learning

The results of this section are based on Paper IV, which proposed X-MPSC as an approach to certify the actions of an RL-based controller. This section discusses how the ensemble size of probabilistic NNs and the usage of a prior dynamics model impact the learning-based safety certification of the closed-loop control system when integrating X-MPSC.

Hypothesis 3.1. *An ensemble of dynamics models decreases the number of safety constraint violations.*

A hyperparameter search over crucial algorithm hyperparameters was conducted in Paper IV to assess the impact of the ensemble size. The results demonstrate that the total number of safety constraint violations decreases with an increasing ensemble size. However, the improvement in terms of safety constraint satisfaction comes at the cost of a reduced cumulative reward. This behavior can be explained by the system becoming more conservative as the ensemble size increases since the constraints imposed by each individual dynamics model must be satisfied. If a single member of the ensemble deviates from the remaining members, the agent is enforced to satisfy ambiguous constraints. This can result in a reduced set of feasible actions, which can lead to diminished policy performance.

Related works that use an ensemble of NNs for safe RL exist but lack an investigation into the effect of the ensemble. Luo and Ma [202] utilized barrier certificates together with an ensemble of NNs, whose usage is argued by the common practice of a well-calibrated dynamics model to address epistemic uncertainty in the predictions. Lütjens et al. [203] used an ensemble of RNNs for uncertainty-aware predictions that are used by an MPC controller to act more cautiously in scenarios with high uncertainty. Similarly, Zhang et al. [204] used an ensemble of probabilistic NNs to select actions that lead to low-variance predictions. The visitation of states with high uncertainty can thus be reduced.

The results of Paper IV confirm Hypothesis 3.1. An increasing number of models improves the learning-based safety certification of actions but also renders the closed-loop system more conservative.

Hypothesis 3.2. *Adding a crude prior model to the learning representation lowers the number of safety constraint violations.*

To assess the validity of this hypothesis, the cumulative constraint violations were compared between learning a dynamics model from scratch and incorporating a prior model to the NN ensemble. For this purpose, each member of the ensemble was augmented with an additive term that described the nominal model. The experiments in Paper IV demonstrate that the total number of cumulative constraint violations can be reduced on average by one order of magnitude when a crude prior model is added to the NN ensemble. The positive impact can be observed in all of the four tested tasks. Moreover, the usage of a prior model improves the safety certification without reducing the policy performance.

The positive impact of adding prior model knowledge to the learning agent was also reported in [205], where the authors showed that a crude prior model in combination with a control barrier function could ensure safe exploration. Other related works did not investigate the impact of the prior model but require a sufficiently calibrated prior model to provide safety guarantees under known operating conditions [118, 119, 206].

The experimental results of Paper IV verify Hypothesis 3.2. The incorporation of prior knowledge in the form of model priors or domain structure is beneficial whenever available, as similarly argued in [23]. Beyond safe exploration, related works reported that prior knowledge about the robot system significantly improves the sample complexity [207, 208] and learning speed [209].

RQ 3. *How can safety be improved in learning-based model predictive safety certification?*

Accurate model predictions are crucial for reliable safety certification. The use of a prior dynamics model requires the NNs to learn only the residual between the nominal model and the actual system dynamics, thus helping to improve prediction accuracy. Even though no data is available from unseen regions, the nominal model can guide the model rollouts through coarse predictions and improved gradient estimates for the propagation of the uncertainty tubes. Furthermore, the utilization of multiple models improves the safety of the closed-loop system since only one model needs to capture the future behavior of the actual system. Since X-MPSC requires satisfying the state-trajectory constraints for each ensemble member, safety can be ensured when at least one ensemble member captures the trajectory of the actual system.

Based on these observations, it can be concluded that approaches enhancing the prediction capabilities of the nominal model and improving uncertainty estimates enhance the effectiveness of learning-based MPSC methods. This conclusion is supported by model-based RL works that found the model quality and the usage of uncertainty during learning to be essential for task performance [58]. Models with more expressive representations could further improve the prediction capabilities, *e.g.*, state-space models [59, 210] or attention-based architectures [211, 212].

6.4 On the Role of Simulation

Despite the possibility of training directly on the real hardware [37, 127, 213, 214], the majority of works deploy policies that were trained on simulated data [31, 33, 35, 36, 38, 105, 189, 200]. Sim-to-real methods mitigate the problem of gathering costly data on the robot and enable faster than real-time learning. However, the training in simulation comes at the expense of the reality gap, as simulation models are often considerably simpler than their real-world equivalents.

In this section, the drawbacks and benefits of using a simulation for training RL policies are summarized. To this end, insights that occurred in Papers II and III are discussed based on the findings of related works.

6.4.1 Drawbacks

Reality Gap. Although the reality gap is inherently inevitable, it can still be approached from two perspectives. First, the richness and fidelity of the simulation can be improved data-driven through system identification [113, 142, 198, 215] and parameter distribution optimization [144]. Both approaches aim to narrow the reality gap by shifting the simulation model as close as possible to the real-world system. However, it is often difficult to account for all phenomena and dynamical effects in the simulation because the modeling of all subtleties can be computationally demanding, *e.g.*, the calculation of aerodynamic effects in drone flights can result in a significant slowdown of the simulation speed. Second, the policy can be made robust against the distributional shift caused by the transfer [105, 141]. Based on the discussion of RQ 2, the use of DR and good representations of the action and state space improve policy robustness. An instance

of the latter is domain adaptation, which can close the reality gap through training a policy on an intermediate space representation instead of the original state space [34, 36, 201]. By making the policy operate on a mutual and uniform representation, disparities between the simulation and reality can be addressed by aligning both domains.

Actuator Model. Throughout the literature, it has been reported that the accurate modeling of the actuator dynamics is a crucial component for reliable zero-shot sim-to-real transfers. The works range from quadrupedal robots [31, 142], bipedal robots [113, 185], and robotic manipulators [105] to quadrotors [33]. Besides the importance of accurate actuator dynamics, the main cause of model errors is the lack of latency modeling in the context of legged robotics [25]. Papers II and III draw a similar conclusion for quadrotors, where the transfer attempts were only successful when system latency and motor dynamics were modeled. Further, Paper II showed that adding latency to the simulation acts as a kind of regularization, preventing high-frequent action changes.

Aggressive Action Selection. Bang-bang control behavior is a phenomenon where policies favor actions close to the boundaries of the action space [136]. It was found that bang-bang control behavior can naturally emerge when using RL for policy training [216, 217]. While jerky actions are not an issue in simulation, they can be prohibitive on physical robots and can diminish the sim-to-real transfer success. A diligent reward function design can enforce a smoother action selection and thus foster transferability to the real-world robot. In the experiments of Paper II, policies produced large changes in the control outputs without additional reward penalties, resulting in deteriorated control performance after the transfer. After adding terms to penalize high action rates and magnitudes to the reward function, policies could reliably be transferred to the quadrotor.

6.4.2 Benefits

Automation of Experiments. Simulations enable the automation of the experimentation. At the end of each episode, the robot can be easily reset into its initial configuration. On real hardware, however, human supervision or intervention mechanisms are required when unexpected events such as failures and breakdowns occur [30] or at episodic resets [218]. Training in simulation accelerates the experimentation cycles through the automated testing and tuning of algorithm hyperparameters [25].

Data Throughput. Simulators typically run faster than real-time and can speed up data generation significantly. Moreover, the data throughput is further increased when the number of simulated robots can be parallelized on the computational resources [196, 219]. High simulation throughput facilitates the usage of on-policy RL algorithms, which require a higher sample complexity but offer more stable learning than off-policy algorithms.

Safety Aspects. Safety is imperative on robots since failures lead to breakdowns of the hardware or damage to the robot's environment. In the simulation, unsafe behavior is not only tolerated but also desirable to learn from the consequences of failures. Furthermore,

simulations offer a convenient way to develop safety mechanisms [220, 221] or learn policies that adhere to safety constraints [155, 222]. For sim-to-real transfer learning, it can be beneficial to include safety specifications already during the training in the simulation, *i.e.*, agents can violate constraints during the simulation but must enforce these before transfer [150].

Non-Markovian Systems. Sensor measurements on real robots are corrupted with noise and generally comprise only a subset of the state space variables necessary for modeling the robot. State estimators based on filters are thus common practice to estimate the robot state and quantify the uncertainty about the estimates [23]. Simulators bypass partial observations since complete state information is available. For instance, ground-truth information can be leveraged to train a privileged agent. Imitation learning can then be used to train a student agent that mimics the behavior of the privileged agent while having only the sensor capabilities of the hardware platform. Privileged learning is an effective method to address partial observability and has been successfully applied to quadrotor robots [34, 133] and legged robots [32, 189]. However, policies can also be trained on raw sensory data instead of state estimates when using RNNs, as shown in Paper III. Another challenge in real-world applications is the reward feedback. Robot systems can require additional sensors to track the task progress and obtain information about goal completion.

Chapter 7

Conclusion

This thesis addresses the application of deep RL algorithms to robots in a sim-to-real transfer learning context. Despite the advances in recent years, a plethora of challenges prevails when deploying RL-based policies to real-world systems. In order to provide contributions to these challenges, this thesis answers three RQs.

The first question addresses the performance of RL algorithms regarding the selected algorithm components and the control problem design. As indicated by the experiments, a minimal configuration of RL algorithm components is sufficient to achieve comparable policy performance to state-of-the-art baseline implementations. Since experimental reproducibility is a known deficiency that exacerbates the adoption of RL algorithms to new tasks, reduced complexity in the algorithm design is beneficial and can accelerate the experimentation cycles. Through zero-shot policy transfers to a quadrotor robot, the impact of the action space representation on the policy performance is examined. Actions in higher abstraction levels are generally favorable due to superior learning speed and final performance. In experiments examining the policy representation, RNN-based policies increased policy performance over FNNs on state estimates and surpassed FNN-based architectures on partial observations. However, RNNs unfold their potential only when trained with DR in simulation. The experiments also provide evidence that RNNs encode useful dynamics information in the latent memory when trained with DR, enabling adaptive control behavior at deployment.

The second question covers the robustness of policies in a zero-shot transfer context. Robustness is crucial for alleviating the model mismatch between the simulation and reality to ensure reliable transfer performance. As evidenced in the experiments with a quadrotor robot, high-level control structures require less simulation fidelity and increase the robustness toward the reality gap. Based on the performance and robustness findings, high-level action representations can be recommended as the default choice for most robotic systems. Regarding the policy representation, RNN-based policies enhance the sim-to-real transfer success but are not robust toward situations that were not encountered in simulation. Furthermore, the experiments indicate an overfitting of RNNs to the simulation when DR is not applied during training.

In the third and last question, the safety certification of actions during the exploration phase is addressed based on X-MPSC. The results indicate that augmenting the learning model with prior physics knowledge enhances both the safety and learning speed of the learning-based system without forfeiting control performance. This suggests incorporating

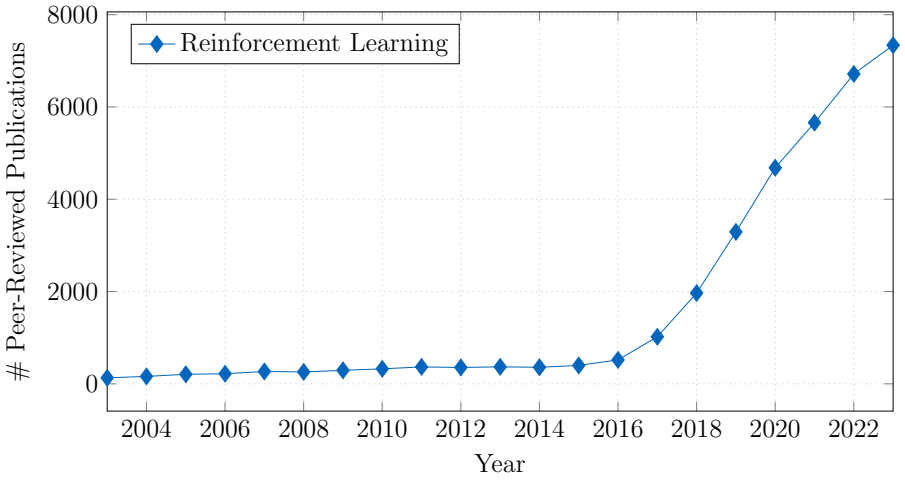


Figure 7.1: Number of peer-reviewed RL publications per year. The computer science bibliography DBLP (`https://dblp.org/`) is used as data source to determine the number of publications per year. Search matching patterns include a paper's title and publication organ. Information from the abstract and the paper's main content is neglected.

prior knowledge whenever available. Furthermore, leveraging an ensemble of probabilistic NNs mitigates model bias and reduces the number of safety constraint violations that occur throughout training. However, the ensemble model renders the closed-loop control system more conservative, which results in reduced cumulative reward.

In summary, the findings of this thesis substantiate the practicability of data-driven paradigms to control robots. Nevertheless, the successful deployment of an RL algorithm on a robot requires careful engineering in various aspects. The algorithm components and hyperparameters must be tuned, and the control problem must be designed properly. Furthermore, a calibrated model is pivotal for accurate simulation and obtaining high-quality state estimates on the real robot. This conclusion objects to the idealistic notion that RL can be adopted in new tasks and robots without prior knowledge and thorough system design.

However, RL has the potential to substitute conventional control approaches in domains where the controller synthesis is challenging, *e.g.*, due to high system complexity or (partially) unknown dynamics. This can be argued for the better optimization objective in RL since the control performance is maximized on task-level in an end-to-end fashion and does not require a decomposition into planning and control [223]. Learning-based methodologies also provide the benefit of continuous adaptation, which is a crucial property when deploying robots to the real world [224]. These advances may explain the trend that can be observed in the increasing adoption of RL and NNs in different robotics areas, *e.g.*, a rising number of NN-based controllers on quadrotors [225] and the deployment of RL-based policies for robotic manipulation tasks [226]. Prompted by seminal deep RL works [14, 15], vigorous research efforts have been made, and the rapid growth of published papers (as depicted in Figure 7.1) underpins the sustained interest in data-driven control paradigms and the high expectations that come along with an automated controller design.

Bibliography

[1] K. J. Åström. "Process Control–Past, Present and Future". In: *IEEE Control Systems Magazine* 5.3 (1985), pp. 3–10. DOI: 10.1109/MCS.1985.1104958.

[2] S.-L. Jämsä-Jounela. "Future Trends in Process Automation". In: *Annual Reviews in Control* 31.2 (2007), pp. 211–220. DOI: 10.1016/j.arcontrol.2007.08.003.

[3] O. Egeland and J. T. Gravdahl. *Modeling and Simulation for Automatic Control*. Marine Cybernetics, 2002.

[4] K. J. Åström and R. M. Murray. *Feedback Systems: An Introduction for Scientists and Engineers*. Princeton University Press, 2021.

[5] L. Hewing, K. P. Wabersich, M. Menner, and M. N. Zeilinger. "Learning-Based Model Predictive Control: Toward Safe Learning in Control". In: *Annual Review of Control, Robotics, and Autonomous Systems* 3.1 (2020), pp. 269–296. DOI: 10.1146/annurev-control-090419-075625.

[6] B. Recht. "A Tour of Reinforcement Learning: The View from Continuous Control". In: *Annual Review of Control, Robotics, and Autonomous Systems* 2.1 (2019), pp. 253–279. DOI: 10.1146/annurev-control-053018-023825.

[7] D. Bertsekas. *Reinforcement Learning and Optimal Control*. Athena Scientific, 2019.

[8] R. S. Sutton and A. G. Barto. *Reinforcement Learning: An Introduction*. 2nd Edition. MIT Press, 2018.

[9] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. "Deep Reinforcement Learning: A Brief Survey". In: *IEEE Signal Processing Magazine* 34.6 (2017), pp. 26–38. DOI: 10.1109/MSP.2017.2743240.

[10] D. Silver, J. Schrittwieser, K. Simonyan, I. Antonoglou, A. Huang, A. Guez, T. Hubert, L. Baker, M. Lai, A. Bolton, Y. Chen, T. Lillicrap, F. Hui, L. Sifre, G. van den Driessche, T. Graepel, and D. Hassabis. "Mastering the Game of Go without Human Knowledge". In: *Nature* 550.7676 (2017), pp. 354–359. DOI: 10.1038/nature24270.

[11] A. M. Annaswamy. "Adaptive Control and Intersections with Reinforcement Learning". In: *Annual Review of Control, Robotics, and Autonomous Systems* 6.1 (2023), pp. 65–93. DOI: 10.1146/annurev-control-062922-090153.

[12] Y. Bengio, A. Courville, and P. Vincent. "Representation Learning: A Review and New Perspectives". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 35.8 (2013), pp. 1798–1828. DOI: 10.1109/TPAMI.2013.50.

[13] Y. LeCun, Y. Bengio, and G. Hinton. "Deep Learning". In: *Nature* 521.7553 (2015), pp. 436–444. DOI: 10.1038/nature14539.

[14] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis. "Human-Level Control through Deep Reinforcement Learning". In: *Nature* 518 (2015), 529 EP. DOI: 10.1038/nature14236.

[15] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis. "Mastering the Game of Go with Deep Neural Networks and Tree Search". In: *Nature* 529.7587 (2016), pp. 484–489. DOI: 10.1038/nature16961.

[16] B. Baker, I. Kanitscheider, T. Markov, Y. Wu, G. Powell, B. McGrew, and I. Mordatch. "Emergent Tool Use From Multi-Agent Autocurricula". In: *8th International Conference on Learning Representations (ICLR)*. 2020.

[17] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis. "A General Reinforcement Learning Algorithm that Masters Chess, Shogi, and Go through Self-play". In: *Science* 362.6419 (2018), pp. 1140–1144. DOI: 10.1126/science.aar6404.

[18] O. Vinyals, I. Babuschkin, W. M. Czarnecki, M. Mathieu, A. Dudzik, J. Chung, D. H. Choi, R. Powell, T. Ewalds, P. Georgiev, J. Oh, D. Horgan, M. Kroiss, I. Danihelka, A. Huang, L. Sifre, T. Cai, J. P. Agapiou, M. Jaderberg, A. S. Vezhnevets, R. Leblond, T. Pohlen, V. Dalibard, D. Budden, Y. Sulsky, J. Molloy, T. L. Paine, C. Gulcehre, Z. Wang, T. Pfaff, Y. Wu, R. Ring, D. Yogatama, D. Wünsch, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, K. Kavukcuoglu, D. Hassabis, C. Apps, and D. Silver. "Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning". In: *Nature* 575.7782 (2019), pp. 350–354. DOI: 10.1038/s41586-019-1724-z.

[19] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester. "Challenges of Real-World Reinforcement Learning: Definitions, Benchmarks and Analysis". In: *Machine Learning* 110.9 (2021), pp. 2419–2468. DOI: 10.1007/s10994-021-05961-4.

[20] L. Busoniu, R. Babuska, and B. De Schutter. "A Comprehensive Survey of Multiagent Reinforcement Learning". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 38.2 (2008), pp. 156–172. DOI: 10.1109/TSMCC.2007.913919.

[21] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel. "A Survey of Zero-shot Generalisation in Deep Reinforcement Learning". In: *Journal of Artificial Intelligence Research* 76 (2023), pp. 201–264. DOI: 10.1613/jair.1.14174.

[22] H. Kurniawati. "Partially Observable Markov Decision Processes and Robotics". In: *Annual Review of Control, Robotics, and Autonomous Systems* 5.1 (2022), pp. 253–277. DOI: 10.1146/annurev-control-042920-092451.

[23] J. Kober, J. A. Bagnell, and J. Peters. "Reinforcement Learning in Robotics: A Survey". In: *The International Journal of Robotics Research* 32.11 (2013), pp. 1238–1274. DOI: 10.1177/0278364913495721.

[24] Y. Yu. "Towards Sample Efficient Reinforcement Learning". In: *Proceedings of the 27th International Joint Conference on Artificial Intelligence (IJCAI)*. 2018, pp. 5739–5743. DOI: 10.24963/ijcai.2018/820.

[25] J. Ibarz, J. Tan, C. Finn, M. Kalakrishnan, P. Pastor, and S. Levine. "How to Train Your Robot with Deep Reinforcement Learning: Lessons We Have Learned". In: *The International Journal of Robotics Research* 40.4-5 (2021), pp. 698–721. DOI: 10.1177/0278364920987859.

[26] L. Brunke, M. Greeff, A. W. Hall, Z. Yuan, S. Zhou, J. Panerati, and A. P. Schoellig. "Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning". In: *Annual Review of Control, Robotics, and Autonomous Systems* 5.1 (2022), pp. 411–444. DOI: 10.1146/annurev-control-042920-020211.

[27] J. García, Fern, and o Fernández. "A Comprehensive Survey on Safe Reinforcement Learning". In: *Journal of Machine Learning Research* 16.42 (2015), pp. 1437–1480.

[28] L. Pinto, J. Davidson, R. Sukthankar, and A. Gupta. "Robust Adversarial Reinforcement Learning". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, pp. 2817–2826.

[29] J. Moos, K. Hansel, H. Abdulsamad, S. Stark, D. Clever, and J. Peters. "Robust Reinforcement Learning: A Review of Foundations and Recent Advances". In: *Machine Learning and Knowledge Extraction* 4.1 (2022), pp. 276–315. DOI: 10.3390/make4010013.

[30] D. Amodei, C. Olah, J. Steinhardt, P. F. Christiano, J. Schulman, and D. Mané. *Concrete Problems in AI Safety*. Preprint. 2016. DOI: 10.48550/arXiv.1606.06565.

[31] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter. "Learning Agile and Dynamic Motor Skills for Legged Robots". In: *Science Robotics* 4.26 (2019), eaau5872. DOI: 10.1126/scirobotics.aau5872.

[32] J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. "Learning Quadrupedal Locomotion over Challenging Terrain". In: *Science Robotics* 5.47 (2020), eabc5986. DOI: 10.1126/scirobotics.abc5986.

[33] E. Kaufmann, L. Bauersfeld, A. Loquercio, M. Müller, V. Koltun, and D. Scaramuzza. "Champion-Level Drone Racing Using Deep Reinforcement Learning". In: *Nature* 620.7976 (2023), pp. 982–987. DOI: 10.1038/s41586-023-06419-4.

[34] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. "Learning High-Speed Flight in the Wild". In: *Science Robotics* 6.59 (2021), eabg5810. DOI: 10.1126/scirobotics.abg5810.

[35] O. M. Andrychowicz, B. Baker, M. Chociej, R. Józefowicz, B. McGrew, J. Pachocki, A. Petron, M. Plappert, G. Powell, A. Ray, J. Schneider, S. Sidor, J. Tobin, P. Welinder, L. Weng, and W. Zaremba. "Learning Dexterous In-Hand Manipulation". In: *The International Journal of Robotics Research* 39.1 (2020), pp. 3–20. DOI: 10.1177/0278364919887447.

[36] S. James, P. Wohlhart, M. Kalakrishnan, D. Kalashnikov, A. Irpan, J. Ibarz, S. Levine, R. Hadsell, and K. Bousmalis. "Sim-To-Real via Sim-To-Sim: Data-Efficient Robotic Grasping via Randomized-To-Canonical Adaptation Networks". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 12619–12629.

[37] D. Kalashnikov, A. Irpan, P. Pastor, J. Ibarz, A. Herzog, E. Jang, D. Quillen, E. Holly, M. Kalakrishnan, V. Vanhoucke, and S. Levine. "Scalable Deep Reinforcement Learning for Vision-Based Robotic Manipulation". In: *Proceedings of the 2nd Conference on Robot Learning (CoRL)*. 2018, pp. 651–673.

[38] J. Degrave, F. Felici, J. Buchli, M. Neunert, B. Tracey, F. Carpanese, T. Ewalds, R. Hafner, A. Abdolmaleki, D. de las Casas, C. Donner, L. Fritz, C. Galperti, A. Huber, J. Keeling, M. Tsimpoukelli, J. Kay, A. Merle, J.-M. Moret, S. Noury, F. Pesamosca, D. Pfau, O. Sauter, C. Sommariva, S. Coda, B. Duval, A. Fasoli, P. Kohli, K. Kavukcuoglu, D. Hassabis, and M. Riedmiller. "Magnetic Control of Tokamak Plasmas through Deep Reinforcement Learning". In: *Nature* 602.7897 (2022), pp. 414–419. DOI: 10.1038/s41586-021-04301-9.

[39] J. Günther, P. M. Pilarski, G. Helfrich, H. Shen, and K. Diepold. "Intelligent Laser Welding through Representation, Prediction, and Control Learning: An Architecture with Deep Neural Networks and Reinforcement Learning". In: *Mechatronics* 34 (2016), pp. 1–11. DOI: 10.1016/j.mechatronics.2015.09.004.

[40] D. J. Mankowitz, A. Michi, A. Zhernov, M. Gelmi, M. Selvi, C. Paduraru, E. Leurent, S. Iqbal, J.-B. Lespiau, A. Ahern, T. Köppe, K. Millikin, S. Gaffney, S. Elster, J. Broshear, C. Gamble, K. Milan, R. Tung, M. Hwang, T. Cemgil, M. Barekatain, Y. Li, A. Mandhane, T. Hubert, J. Schrittwieser, D. Hassabis, P. Kohli, M. Riedmiller, O. Vinyals, and D. Silver. "Faster Sorting Algorithms Discovered using Deep Reinforcement Learning". In: *Nature* 618.7964 (2023), pp. 257–263. DOI: 10.1038/s41586-023-06004-9.

[41] A. Fawzi, M. Balog, A. Huang, T. Hubert, B. Romera-Paredes, M. Barekatain, A. Novikov, F. J. R. Ruiz, J. Schrittwieser, G. Swirszcz, D. Silver, D. Hassabis, and P. Kohli. "Discovering Faster Matrix Multiplication Algorithms with Reinforcement Learning". In: *Nature* 610.7930 (2022), pp. 47–53. DOI: 10.1038/s41586-022-05172-4.

[42] A. Mirhoseini, A. Goldie, M. Yazgan, J. W. Jiang, E. Songhori, S. Wang, Y.-J. Lee, E. Johnson, O. Pathak, A. Nazi, J. Pak, A. Tong, K. Srinivasa, W. Hang, E. Tuncer, Q. V. Le, J. Laudon, R. Ho, R. Carpenter, and J. Dean. "A Graph Placement Methodology for Fast Chip Design". In: *Nature* 594.7862 (2021), pp. 207–212. DOI: 10.1038/s41586-021-03544-w.

[43] M. Popova, O. Isayev, and A. Tropsha. "Deep Reinforcement Learning for de novo Drug Design". In: *Science Advances* 4.7 (2018), eaap7885. DOI: 10.1126/sciadv.aap7885.

[44] L. Engstrom, A. Ilyas, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry. "Implementation Matters in Deep RL: A Case Study on PPO and TRPO". In: *8th International Conference on Learning Representations (ICLR)*. 2020.

[45] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger. "Deep Reinforcement Learning that Matters". In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*. 2018, pp. 3207–3214. DOI: 10.1609/aaai.v32i1.11694.

[46] D. P. Bertsekas and J. N. Tsitsiklis. "Neuro-Dynamic Programming: An Overview". In: *IEEE Conference on Decision and Control (CDC)*. 1995, pp. 560–564. DOI: 10.1109/CDC.1995.478953.

[47] F. Borrelli, A. Bemporad, and M. Morari. *Predictive Control for Linear and Hybrid Systems*. Cambridge University Press, 2017. DOI: 10.1017/9781139061759.

[48] J. B. Rawlings, D. Q. Mayne, and M. Diehl. *Model Predictive Control: Theory, Computation, and Design*. 2nd Edition. Nob Hill Publishing, 2017.

[49] K. P. Wabersich, L. Hewing, A. Carron, and M. N. Zeilinger. "Probabilistic Model Predictive Safety Certification for Learning-Based Control". In: *IEEE Transactions on Automatic Control* 67.1 (2022), pp. 176–188. DOI: 10.1109/TAC.2021.3049335.

[50] K. P. Wabersich and M. N. Zeilinger. "Linear Model Predictive Safety Certification for Learning-Based Control". In: *IEEE Conference on Decision and Control (CDC)*. 2018, pp. 7130–7135. DOI: 10.1109/CDC.2018.8619829.

[51] K. P. Wabersich and M. N. Zeilinger. "Nonlinear Learning-Based Model Predictive Control Supporting State and Input Dependent Model Uncertainty Estimates". In: *International Journal of Robust and Nonlinear Control* 31.18 (2021), pp. 8897–8915. DOI: 10.1002/rnc.5688.

[52] K. P. Wabersich and M. N. Zeilinger. "A Predictive Safety Filter for Learning-Based Control of Constrained Nonlinear Dynamical Systems". In: *Automatica* 129 (2021), p. 109597. DOI: 10.1016/j.automatica.2021.109597.

[53] L. P. Kaelbling, M. L. Littman, and A. W. Moore. "Reinforcement Learning: A Survey". In: *Journal of Artificial Intelligence Research* 4.1 (1996), pp. 237–285.

[54] C. Szepesvári. *Algorithms for Reinforcement Learning (Synthesis Lectures on Artificial Intelligence and Machine Learning)*. Springer Cham, 2010, pp. 1–103. DOI: 10.1007/978-3-031-01551-9.

[55] J. C. Spall. *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*. Wiley, 2003.

[56] O. Sigaud and F. Stulp. "Policy Search in Continuous Action Domains: An Overview". In: *Neural Networks* 113 (2019), pp. 28–40. DOI: 10.1016/j.neunet.2019.01.011.

[57] S. Levine and V. Koltun. "Guided Policy Search". In: *Proceedings of the 30th International Conference on Machine Learning (ICML)*. 2013, pp. 1–9.

[58] K. Chua, R. Calandra, R. McAllister, and S. Levine. "Deep Reinforcement Learning in a Handful of Trials Using Probabilistic Dynamics Models". In: *Advances in Neural Information Processing Systems 31 (NeurIPS)*. 2018, pp. 4759–4770.

[59] D. Hafner, T. Lillicrap, J. Ba, and M. Norouzi. "Dream to Control: Learning Behaviors by Latent Imagination". In: *8th International Conference on Learning Representations (ICLR)*. 2020.

[60] M. Janner, J. Fu, M. Zhang, and S. Levine. "When to Trust Your Model: Model-Based Policy Optimization". In: *Advances in Neural Information Processing Systems 32 (NeurIPS)*. 2019, pp. 12519–12530.

[61] T. M. Moerland, J. Broekens, A. Plaat, and C. M. Jonker. "Model-Based Reinforcement Learning: A Survey". In: *Foundations and Trends® in Machine Learning* 16.1 (2023), pp. 1–118. DOI: 10.1561/2200000086.

[62] S. Levine, A. Kumar, G. Tucker, and J. Fu. *Offline Reinforcement Learning: Tutorial, Review, and Perspectives on Open Problems*. Preprint. 2020. DOI: 10.48550/arXiv.2005.01643.

[63] R. F. Prudencio, M. R. O. A. Maximo, and E. L. Colombini. "A Survey on Offline Reinforcement Learning: Taxonomy, Review, and Open Problems". In: *IEEE Transactions on Neural Networks and Learning Systems* (2023). DOI: 10.1109/TNNLS.2023.3250269.

[64] X. Chen, C. Wang, Z. Zhou, and K. W. Ross. "Randomized Ensembled Double Q-Learning: Learning Fast Without a Model". In: *9th International Conference on Learning Representations (ICLR)*. 2021.

[65] T. Hiraoka, T. Imagawa, T. Hashimoto, T. Onishi, and Y. Tsuruoka. "Dropout Q-Functions for Doubly Efficient Reinforcement Learning". In: *10th International Conference on Learning Representations (ICLR)*. 2022.

[66] R. E. Bellman. *Adaptive Control Processes*. Princeton University Press, 1961.

[67] C. M. Bishop and H. Bishop. *Deep Learning: Foundations and Concepts*. Springer, 2024.

[68] H. Shen. "Towards a Mathematical Understanding of the Difficulty in Learning With Feedforward Neural Networks". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 811–820. DOI: 10.1109/CVPR.2018.00091.

[69] M. Hausknecht and P. Stone. "Deep Recurrent Q-Learning for Partially Observable MDPs". In: *Papers from the AAAI 2015 Fall Symposium, Sequential Decision Making for Intelligent Agents (AAAI)*. 2015.

[70] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver, and K. Kavukcuoglu. "Asynchronous Methods for Deep Reinforcement Learning". In: *Proceedings of The 33rd International Conference on Machine Learning (ICML)*. 2016, pp. 1928–1937.

[71] S. Hochreiter and J. Schmidhuber. "Long Short-Term Memory". In: *Neural Computation* 9.8 (1997), pp. 1735–1780. DOI: 10.1162/neco.1997.9.8.1735.

[72] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. "Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 2014, pp. 1724–1734. DOI: 10.3115/v1/D14-1179.

[73] D. P. Bertsekas. *Lessons from AlphaZero for Optimal, Model Predictive, and Adaptive Control*. Athena Scientific, 2022.

[74] V. Konda and J. Tsitsiklis. "Actor-Critic Algorithms". In: *Advances in Neural Information Processing Systems 12 (NIPS)*. 1999, pp. 1008–1014.

[75] R. J. Williams. "Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning". In: *Machine Learning* 8 (1992), pp. 229–256.

[76] R. S. Sutton, D. McAllester, S. Singh, and Y. Mansour. "Policy Gradient Methods for Reinforcement Learning with Function Approximation". In: *Advances in Neural Information Processing Systems 12 (NIPS)*. 1999, pp. 1057–1063.

[77] J. Peters and S. Schaal. "Reinforcement Learning of Motor Skills with Policy Gradients". In: *Neural Networks* 21.4 (2008), pp. 682–697. DOI: 10.1016/j.neunet.2008.02.003.

[78] A. Kuznetsov, P. Shvechikov, A. Grishin, and D. Vetrov. "Controlling Overestimation Bias with Truncated Mixture of Continuous Distributional Quantile Critics". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 2020, pp. 5556–5566.

[79] M. Andrychowicz, F. Wolski, A. Ray, J. Schneider, R. Fong, P. Welinder, B. McGrew, J. Tobin, O. Pieter Abbeel, and W. Zaremba. "Hindsight Experience Replay". In: *Advances in Neural Information Processing Systems 30 (NIPS)*. 2017, pp. 5048–5058.

[80] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. Lowe. "Training Language Models to Follow Instructions with Human Feedback". In: *Advances in Neural Information Processing Systems 35*. 2022, pp. 27730–27744.

[81] J. Nocedal and S. J. Wright. *Numerical Optimization*. 2nd Edition. Springer New York, 2006. DOI: 10.1007/978-0-387-40065-5.

[82] Y. Duan, X. Chen, R. Houthooft, J. Schulman, and P. Abbeel. "Benchmarking Deep Reinforcement Learning for Continuous Control". In: *Proceedings of the 33rd International Conference on Machine Learning (ICML)*. 2016, pp. 1329–1338.

[83] P. W. Glynn. "Likelihood Ratio Gradient Estimation for Stochastic Systems". In: *Communications of the ACM* 33.10 (1990), pp. 75–84. DOI: 10.1145/84537.84552.

[84] J. Schulman, P. Moritz, S. Levine, M. I. Jordan, and P. Abbeel. "High-Dimensional Continuous Control Using Generalized Advantage Estimation". In: *4th International Conference on Learning Representations (ICLR)*. 2016.

[85] S. M. Kakade and J. Langford. "Approximately Optimal Approximate Reinforcement Learning". In: *Proceedings of the 19th International Conference on Machine Learning (ICML)*. 2002.

[86] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz. "Trust Region Policy Optimization". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. 2015, pp. 1889–1897.

[87] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine. "Soft Actor-Critic: Off-Policy Maximum Entropy Deep Reinforcement Learning with a Stochastic Actor". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018, pp. 1856–1865.

[88] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra. "Continuous Control with Deep Reinforcement Learning". In: *4th International Conference on Learning Representations (ICLR)*. 2016.

[89] S. Fujimoto, H. van Hoof, and D. Meger. "Addressing Function Approximation Error in Actor-Critic Methods". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018, pp. 1582–1591.

[90] D. P. Kingma and J. Ba. "Adam: A Method for Stochastic Optimization". In: *3rd International Conference on Learning Representations (ICLR)*. 2015.

[91] A. Graves. *Generating Sequences With Recurrent Neural Networks*. Preprint. 2013. DOI: 10.48550/arXiv.1308.0850.

[92] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. *Proximal Policy Optimization Algorithms*. Preprint. 2017. DOI: 10.48550/ARXIV.1707.06347.

[93] T. Haarnoja, A. Zhou, K. Hartikainen, G. Tucker, S. Ha, J. Tan, V. Kumar, H. Zhu, A. Gupta, P. Abbeel, and S. Levine. *Soft Actor-Critic Algorithms and Applications*. Preprint. 2018.

[94] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. WU. "The Surprising Effectiveness of PPO in Cooperative Multi-Agent Games". In: *Advances in Neural Information Processing Systems 35 (NeurIPS)*. 2022, pp. 24611–24624.

[95] K. Zhang, Z. Yang, and T. Başar. "Multi-Agent Reinforcement Learning: A Selective Overview of Theories and Algorithms". In: *Handbook of Reinforcement Learning and Control*. Springer International Publishing, 2021, pp. 321–384. DOI: 10.1007/978-3-030-60990-0_12.

[96] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castañeda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman, N. Sonnerat, T. Green, L. Deason, J. Z. Leibo, D. Silver, D. Hassabis, K. Kavukcuoglu, and T. Graepel. "Human-Level Performance in 3D Multiplayer Games with Population-Based Reinforcement Learning". In: *Science* 364.6443 (2019), pp. 859–865. DOI: 10.1126/science.aau6249.

[97] S. Zheng, A. Trott, S. Srinivasa, D. C. Parkes, and R. Socher. "The AI Economist: Taxation Policy Design via Two-Level Deep Multiagent Reinforcement Learning". In: *Science Advances* 8.18 (2022), eabk2607. DOI: 10.1126/sciadv.abk2607.

[98] M. L. Littman. "Markov Games As a Framework for Multi-agent Reinforcement Learning". In: *Proceedings of the 11th International Coference on Machine Learning (ICML)*. 1994, pp. 157–163.

[99] D. S. Bernstein, R. Givan, N. Immerman, and S. Zilberstein. "The Complexity of Decentralized Control of Markov Decision Processes". In: *Mathematics of Operations Research* 27.4 (2002), pp. 819–840. DOI: 10.1287/moor.27.4.819.297.

[100] F. A. Oliehoek and C. Amato. *A Concise Introduction to Decentralized POMDPs*. 1st. Springer Cham, 2016. DOI: 10.1007/978-3-319-28929-8.

[101] G. Tucker, S. Bhupatiraju, S. Gu, R. Turner, Z. Ghahramani, and S. Levine. "The Mirage of Action-Dependent Baselines in Reinforcement Learning". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018, pp. 5015–5024.

[102] J. Rissanen. "Modeling by Shortest Data Description". In: *Automatica* 14.5 (1978), pp. 465–471. DOI: 10.1016/0005-1098(78)90005-5.

[103] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza. "Deep Drone Racing: From Simulation to Reality with Domain Randomization". In: *IEEE Transactions on Robotics* (2019), pp. 1–14. DOI: 10.1109/TRO.2019.2942989.

[104] B. Mehta, M. Diaz, F. Golemo, C. J. Pal, and L. Paull. "Active Domain Randomization". In: *Proceedings of the 4th Conference on Robot Learning (CoRL)*. 2020, pp. 1162–1176.

[105] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel. "Sim-to-Real Transfer of Robotic Control with Dynamics Randomization". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 3803–3810. DOI: 10.1109/ICRA.2018.8460528.

[106] J. Siekmann, S. Valluri, J. Dao, F. Bermillo, H. Duan, A. Fern, and J. Hurst. "Learning Memory-Based Control for Human-Scale Bipedal Locomotion". In: *Proceedings of Robotics: Science and Systems (RSS)*. 2020. DOI: 10.15607/RSS.2020.XVI.031.

[107] W. Zhao, J. P. Queralta, and T. Westerlund. "Sim-to-Real Transfer in Deep Reinforcement Learning for Robotics: A Survey". In: *IEEE Symposium Series on Computational Intelligence (SSCI)*. 2020, pp. 737–744. DOI: 10.1109/ssci47803.2020.9308468.

[108] A. Rajeswaran, S. Ghotra, S. Levine, and B. Ravindran. "EPOpt: Learning Robust Neural Network Policies Using Model Ensembles". In: *5th International Conference on Learning Representations (ICLR)*. 2017.

[109] C. Tessler, Y. Efroni, and S. Mannor. "Action Robust Reinforcement Learning and Applications in Continuous Control". In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 2019, pp. 6215–6224.

[110] Q. Shen, Y. Li, H. Jiang, Z. Wang, and T. Zhao. "Deep Reinforcement Learning with Robust and Smooth Policy". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 2020, pp. 8707–8718.

[111] E. Aljalbout, F. Frank, M. Karl, and P. van der Smagt. *On the Role of the Action Space in Robot Manipulation Learning and Sim-to-Real Transfer*. Preprint. 2023. DOI: 10.48550/arXiv.2312.03673.

[112] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza. "A Benchmark Comparison of Learned Control Policies for Agile Quadrotor Flight". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2022, pp. 10504–10510. DOI: 10.1109/ICRA46639.2022.9811564.

[113] T. Haarnoja, B. Moran, G. Lever, S. H. Huang, D. Tirumala, M. Wulfmeier, J. Humplik, S. Tunyasuvunakool, N. Y. Siegel, R. Hafner, M. Bloesch, K. Hartikainen, A. Byravan, L. Hasenclever, Y. Tassa, F. Sadeghi, N. Batchelor, F. Casarini, S. Saliceti, C. Game, N. Sreendra, K. Patel, M. Gwira, A. Huber, N. Hurley, F. Nori, R. Hadsell, and N. Heess. *Learning Agile Soccer Skills for a Bipedal Robot with Deep Reinforcement Learning*. Preprint. 2023. DOI: 10.48550/arXiv.2304.13653.

[114] D. Pathak, P. Agrawal, A. A. Efros, and T. Darrell. "Curiosity-Driven Exploration by Self-Supervised Prediction". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, pp. 2778–2787.

[115] R. Houthooft, X. Chen, X. Chen, Y. Duan, J. Schulman, F. De Turck, and P. Abbeel. "VIME: Variational Information Maximizing Exploration". In: *Advances in Neural Information Processing Systems 29 (NIPS)*. 2016, pp. 1109–1117.

[116] A. Nagabandi, G. Kahn, R. S. Fearing, and S. Levine. "Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 7559–7566. DOI: 10.1109/ICRA.2018.8463189.

[117] T. Kurutach, I. Clavera, Y. Duan, A. Tamar, and P. Abbeel. "Model-Ensemble Trust-Region Policy Optimization". In: *6th International Conference on Learning Representations (ICLR)*. 2018.

[118] F. Berkenkamp, M. Turchetta, A. Schoellig, and A. Krause. "Safe Model-Based Reinforcement Learning with Stability Guarantees". In: *Advances in Neural Information Processing Systems 30 (NIPS)*. 2017, pp. 908–918.

[119] T. Koller, F. Berkenkamp, M. Turchetta, and A. Krause. "Learning-Based Model Predictive Control for Safe Exploration". In: *IEEE Conference on Decision and Control (CDC)*. 2018, pp. 6059–6066. DOI: 10.1109/CDC.2018.8619572.

[120] R. Islam, P. Henderson, M. Gomrokchi, and D. Precup. *Reproducibility of Benchmarked Deep Reinforcement Learning Tasks for Continuous Control*. Workshop on Reproducibility in Machine Learning (ICML). 2017.

[121] M. Andrychowicz, A. Raichuk, P. Stanczyk, M. Orsini, S. Girgin, R. Marinier, L. Hussenot, M. Geist, O. Pietquin, M. Michalski, S. Gelly, and O. Bachem. "What Matters for On-Policy Deep Actor-Critic Methods? A Large-Scale Study". In: *9th Proceedings of the International Conference on Learning Representations (ICLR)*. 2021.

[122] D. Reda, T. Tao, and M. van de Panne. "Learning to Locomote: Understanding How Environment Design Matters for Deep Reinforcement Learning". In: *Motion, Interaction and Games (MIG)*. 2020, pp. 1–10. DOI: 10.1145/3424636.3426907.

[123] R. Agarwal, M. Schwarzer, P. S. Castro, A. C. Courville, and M. G. Bellemare. "Deep Reinforcement Learning at the Edge of the Statistical Precipice". In: *Advances in Neural Information Processing Systems 34 (NeurIPS)*. 2021, pp. 29304–29320.

[124] S. M. Kakade. "A Natural Policy Gradient". In: *Advances in Neural Information Processing Systems 14 (NIPS)*. 2002, pp. 1531–1538.

[125] J. Baxter and P. L. Bartlett. "Infinite-Horizon Policy-Gradient Estimation". In: *Journal of Artificial Intelligence Research* 15.1 (2001), pp. 319–350.

[126] L. Espeholt, H. Soyer, R. Munos, K. Simonyan, V. Mnih, T. Ward, Y. Doron, V. Firoiu, T. Harley, I. Dunning, S. Legg, and K. Kavukcuoglu. "IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018, pp. 1406–1415.

[127] T. Haarnoja, S. Ha, A. Zhou, J. Tan, G. Tucker, and S. Levine. "Learning to Walk Via Deep Reinforcement Learning". In: *Proceedings of Robotics: Science and Systems (RSS)*. 2019. DOI: 10.15607/RSS.2019.XV.011.

[128] S. Ioffe and C. Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *Proceedings of the 32nd International Conference on Machine Learning (ICML)*. 2015, pp. 448–456.

[129] T. Haarnoja, H. Tang, P. Abbeel, and S. Levine. "Reinforcement Learning with Deep Energy-Based Policies". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*, pp. 1352–1361.

[130] K. He, X. Zhang, S. Ren, and J. Sun. "Delving Deep into Rectifiers: Surpassing Human-Level Performance on ImageNet Classification". In: *IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 1026–1034. DOI: 10.1109/ICCV.2015.123.

[131] X. Glorot and Y. Bengio. "Understanding the Difficulty of Training Deep Feedforward Neural Networks". In: *Proceedings of the 30th International Conference on Artificial Intelligence and Statistics (AISTATS)*. 2010, pp. 249–256.

[132] A. M. Saxe, J. L. McClelland, and S. Ganguli. "Exact Solutions to the Nonlinear Dynamics of Learning in Deep Linear Neural Networks". In: *2nd International Conference on Learning Representations (ICLR)*. 2014.

[133] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza. "Deep Drone Acrobatics". In: *Proceedings of Robotics: Science and Systems (RSS)*. 2020. DOI: 10.15607/RSS.2020.XVI.040.

[134] M. Müller, A. Dosovitskiy, B. Ghanem, and V. Koltun. "Driving Policy Transfer via Modularity and Abstraction". In: *Proceedings of the 2nd Conference on Robot Learning (CoRL)*. 2018, pp. 1–15.

[135] X. B. Peng and M. van de Panne. "Learning Locomotion Skills Using DeepRL: Does the Choice of Action Space Matter?" In: *Proceedings of the ACM SIGGRAPH / Eurographics Symposium on Computer Animation (SIGGRAPH)*. 2017, pp. 1–13. DOI: 10.1145/3099564.3099567.

[136] T. Seyde, I. Gilitschenski, W. Schwarting, B. Stellato, M. A. Riedmiller, M. Wulfmeier, and D. Rus. "Is Bang-Bang Control All You Need? Solving Continuous Control with Bernoulli Policies". In: *Advances in Neural Information Processing Systems 34 (NeurIPS)*. 2021, pp. 27209–27221.

[137] P. Varin, L. Grossman, and S. Kuindersma. "A Comparison of Action Spaces for Learning Manipulation Tasks". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 6015–6021. DOI: 10.1109/IROS40897.2019.8967946.

[138] P. Dhariwal, C. Hesse, O. Klimov, A. Nichol, M. Plappert, A. Radford, J. Schulman, S. Sidor, Y. Wu, and P. Zhokhov. *OpenAI Baselines*. https://github.com/openai/baselines. Accessed: 2024-02-12. 2017.

[139] E. Coumans and Y. Bai. *PyBullet: A Python Module for Physics Simulation for Games, Robotics and Machine Learning*. http://pybullet.org. Accessed: 2024-02-12. 2016.

[140] M. E. Taylor and P. Stone. "Transfer Learning for Reinforcement Learning Domains: A Survey". In: *Journal of Machine Learning Research* 10 (2009), pp. 1633–1685.

[141] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel. "Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2017, pp. 23–30. DOI: 10.1109/IROS.2017.8202133.

[142] J. Tan, T. Zhang, E. Coumans, A. Iscen, Y. Bai, D. Hafner, S. Bohez, and V. Vanhoucke. "Sim-to-Real: Learning Agile Locomotion For Quadruped Robots". In: *Proceedings of Robotics: Science and Systems (RSS)*. 2018. DOI: 10.15607/RSS.2018.XIV.010.

[143] Y. Carson and A. Maria. "Simulation Optimization: Methods and Applications". In: *Proceedings of the 29th Conference on Winter Simulation (WSC)*. 1997, pp. 118–126. DOI: 10.1145/268437.268460.

[144] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox. "Closing the Sim-to-Real Loop: Adapting Simulation Randomization with Real World Experience". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8973–8979. DOI: 10.1109/ICRA.2019.8793789.

[145] P. I. Frazier. *A Tutorial on Bayesian Optimization*. Preprint. 2018. DOI: 10.48550/arXiv.1807.02811.

[146] J. Leike, M. Martic, V. Krakovna, P. A. Ortega, T. Everitt, A. Lefrancq, L. Orseau, and S. Legg. *AI Safety Gridworlds*. Preprint. 2017. DOI: 10.48550/arXiv.1711.09883.

[147] M. Pecka and T. Svoboda. "Safe Exploration Techniques for Reinforcement Learning – An Overview". In: *Modelling and Simulation for Autonomous Systems (MESAS)*. 2014, pp. 357–375.

[148] T. M. Moldovan and P. Abbeel. "Safe Exploration in Markov Decision Processes". In: *Proceedings of the 29th International Coference on Machine Learning (ICML)*. 2012, pp. 1451–1458.

[149] A. D. Ames, S. Coogan, M. Egerstedt, G. Notomista, K. Sreenath, and P. Tabuada. "Control Barrier Functions: Theory and Applications". In: *European Control Conference (ECC)*. 2019, pp. 3420–3431.

[150] S. Gronauer. *Bullet-Safety-Gym: A Framework for Constrained Reinforcement Learning*. Tech. rep. mediaTUM, 2022. DOI: 10.14459/2022md1639974.

[151] J. Ji, B. Zhang, J. Zhou, X. Pan, W. Huang, R. Sun, Y. Geng, Y. Zhong, J. Dai, and Y. Yang. "Safety Gymnasium: A Unified Safe Reinforcement Learning Benchmark". In: *Advances in Neural Information Processing Systems 37 Datasets and Benchmarks Track (NeurIPS)*. 2023.

[152] A. Ray, J. Achiam, and D. Amodei. *Benchmarking Safe Exploration in Deep Reinforcement Learning*. Preprint. 2019.

[153] E. Altman. *Constrained Markov Decision Processes*. CRC Press, 1999.

[154] K. P. Wabersich, A. J. Taylor, J. J. Choi, K. Sreenath, C. J. Tomlin, A. D. Ames, and M. N. Zeilinger. "Data-Driven Safety Filters: Hamilton-Jacobi Reachability, Control Barrier Functions, and Predictive Methods for Uncertain Systems". In: *IEEE Control Systems Magazine* 43.5 (2023), pp. 137–177. DOI: 10.1109/MCS.2023.3291885.

[155] J. Achiam, D. Held, A. Tamar, and P. Abbeel. "Constrained Policy Optimization". In: *Proceedings of the 34th International Conference on Machine Learning (ICML)*. 2017, pp. 22–31.

[156] H. Sikchi, W. Zhou, and D. Held. *Lyapunov Barrier Policy Optimization*. Preprint. 2021. DOI: 10.48550/arXiv.2103.09230.

[157] G. Dalal, K. Dvijotham, M. Vecerik, T. Hester, C. Paduraru, and Y. Tassa. *Safe Exploration in Continuous Action Spaces*. Preprint. 2018. DOI: 10.48550/arXiv.1801.08757.

[158] K. Srinivasan, B. Eysenbach, S. Ha, J. Tan, and C. Finn. *Learning to be Safe: Deep RL with a Safety Critic*. Preprint. 2020. DOI: 10.48550/arXiv.2010.14603.

[159] S. Gronauer, M. Gottwald, and K. Diepold. "The Successful Ingredients of Policy Gradient Algorithms". In: *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*. 2021, pp. 2455–2461. DOI: 10.24963/ijcai.2021/338.

[160] H. Mania, A. Guy, and B. Recht. "Simple Random Search of Static Linear Policies is Competitive for Reinforcement Learning". In: *Advances in Neural Information Processing Systems 31 (NeurIPS)*. 2018, pp. 1800–1809.

[161] A. Rajeswaran, K. Lowrey, E. V. Todorov, and S. M. Kakade. "Towards Generalization and Simplicity in Continuous Control". In: *Advances in Neural Information Processing Systems 30 (NIPS)*. 2017, pp. 6550–6561.

[162] S. Gronauer, M. Kissel, L. Sacchetto, M. Korte, and K. Diepold. "Using Simulation Optimization to Improve Zero-Shot Policy Transfer of Quadrotors". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 10170–10176. DOI: 10.1109/IROS47612.2022.9981229.

[163] S. Gronauer, D. Stümke, and K. Diepold. "Comparing Quadrotor Control Policies for Zero-Shot Reinforcement Learning under Uncertainty and Partial Observability". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 7508–7514. DOI: 10.1109/IROS55552.2023.10341941.

[164] T. Hester and P. Stone. "TEXPLORE: Real-Time Sample-Efficient Reinforcement Learning for Robots". In: *Machine Learning* 90.3 (2013), pp. 385–429. DOI: 10.1007/s10994-012-5322-7.

[165] X. Chen, J. Hu, C. Jin, L. Li, and L. Wang. "Understanding Domain Randomization for Sim-to-real Transfer". In: *10th International Conference on Learning Representations (ICLR)*. 2022.

[166] S. Gronauer, T. Haider, F. Schmoeller da Roza, and K. Diepold. "Reinforcement Learning with Ensemble Model Predictive Safety Certification". In: *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2024.

[167] N. Fulton and A. Platzer. "Safe Reinforcement Learning via Formal Methods: Toward Safe Control Through Proof and Learning". In: *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*. 2018. DOI: 10.1609/aaai.v32i1.12107.

[168] C. Tessler, D. J. Mankowitz, and S. Mannor. "Reward Constrained Policy Optimization". In: *7th International Conference on Learning Representations (ICLR)*. 2019.

[169] L. Yang, J. Ji, J. Dai, L. Zhang, B. Zhou, P. Li, Y. Yang, and G. Pan. "Constrained Update Projection Approach to Safe Policy Optimization". In: *Advances in Neural Information Processing Systems 35 (NeurIPS)*. 2022, pp. 9111–9124.

[170] Y. Zhang, Q. Vuong, and K. Ross. "First Order Optimization in Policy Space for Constrained Deep Reinforcement Learning". In: *Advances in Neural Information Processing Systems 33 (NeurIPS)*. 2020, pp. 15338–15349.

[171] B. Thananjeyan, A. Balakrishna, S. Nair, M. Luo, K. Srinivasan, M. Hwang, J. E. Gonzalez, J. Ibarz, C. Finn, and K. Goldberg. "Recovery RL: Safe Reinforcement Learning With Learned Recovery Zones". In: *IEEE Robotics and Automation Letters* 6.3 (2021), pp. 4915–4922. DOI: 10.1109/LRA.2021.3070252.

[172] T. Li, K. Zhu, N. C. Luong, D. Niyato, Q. Wu, Y. Zhang, and B. Chen. "Applications of Multi-Agent Reinforcement Learning in Future Internet: A Comprehensive Survey". In: *IEEE Communications Surveys and Tutorials* 24.2 (2022), pp. 1240–1279. DOI: 10.1109/COMST.2022.3160697.

[173] N. C. Luong, D. T. Hoang, S. Gong, D. Niyato, P. Wang, Y. Liang, and D. I. Kim. "Applications of Deep Reinforcement Learning in Communications and Networking: A Survey". In: *IEEE Communications Surveys and Tutorials* (2019), pp. 1–1. DOI: 10.1109/COMST.2019.2916583.

[174] S. Gronauer and K. Diepold. "Multi-Agent Deep Reinforcement Learning: A Survey". In: *Artificial Intelligence Review* 55.2 (2022), pp. 895–943. DOI: 10.1007/s10462-021-09996-w.

[175] O. Nachum, M. Ahn, H. Ponte, S. Gu, and V. Kumar. "Multi-Agent Manipulation via Locomotion using Hierarchical Sim2Real". In: *Proceedings of the 4th Conference on Robot Learning (CoRL)*. 2020, pp. 110–121.

[176] P. Hernandez-Leal, B. Kartal, and M. E. Taylor. "A Survey and Critique of Multiagent Deep Reinforcement Learning". In: *Autonomous Agents and Multi-Agent Systems* 33.6 (2019), pp. 750–797. DOI: 10.1007/s10458-019-09421-1.

[177] T. T. Nguyen, N. D. Nguyen, and S. Nahavandi. "Deep Reinforcement Learning for Multiagent Systems: A Review of Challenges, Solutions, and Applications". In: *IEEE Transactions on Cybernetics* 50.9 (2020), pp. 3826–3839. DOI: 10.1109/TCYB.2020.2977374.

[178] Y. Shoham, R. Powers, and T. Grenager. *Multi-Agent Reinforcement Learning: A Critical Survey*. Tech. rep. 2003.

[179] P. Stone and M. Veloso. "Multiagent Systems: A Survey from a Machine Learning Perspective". In: *Autonomous Robots* 8.3 (2000), pp. 345–383. DOI: 10.1023/A:1008942012299.

[180] K. Tuyls and G. Weiss. "Multiagent Learning: Basics, Challenges, and Prospects". In: *AI Magazine* 33.3 (2012), p. 41. DOI: 10.1609/aimag.v33i3.2426.

[181] A. Ilyas, L. Engstrom, S. Santurkar, D. Tsipras, F. Janoos, L. Rudolph, and A. Madry. "A Closer Look at Deep Policy Gradients". In: *8th International Conference on Learning Representations (ICLR)*. 2020.

[182] H. Duan, J. Dao, K. Green, T. Apgar, A. Fern, and J. Hurst. "Learning Task Space Actions for Bipedal Locomotion". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 1276–1282. DOI: 10.1109/ICRA48506.2021.9561705.

[183] S. Chen, B. Zhang, M. W. Mueller, A. Rai, and K. Sreenath. "Learning Torque Control for Quadrupedal Locomotion". In: *IEEE-RAS 22nd International Conference on Humanoid Robots (Humanoids)*. 2023, pp. 1–8. DOI: 10.1109/Humanoids57100.2023.10375154.

[184] J. Schneider, P. Schumacher, D. Häufle, B. Schölkopf, and D. Büchler. *Investigating the Impact of Action Representations in Policy Gradient Algorithms*. Workshop on effective Representations, Abstractions, and Priors for Robot Learning (ICRA). 2023. DOI: 10.48550/arXiv.2309.06921.

[185] R. P. Singh, Z. Xie, P. Gergondet, and F. Kanehiro. "Learning Bipedal Walking for Humanoids With Current Feedback". In: *IEEE Access* 11 (2023), pp. 82013–82023. DOI: 10.1109/ACCESS.2023.3301175.

[186] Z. Liu, X. Li, B. Kang, and T. Darrell. "Regularization Matters in Policy Optimization - An Empirical Study on Continuous Control". In: *9th International Conference on Learning Representations (ICLR)*. 2021.

[187] F. Golemo, A. A. Taiga, A. Courville, and P.-Y. Oudeyer. "Sim-to-Real Transfer with Neural-Augmented Robot Simulation". In: *Proceedings of the 2nd Conference on Robot Learning (CoRL)*. 2018, pp. 817–828.

[188] S. Choi, G. Ji, J. Park, H. Kim, J. Mun, J. H. Lee, and J. Hwangbo. "Learning Quadrupedal Locomotion on Deformable Terrain". In: *Science Robotics* 8.74 (2023), eade2256. DOI: 10.1126/scirobotics.ade2256.

[189] T. Miki, J. Lee, J. Hwangbo, L. Wellhausen, V. Koltun, and M. Hutter. "Learning Robust Perceptive Locomotion for Quadrupedal Robots in the Wild". In: *Science Robotics* 7.62 (2022), eabk2822. DOI: 10.1126/scirobotics.abk2822.

[190] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme. "Sim-to-(Multi)-Real: Transfer of Low-Level Robust Control Policies to Multiple Quadrotors". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2019, pp. 59–66. DOI: 10.1109/IROS40897.2019.8967695.

[191] J. T. Kim and S. Ha. "Observation Space Matters: Benchmark and Optimization Algorithm". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 1527–1534. DOI: 10.1109/ICRA48506.2021.9561019.

[192] Y. Hu, W. Wang, H. Jia, Y. Wang, Y. Chen, J. Hao, F. Wu, and C. Fan. "Learning to Utilize Shaping Rewards: A New Approach of Reward Shaping". In: *Advances in Neural Information Processing Systems 33 (NeurIPS)*. 2020, pp. 15931–15941.

[193] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Debiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse, R. Józefowicz, S. Gray, C. Olsson, J. Pachocki, M. Petrov, H. P. de Oliveira Pinto, J. Raiman, T. Salimans, J. Schlatter, J. Schneider, S. Sidor, I. Sutskever, J. Tang, F. Wolski, and S. Zhang. *Dota 2 with Large Scale Deep Reinforcement Learning*. Preprint. 2019. DOI: 10.48550/arXiv.1912.06680.

[194] G. Lample and D. S. Chaplot. "Playing FPS Games with Deep Reinforcement Learning". In: *Proceedings of the 31st AAAI Conference on Artificial Intelligence (AAAI)*. 2017. DOI: 10.1609/aaai.v31i1.10827.

[195] F. Pardo, A. Tavakoli, V. Levdik, and P. Kormushev. "Time Limits in Reinforcement Learning". In: *Proceedings of the 35th International Conference on Machine Learning (ICML)*. 2018, pp. 4045–4054.

[196] N. Rudin, D. Hoeller, P. Reist, and M. Hutter. "Learning to Walk in Minutes Using Massively Parallel Deep Reinforcement Learning". In: *Proceedings of the 5th Conference on Robot Learning (CoRL)*. 2021, pp. 91–100.

[197] Z. Xie, P. Clary, J. Dao, P. Morais, J. Hurst, and M. van de Panne. "Learning Locomotion Skills for Cassie: Iterative Design and Sim-to-Real". In: *Proceedings of the 4th Conference on Robot Learning (CoRL)*. 2020, pp. 317–329.

[198] M. Kaspar, J. D. Muñoz Osorio, and J. Bock. "Sim2Real Transfer for Reinforcement Learning without Dynamics Randomization". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2020, pp. 4383–4388. DOI: 10.1109/IROS45743.2020.9341260.

[199] J. Josifovski, M. Malmir, N. Klarmann, B. L. Žagar, N. Navarro-Guerrero, and A. Knoll. "Analysis of Randomization Effects on Sim2Real Transfer in Reinforcement Learning for Robotic Manipulation Tasks". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 10193–10200. DOI: 10.1109/IROS47612.2022.9981951.

[200] F. Sadeghi and S. Levine. "CAD2RL: Real Single-Image Flight Without a Single Real Image". In: *Proceedings of Robotics: Science and Systems (RSS)*. 2017. DOI: 10.15607/RSS.2017.XIII.034.

[201] K. Bousmalis, A. Irpan, P. Wohlhart, Y. Bai, M. Kelcey, M. Kalakrishnan, L. Downs, J. Ibarz, P. Pastor, K. Konolige, S. Levine, and V. Vanhoucke. "Using Simulation and Domain Adaptation to Improve Efficiency of Deep Robotic Grasping". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2018, pp. 4243–4250. DOI: 10.1109/ICRA.2018.8460875.

[202] Y. Luo and T. Ma. "Learning Barrier Certificates: Towards Safe Reinforcement Learning with Zero Training-time Violations". In: *Advances in Neural Information Processing Systems 34 (NeurIPS)*. 2021, pp. 25621–25632.

[203] B. Lütjens, M. Everett, and J. P. How. "Safe Reinforcement Learning With Model Uncertainty Estimates". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2019, pp. 8662–8668. DOI: 10.1109/ICRA.2019.8793611.

[204] J. Zhang, B. Cheung, C. Finn, S. Levine, and D. Jayaraman. "Cautious Adaptation for Reinforcement Learning in Safety-Critical Settings". In: *Proceedings of the 37th International Conference on Machine Learning (ICML)*. 2020, pp. 11055–11065.

[205] R. Cheng, G. Orosz, R. M. Murray, and J. W. Burdick. "End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks". In: *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*. 2019, pp. 3387–3395. DOI: 10.1609/aaai.v33i01.33013387.

[206] Z. Zhou, O. S. Oguz, M. Leibold, and M. Buss. "Learning a Low-Dimensional Representation of a Safe Region for Safe Reinforcement Learning on Dynamical Systems". In: *IEEE Transactions on Neural Networks and Learning Systems* 34.5 (2023), pp. 2513–2527. DOI: 10.1109/TNNLS.2021.3106818.

[207] C. Xie, S. Patil, T. Moldovan, S. Levine, and P. Abbeel. "Model-Based Reinforcement Learning with Parametrized Physical Models and Optimism-Driven Exploration". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2016, pp. 504–511. DOI: 10.1109/ICRA.2016.7487172.

[208] M. Cutler and J. P. How. "Efficient Reinforcement Learning for Robots Using Informative Simulated Priors". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2015, pp. 2605–2612. DOI: 10.1109/ICRA.2015.7139550.

[209] D. Nguyen-Tuong and J. Peters. "Using Model Knowledge for Learning Inverse Dynamics". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2010, pp. 2677–2682. DOI: 10.1109/ROBOT.2010.5509858.

[210] D. Hafner, T. Lillicrap, I. Fischer, R. Villegas, D. Ha, H. Lee, and J. Davidson. "Learning Latent Dynamics for Planning from Pixels". In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 2019, pp. 2555–2565.

[211] M. Janner, Q. Li, and S. Levine. "Offline Reinforcement Learning as One Big Sequence Modeling Problem". In: *Advances in Neural Information Processing Systems (NeurIPS)*. 2021, pp. 1273–1286.

[212] Y. Seo, D. Hafner, H. Liu, F. Liu, S. James, K. Lee, and P. Abbeel. "Masked World Models for Visual Control". In: *Proceedings of The 6th Conference on Robot Learning (CoRL)*. 2023, pp. 1332–1344.

[213] L. M. Smith, I. Kostrikov, and S. Levine. "A Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning". In: *Proceedings of Robotics: Science and Systems (RSS)*. 2023.

[214] P. Wu, A. Escontrela, D. Hafner, P. Abbeel, and K. Goldberg. "DayDreamer: World Models for Physical Robot Learning". In: *Proceedings of the 6th Conference on Robot Learning (CoRL)*. 2022, pp. 2226–2240.

[215] Y. Du, O. Watkins, T. Darrell, P. Abbeel, and D. Pathak. "Auto-Tuned Sim-to-Real Transfer". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 1290–1296. DOI: 10.1109/ICRA48506.2021.9562091.

[216] T. G. Thuruthel, E. Falotico, F. Renda, and C. Laschi. "Model-Based Reinforcement Learning for Closed-Loop Dynamic Control of Soft Robotic Manipulators". In: *IEEE Transactions on Robotics* 35.1 (2019), pp. 124–134. DOI: 10.1109/TRO.2018.2878318.

[217] G. Novati and P. Koumoutsakos. "Remember and Forget for Experience Replay". In: *Proceedings of the 36th International Conference on Machine Learning (ICML)*. 2019, pp. 4851–4860.

[218] A. Gupta, J. Yu, T. Z. Zhao, V. Kumar, A. Rovinsky, K. Xu, T. Devlin, and S. Levine. "Reset-Free Reinforcement Learning via Multi-Task Learning: Learning Dexterous Manipulation Behaviors without Human Intervention". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2021, pp. 6664–6671. DOI: 10.1109/ICRA48506.2021.9561384.

[219] V. Makoviychuk, L. Wawrzyniak, Y. Guo, M. Lu, K. Storey, M. Macklin, D. Hoeller, N. Rudin, A. Allshire, A. Handa, and G. State. "Isaac Gym: High Performance GPU Based Physics Simulation For Robot Learning". In: *Advances in Neural Information Processing Systems 34 Track on Datasets and Benchmarks (NeurIPS)*. 2021.

[220] C. Dawson, Z. Qin, S. Gao, and C. Fan. "Safe Nonlinear Control Using Robust Neural Lyapunov-Barrier Functions". In: *Proceedings of the 5th Conference on Robot Learning (CoRL)*. 2022, pp. 1724–1735.

[221] S. Liu, C. Liu, and J. Dolan. "Safe Control Under Input Limits with Neural Control Barrier Functions". In: *Proceedings of the 6th Conference on Robot Learning (CoRL)*. 2023, pp. 1970–1980.

[222] T.-Y. Yang, J. Rosca, K. Narasimhan, and P. J. Ramadge. "Projection-Based Constrained Policy Optimization". In: *8th International Conference on Learning Representations (ICLR)*. 2020.

[223] Y. Song, A. Romero, M. Müller, V. Koltun, and D. Scaramuzza. "Reaching the Limit in Autonomous Racing: Optimal Control versus Reinforcement Learning". In: *Science Robotics* 8.82 (2023), eadg1462. DOI: 10.1126/scirobotics.adg1462.

[224] K. Khetarpal, M. Riemer, I. Rish, and D. Precup. "Towards Continual Reinforcement Learning: A Review and Perspectives". In: *Journal of Artificial Intelligence Research* 75 (2022), pp. 1401–1476. DOI: 10.1613/jair.1.13673.

[225] M. W. Mueller, S. J. Lee, and R. D'Andrea. "Design and Control of Drones". In: *Annual Review of Control, Robotics, and Autonomous Systems* 5.1 (2022), pp. 161–177. DOI: 10.1146/annurev-control-042920-012045.

[226] M. Suomalainen, Y. Karayiannidis, and V. Kyrki. "A Survey of Robot Manipulation in Contact". In: *Robotics and Autonomous Systems* 156 (2022), p. 104224. DOI: 10.1016/j.robot.2022.104224.

Part A

Single-Agent Reinforcement Learning

Paper I

The Successful Ingredients of Policy Gradient Algorithms

Sven Gronauer, Martin Gottwald, Klaus Diepold

Abstract

Despite the sublime success in recent years, the underlying mechanisms powering the advances of reinforcement learning are yet poorly understood. In this paper, we identify these mechanisms - which we call ingredients - in on-policy policy gradient methods and empirically determine their impact on the learning. To allow an equitable assessment, we conduct our experiments based on a unified and modular implementation. Our results underline the significance of recent algorithmic advances and demonstrate that reaching state-of-the-art performance may not need sophisticated algorithms but can also be accomplished by the combination of a few simple ingredients.

S. Gronauer, M. Gottwald, and K. Diepold. "The Successful Ingredients of Policy Gradient Algorithms". In: *Proceedings of the 30th International Joint Conference on Artificial Intelligence (IJCAI)*. 2021, pp. 2455–2461. DOI: 10.24963/ijcai.2021/338

Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence (IJCAI-21)

The Successful Ingredients of Policy Gradient Algorithms

Sven Gronauer* , **Martin Gottwald** , **Klaus Diepold**
Technical University of Munich, Germany
{sven.gronauer, martin.gottwald, kldi}@tum.de

Abstract

Despite the sublime success in recent years, the underlying mechanisms powering the advances of reinforcement learning are yet poorly understood. In this paper, we identify these mechanisms - which we call ingredients - in on-policy policy gradient methods and empirically determine their impact on the learning. To allow an equitable assessment, we conduct our experiments based on a unified and modular implementation. Our results underline the significance of recent algorithmic advances and demonstrate that reaching state-of-the-art performance may not need sophisticated algorithms but can also be accomplished by the combination of a few simple ingredients.

1 Introduction

Reinforcement learning (RL) is a data-driven paradigm that can be leveraged to learn complex strategies for controlling dynamical systems. RL algorithms excel at problems that can be simulated or where exact models are known, but conventional planning is not feasible, e.g. in the game of Go [Silver *et al.*, 2016]. Although the RL domain has witnessed significant advances in recent years [Arulkumaran *et al.*, 2017; Mnih *et al.*, 2015], the progress is aggravated by various impediments. First, experiments are difficult to reproduce because numerical results are sensitive to the selection of hyper-parameters and can vary over different random seeds [Henderson *et al.*, 2018; Islam *et al.*, 2017]. More drastically, different code bases produce inconsistent results, although describing the same algorithm. Second, a major share of the claimed performance increments in recently proposed methods is less achieved by innovative algorithmic properties but more through clever implementation [Engstrom *et al.*, 2020; Tucker *et al.*, 2018]. Third, state-of-the-art algorithms based on neural networks can be disputed by simpler learning models. Despite their expressiveness, works demonstrated that representations like radial basis functions or linear function mappings could achieve similar results on contemporary benchmarks [Mania *et al.*, 2018; Rajeswaran *et al.*, 2017].

Until yet, it is poorly understood which underlying mechanisms drive the learning of RL agents. The intricate interplay between different algorithm components and the abundance of adjustable parameters render it difficult to study the roots of the recent progress. Our understanding remains opaque until we assess the importance of new algorithmic innovations through careful analysis. First works investigated the underlying mechanisms by conducting ablation studies in large-scale experiments [Andrychowicz *et al.*, 2020] or on a selected subset of parameters [Engstrom *et al.*, 2020]. To maintain sustainable progress, the RL community must build a profound understanding of the ingredients and their respective proportion to the learning success, individually and as a whole. Empirical as well as theoretical analysis about why an algorithm surpasses the other is therefore crucial as argued by Sigaud and Stulp [2019].

In this paper, we shed light on the components - which we denote as *ingredients* - powering on-policy policy gradient algorithms in continuous control problems.[1] Our contribution is two-fold: (1) we empirically study the significance of specific ingredients and show the roots of algorithmic progress based on a modular and unified code base and (2) identify a minimal setup of ingredients that challenges state-of-the-art approaches while exhibiting a manageable size of algorithm complexity. According to the principle of Occam's Razor,[2] a simple baseline is preferable because fewer hyper-parameters must be tuned and, thus, is easier to reproduce.

2 Related Work

A plethora of literature has emerged in recent years that criticizes the reproducibility of RL. First of all, [Islam *et al.*, 2017] investigated the influence of hyper-parameter choices and revealed the inherent performance variance of RL algorithms. The authors objected to the under-reporting of hyper-parameters, which are crucial for a fair comparison between different algorithms, and provided recommendations for good research practice. In a similar vein, [Henderson *et al.*, 2018]

*Contact Author

[1] For the supplemental materials and the implementation see: https://github.com/SvenGronauer/successful-ingredients-paper

[2] We associate the minimum description length as one form of interpretation with the principle of Occam's Razor, which states the trade-off between the quality of data fitting and the complexity of the used algorithm model [Rissanen, 1978].

emphasized the intricate interplay between hyper-parameter selections and underlined the importance of independent random seeds used to evaluate experiment trials. Further, Henderson *et al.* [2018] pointed out that different code bases, although describing the same underlying algorithm, can yield inconsistent results, and they gave evidence that these performance deviations can be attributed to implementation details which are often not neatly reported. Related to performance disparities caused by different code bases, Engstrom *et al.* [2020] showed that the implementation details indeed matter. The authors investigated selected code-level ingredients, which are found only in the implementation or described in detail in the appendices, and showed that these could be accounted for the performance gains observed between the TRPO and PPO algorithm. Alike, Tucker *et al.* [2018] observed that recent advances in gradient estimation, which were originally related to algorithmic improvements, can be attributed to subtle implementation details.

Similar to our empirical research, Reda *et al.* [2020] investigated the impact of the environment design on the learned policy performance in locomotion tasks, showing that the problem design is at least as important as the selected algorithm. Most related to our work, Andrychowicz *et al.* [2020] discussed the effect of ingredients on on-policy RL algorithms through the conduction of large-scale experiments. They provided practical recommendations for high and low-level choices and corresponding hyper-parameters values but, in contrast to our work, conducted only ablation studies and no incremental approach.

3 Background

We consider model-free policy gradient algorithms in continuous control problems where a policy is searched by interacting with an environment with unknown dynamics. We make use of the standard formulation for discounted infinite-horizon Markov decision processes (MDP) which are formalized by the tuple $(\mathcal{X}, \mathcal{U}, \mathcal{P}, r, \gamma)$, where $\mathcal{X}$ and $\mathcal{U}$ denote state and action space, respectively. The system transition probability is described by $\mathcal{P}$ while r is the reward function and γ denotes the discount factor. The agent's goal is to learn a policy $\pi : \mathcal{X} \to \mathcal{P}(\mathcal{U})$ that maximizes the expected return

$$J(\pi) = \int_{\mathcal{X}} \rho_\pi(x) \int_{\mathcal{U}} \pi(u|x)\, r(x,u)\, du\, dx. \tag{1}$$

Under the assumption of an ergodic MDP and an infinite horizon, the problem is stationary and the expected policy performance can be calculated over the un-normalized steady state distribution $\rho_\pi(x) = \gamma^0 P(x_0 = x) + \gamma^1 P(x_1 = x) + \dots$ under policy π. The policy π_θ is represented by a neural network that is parametrized by the vector θ and assumed to be at least once differentiable. We denote the state-value function as $V_{\pi_\theta}(x) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, u_t) \mid x_0 = x\right]$, and similarly $Q_{\pi_\theta}(x,u) = \mathbb{E}_{\pi_\theta}\left[\sum_{t=0}^{\infty} \gamma^t r(x_t, u_t) \mid x_0 = x, u_0 = u\right]$ as the action-value function. Policy gradient methods optimize the learning objective by building the gradient of the expected return with respect to the policy parameters $\nabla_\theta J(\pi_\theta)$ and update the policy parameters by taking a step along the gradient direction. Likelihood ratio methods require a re-sampling of data for every gradient step. To reuse generated trajectory data over multiple iterations, importance sampling can be applied to perform updates based on a local approximation

$$\hat{J}(\pi_\theta) = \mathbb{E}_\mu\left[\frac{\pi_\theta(u|x)}{\mu(u|x)} \Psi(x,u)\right] \tag{2}$$

that matches J to first order. Off-policy samples can be incorporated under the assumption that $\mu(\cdot|x) = 0 \Rightarrow \pi_\theta(\cdot|x) = 0$ for all x. However, the estimation of the policy gradient can exhibit high variance. Thus, a reward estimator Ψ is typically used which can be expressed by several terms as proposed by Schulman *et al.* [2016] but typically takes the form of the advantage function $A_{\pi_\theta}(x,u) = Q_{\pi_\theta}(x,u) - V_{\pi_\theta}(x)$.

4 Methodology

In this paper, we consider on-policy gradient methods[3] in continuous control domains. We take the importance-weighted policy gradient (IWPG) objective from Eq. (2) as the basis and systematically add ingredients to observe their impact on the learned policy performance. We set up our experiments in an incremental and modular approach such that the influence of individual ingredients becomes transparent. In particular, we study ingredients chronologically in three stages:

1. *Code-level ingredients* are enhancements to the algorithm, which are provided as supplementary details or can only be found in the implementation. Most of these ingredients are considered as good practices in the RL community and are not regarded in the hyper-parameter search, leaving their impact on the learning unexplored.
2. *Algorithmic ingredients* lie at the heart of new algorithm proposals and depict the core innovation. These ingredients are precisely described in theoretical terms, but the realization may only become transparent from the provided implementation code.
3. *Structural ingredients* refer to choices and hyper-parameters that describe the neural network architecture and the optimization. We consider those ingredients that can vary between different implementation frameworks and are often neglected in the experimental description.

Our experiments are conducted in the procedure from above to cope with two challenges. First, ingredients cross-correlate, making it difficult to determine their causal direction of influence. We alleviate cross-correlations by randomly adding ingredients and assess their impact based on the performance difference, whether an ingredient is applied or not. Second, the combination of configurations grows exponentially with the number of tunable hyper-parameters. We try to contain this through the proposed experimental structure in three stages.

[3] As on-policy, we also denote policy iteration algorithms that collect data based on the current policy and update a policy several times by using the same data through importance weighting before generating new data with the iterated policy. Transition samples become off-policy after the first policy update but are still close to the generating policy.

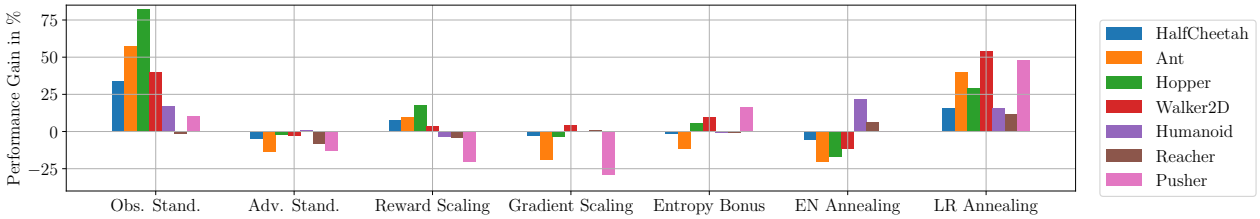


Figure 1: The impact of code-level ingredients on the policy performance. The normalized scores show the relative change of the policy performance when a particular ingredient is used. Note that these numbers are generated based on the IWPG objective without algorithmic ingredients such as trust-regions, which are added thereafter in Experiments 5.2. The Kuka task is excluded since we were not able to improve upon a random policy without advanced variance reduction methods.

To benchmark the performance in continuous control problems, we use the five locomotion environments HalfCheetah, Hopper, Ant, Walker2D, and Humanoid as well as the three robotic manipulations tasks Reacher, Pusher, and Kuka. We measure the expected return after 10^7 environment interactions when exploration noise is disabled. All eight tasks are evaluated in the PyBullet physics engine [Coumans and Bai, 2016].

5 Experiments

In this section, we explain the details of our experiments and present the results. We aligned the hyper-parameters to the ones suggested in Henderson *et al.* [2018]. We applied a single learner setup, used as discount factor $\gamma = 0.99$, collected batches of size 32000 for each policy iteration, and ran each seed over a total of 10^7 environment interactions. For the neural networks, we used the same structure for both policy and value networks, i.e. multi-layer perceptrons with two hidden layers consisting of 64 neurons each followed by tanh non-linearities. The default optimizer was Adam [Kingma and Ba, 2015] for the policy and value network, respectively. Our studied ingredients are only applied to the policy network but not to the value network. For all experiments, we performed hyper-parameter grid searches to average the effect of ingredients over a wide range of hyper-parameters (Experiments 5.1) or to determine the best hyper-parameter configuration (Experiments 5.2-5.4). An overview of the used hyper-parameters and configurations is provided in the supplemental.

5.1 Impact of Code-level Ingredients

Being augmentations to the algorithmic core, code-level ingredients are provided as supplementary details or can only be found in implementations. Although not grounded on theoretic insights, many code-level ingredients are used as a heuristic on current benchmarks. We agree on these ingredients as the first stage since they can be applied to all on-policy gradient algorithms and are leveraged in the succeeding experiments. We investigated the seven following ingredients:

1. *Observation Standardization.* Each batch of observations is made mean-free and re-scaled to unit variance by first subtracting the mean value and then dividing through the standard deviation.
2. *Advantage Standardization.* Analogous to observation standardization, each batch is transformed into a mean-free and unit variance distribution.
3. *Reward Scaling.* The received rewards are divided by the standard deviation of a running discounted sum of rewards.
4. *Gradient Scaling.* To maintain the magnitude, parameter gradients (as they are concatenated into a single vector) are re-scaled if they exceed a certain threshold.
5. *Entropy Bonus Term.* To promote exploration, the entropy bonus term can be added to the optimization objective, which encourages uniformly distributed actions.
6. *Exploration Noise Annealing.* Outputs of the policy networks typically represent the mean of a multivariate Gaussian distribution. Through exploration noise annealing, the entries of the covariance matrix are linearly decreased over the training.
7. *Learning Rate Annealing.* The learning rate of the policy optimizer is linearly decayed to zero over the training.

Approach. To study how code-level ingredients affect learning, we randomly added each ingredient to the IWPG objective from (2) by a chance of 50% in each run. We used the random enabling to remedy cross-correlations between the investigated ingredients. We conducted experiments over a 4×4 grid of learning rate and number of policy iterations combinations. Each combination was evaluated over 16 independent runs, resulting in a total of 256 random seeds for each environment. The impact of each code-level ingredients was determined by $(J_\iota^+ - J_\chi)/(J_\iota^- - J_\chi)$ where J_ι^+ denotes the average policy performance when ingredient ι is applied, and J_ι^- when ingredient ι is not used, and J_χ is the performance of a uniform random policy χ. We used plain advantage estimation $\Psi = A$ as the reward estimator.

Results. As shown in Figure 1, observation standardization and learning rate annealing significantly affect the learned policy performance. Both ingredients yield in at least six out of seven tasks performance gains, where observation standardization yielded an average performance increase of 34.3% and learning rate annealing an improvement of 30.7%. Reward Scaling showed in four out of five locomotion tasks performance gains, while for Humanoid a negative score.

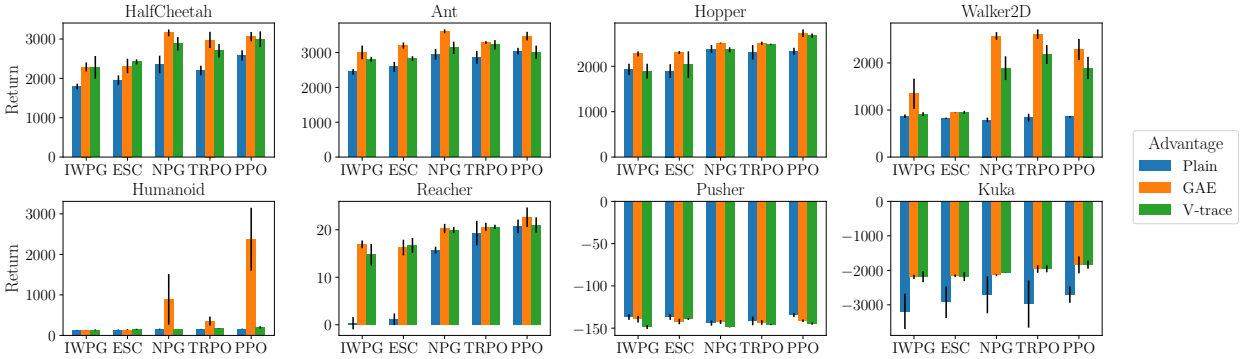


Figure 2: Impact of the algorithmic ingredients on the policy performance. Different trust-region enforcement methods are compared conditioned on the variance reduction method of the critic estimates. Only the best results obtained through grid search are depicted.

The manipulation tasks did neither profit from reward scaling. The entropy bonus term had a mixed impact, showing in Pusher and Walker gains of at least 9.7%, whereas Ant denotes a performance loss of 11.6%. The annealing of the exploration noise showed in four out of five tasks a negative impact, but performance gains in Humanoid, Reacher, and Pusher. In contrast to the former ingredients, both the scaling of policy gradients and the standardization of advantages showed on average performance losses. Note that we could not achieve an improvement over a random policy at the Kuka task without advanced variance reduction methods and, thus, we excluded the Kuka environment from Figure 1. The plots visualizing the learning curves and the numerical results of all eight tasks can be taken from the supplemental.

5.2 Effect of Algorithmic Ingredients

In this section, we elaborate on the former experiment's code-level findings and endow the learning objective with *algorithmic ingredients*, which are claimed to be responsible for the state-of-the-art achievements.

Jumps in policy parameter updates can deteriorate the learned performance and may eventually cause policy collapse [Duan *et al.*, 2016]. However, well-chosen policy updates can alleviate this issue by constraining the distance between consecutive policy iterates through *trust-region enforcement*. We investigated the following four methods:

1. *IWPG with an Early Stopping Criterion* (ESC) measures the KL divergence between consecutive policy iterates and terminates the updates when the distance criterion is met.
2. *Natural Policy Gradient* (NPG) [Kakade, 2002] regards the curvature of the policy parameter space by approximating the Fisher information matrix. The update direction is given by the matrix-vector multiplication between the Fisher information matrix and the policy gradient.
3. *Trust-region Policy Optimization* (TRPO) [Schulman *et al.*, 2015] optimizes over a surrogate function, which matches the original objective to first order. In contrast to NPG, TRPO explicitly regards the KL divergence between policy iterates by performing a backtracking line search until the distance criterion is fulfilled.
4. *Proximal Policy Optimization* (PPO) [Schulman *et al.*, 2017] utilizes clipped probability ratios to maintain the step size in the parameter space. The resulting surrogate objective is then optimized to first order.

Low variance critic estimates are considered as crucial component for good policy performance [Peters and Schaal, 2008]. In our experiments we studied three techniques for the *Variance Reduction of Critic Estimations*:

1. *Plain Advantage Estimation* is used as baseline where the advantages are determined by $A(x_t, u_t) = r(x_t, u_t) + \gamma V_\pi(x_{t+1}) - V_\pi(x_t)$.
2. *Generalized Advantage Estimation* (GAE) [Schulman *et al.*, 2016] aims to reduce variance in critic estimates at the cost of introducing bias. Advantages are calculated by the weighted sum $A(x_t, u_t) = \sum_{k=0}^{n-1} (\lambda\gamma)^k \, \delta_{t+k}$ of temporal differences $\delta_{t+k} = r(x_{t+k}, u_{t+k}) + \gamma V_\pi(x_{t+k+1}) - V_\pi(x_{t+k})$. The scalar λ governs the trade-off between variance and bias.
3. *V-trace* [Espeholt *et al.*, 2018] was introduced to account for off-policy critic updates in distributed architectures to compensate policy lags. Updates follow $A(x_t, u_t) = r(x_t, u_t) + \gamma v(x_{t+1}) - V(x_t)$ while the values become $v(x_t) = V(x_t) + \sum_{s=t}^{t+n-1} \gamma^{s-t} \left(\prod_{i=t}^{s-1} c_i \right) \delta_s V$.

Approach. We continued to apply observation standardization and linear learning rate decay as default code-level ingredients. Reward scaling was added only to the locomotion tasks but not to the manipulation tasks. The experiments were run over all possible combinations of trust-region enforcement and variance reduction methods. We conducted a grid search over learning rates and numbers of policy iterations and determined the best configuration according to the highest $\text{mean}(J) - \text{std}(J)$ averaged over four independent seeds (see detailed hyper-parameters in the supplemental). As a reference for comparison, we used the plain IWPG objective from Experiments 5.1 which makes no use of a trust-region enforcement and uses plain advantage estimation.

Results. Figure 2 indicates that low variance critic estimates significantly boost the policy performance. In particular, for complex domains such as Humanoid and Kuka,

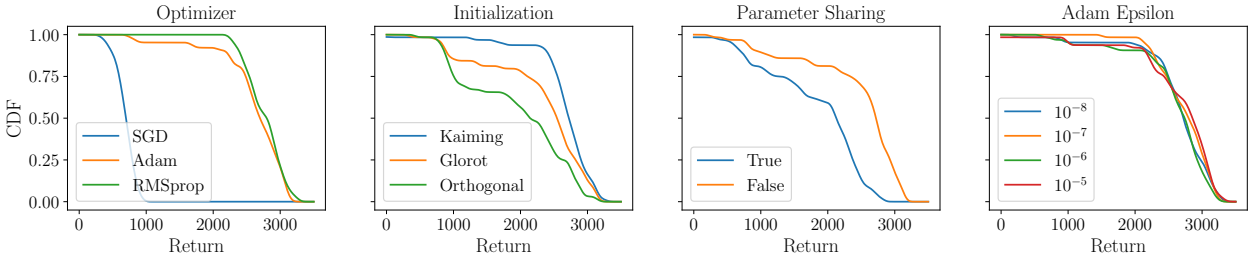


Figure 3: The CDF line plot of the policy performance smoothed over 64 independent seeds for each configuration in the Ant environment.

all tested algorithms could not accomplish high returns with plain advantage estimates. The combination of PPO and GAE showed the best results, scoring in three out of eight tasks the highest performance. Even when plain advantage estimation is used, PPO showed the most robust results across all tasks. Surprisingly and despite its simplicity, the IWPG algorithm performed without trust-region enforcement only slightly worse than its counterparts but necessitated the use of GAE or V-trace. ESC adds an early stopping criterion to IWPG which resulted in a more stable learning and overall small performance gains. This underlines the importance of constraining step sizes in the parameter space. The learning curves of all eight environments can be found in the supplemental material.

5.3 Study of Structural Ingredients

In this part, we analyze the effect of structural ingredients on the learned performance that can vary between different software frameworks. Some might be neglected in the experimental description and can only be found in the implementation. We investigated the following four ingredients:

1. *Optimizer.* Besides Adam, we investigated Stochastic Gradient Descent (SGD) and RMSprop as optimizers for the policy network.
2. *Initialization scheme.* We compared Kaiming Uniform, Glorot, and Orthogonal which determine the initial values of the neural network weights. The bias parameters were initialized as zero vectors.
3. *Parameter Sharing.* Accelerating the learning of representation features, parameters can be shared and updated by both policy and value network.
4. *Adam Epsilon.* A term added to the denominator of the update step to improve the numerical stability of Adam.

Approach. Continuing the stage-wise analysis, we built on the findings from previous experiments and used IWPG with observation standardization and learning rate annealing as well as reward scaling (only for locomotion tasks) as code-level ingredients and applied GAE for variance reduction. We performed a grid search over 4×4 combinations of policy learning rate and number of training iterations and accumulated the scores as cumulative distribution functions (CDF) over four independent seeds, resulting in 64 independent runs for each environment (see Figure 3).

Results. The scores for the optimizer vary from environment to environment. RMSprop showed across all environments the best average CDF score and was the most robust choice over the grid search, whereas Adam yielded the highest performing configuration. The weight initialization scheme showed in Ant, Humanoid, and Kuka large impacts, whereas in other tasks negligible effects. Overall, the Kaiming Uniform scheme yielded the most robust scores. The parameter sharing of two hidden layers between the policy and value network resulted in all tasks to worse performance scores than separated networks, except for the Pusher environment. Surprisingly, the ϵ term added to the denominator of Adam's policy updates showed an influence on the learning, where $\epsilon \in \{10^{-8}, 10^{-5}\}$ yielded the best average performance scores across all tasks. The plots for all eight environments are provided in the supplemental materials.

5.4 Identifying the Minimal Setup

The second objective of our paper is to find a minimal configuration that can challenge state-of-the-art RL algorithms. We inferred from our previous findings and used the IWPG objective with standardized observations, learning rate annealing and GAE as the minimal setup. Reward scaling was applied in the locomotion tasks but not in the manipulation tasks. We tested both RMSprop and Adam as policy optimizers and applied the Kaiming initialization scheme. To validate the reliability of our results, we benchmarked the identified setup against the popular OpenAI Baselines repository [Dhariwal *et al.*, 2017]. For each algorithm, we conducted a grid search over different learning rates, the number of policy iterations and trust-region sizes, while fixing the other hyperparameters such as the batch size of generated trajectories, networks architectures for policy and value network, etc. We depict the best score averaged over four independent seeds for each code base in Figure 4. The results show that our identified setup of ingredients can keep up with the state-of-the-art algorithms TRPO and PPO.

6 Discussion

Our experiments showed that observation standardization is crucial for good policy performance, which was also reported in Andrychowicz *et al.* [2020] and Engstrom *et al.* [2020]. The only task that did not profit was Reacher. We suppose that this is due to the well-designed observation space with small magnitudes and that random exploration already discovers the state space sufficiently well. The second essen-

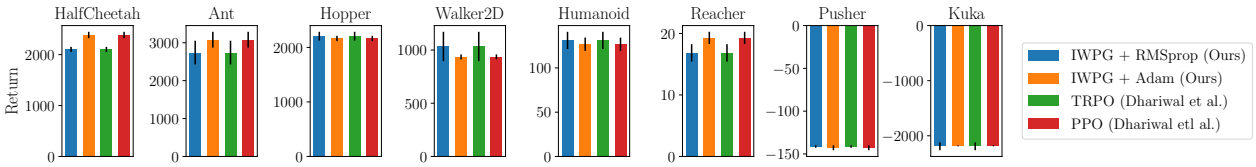


Figure 4: Comparison between our identified minimal setup of ingredients and the two algorithms TRPO and PPO from the OpenAI's Baselines repository. All algorithms use the same default hyper-parameters and the best scores are determined through grid search.

tial ingredient is learning rate annealing. We utilized a linear decay scheme which showed in all tasks performance gains. Our conclusion is similar to the one of Andrychowicz *et al.* [2020], who noticed on average small gains while using GAE. Our extended analysis showed performance increases independent from the applied advantage estimation technique. The decay of the learning rate reduces the step size in the policy parameter space towards the end of learning, eventually promoting the convergence to better local solutions. Reward scaling was found to be beneficial by Engstrom *et al.* [2020] but showed in our experiments only improvements in the locomotion tasks. We claim that reward scaling can be omitted when environments exhibit dense rewards with well-scaled magnitudes. An additional entropy bonus usage showed mixed outcomes, being preferable in environments with more unstable dynamics such as Hopper, Walker, and Pusher over tasks with more stable dynamics such as HalfCheetah and Ant. In contrast, the standardization of advantages yielded performance losses in all except one environment, which was not investigated but used by default in Engstrom *et al.* [2020]. Our extended analysis showed that advantage standardization yields profits when no reward scaling is applied but reduces performance when both are applied. We recommend to exclusively use the one or the other, where reward scaling is preferable. We noticed that gradient scaling hindered learning on average. Since Adam is in-variant to gradient magnitudes [Kingma and Ba, 2015], we assume that the non-linear operation of gradient scaling disturbs Adam's internal updates.

Our investigation of algorithmic ingredients showed that those are necessary to reach state-of-the-art results. In fact, variance reduction techniques for critic estimates deliver significant performance increases. A similar assertion can be made for trust-region enforcement methods, which helped to improve the policy performance and the robustness towards the hyper-parameter selection in all tasks, but had less effect on the learning performance than variance reduction techniques.

Regarding the structural ingredients, we did not find parameter-sharing useful similar to the results of Andrychowicz *et al.* [2020]. However, the choice of the optimizer and the weight initialization scheme had an impact, while other works reported no performance difference [Andrychowicz *et al.*, 2020]. Our experiments showed that RMSprop is more robust over a variety of hyper-parameters than Adam.

Overall, the preceding discussion points to a broader problem of RL algorithms: numerical sensitivity. Many ingredients aim to improve learning stability through standardized inputs and regression targets. Many of the discussed ingredients may be obsolete for algorithms if the environment is already well-specified in terms of input-output range and tailored to the peculiarities of neural networks. This suggests that environment specifics also matter and demands the practitioner's carefulness already during the problem design.

7 Conclusion

In this work, we took a step towards a transparent approach that investigates the inner workings of on-policy policy gradient algorithms. We studied algorithm ingredients in a modular and incremental approach and empirically assessed their contribution to the learning. We found that only a subset of such ingredients is necessary to achieve state-of-the-art results on contemporary benchmarks. Further, we confirmed that recent algorithmic advances such as the variance reduction methods of critic estimates are essential to obtain good policy performance. To this end, we identified a minimum setup of algorithm ingredients that can confront state-of-the-art RL algorithms while promising better reproducibility due to less algorithm complexity.

Our paper shows that simple algorithms can also perform well and may not be limited to the currently benchmark-driven development of new algorithms. For a steady progress, we suggest that new RL proposals should be assessed on a unified implementation with a modular structure. Otherwise, side-effects such as code-level ingredients may be accountable for the claimed performance gains. Further works on this topic may investigate off-policy algorithms and test a broader range of tasks in both continuous and discrete state spaces.

Acknowledgments

We thank Matthias Kissel and appreciate the support of the Federal Ministry of Education and Research who sponsored this project under the funding code 01IS17049 and the Deutsche Forschungsgemeinschaft (DFG) through TUM International Graduate School of Science and Engineering (IGSSE), GSC 81.

References

[Andrychowicz *et al.*, 2020] Marcin Andrychowicz, Anton Raichuk, Piotr Stanczyk, Manu Orsini, Sertan Girgin, Raphael Marinier, L'eonard Hussenot, Matthieu Geist, Olivier Pietquin, Marcin Michalski, Sylvain Gelly, and Olivier Bachem. What matters in on-policy reinforcement learning? a large-scale empirical study. *ArXiv*, abs/2006.05990, 2020.

[Arulkumaran *et al.*, 2017] K. Arulkumaran, M. P. Deisenroth, M. Brundage, and A. A. Bharath. Deep reinforcement learning: A brief survey. *IEEE Signal Processing Magazine*, 34(6):26–38, Nov 2017.

[Coumans and Bai, 2016] Erwin Coumans and Yunfei Bai. Pybullet, a python module for physics simulation for games, robotics and machine learning. http://pybullet.org, 2016. Accessed: 2021-05-23.

[Dhariwal *et al.*, 2017] Prafulla Dhariwal, Christopher Hesse, Oleg Klimov, Alex Nichol, Matthias Plappert, Alec Radford, John Schulman, Szymon Sidor, Yuhuai Wu, and Peter Zhokhov. Openai baselines. https://github.com/openai/baselines, 2017. Accessed: 2021-05-30.

[Duan *et al.*, 2016] Yan Duan, Xi Chen, Rein Houthooft, John Schulman, and Pieter Abbeel. Benchmarking deep reinforcement learning for continuous control. In *Proceedings of the 33rd International Conference on International Conference on Machine Learning*, pages 1329–1338, 2016.

[Engstrom *et al.*, 2020] Logan Engstrom, Andrew Ilyas, Shibani Santurkar, Dimitris Tsipras, Firdaus Janoos, Larry Rudolph, and Aleksander Madry. Implementation matters in deep rl: A case study on ppo and trpo. In *International Conference on Learning Representations*, 2020.

[Espeholt *et al.*, 2018] Lasse Espeholt, Hubert Soyer, Rémi Munos, Karen Simonyan, Volodymyr Mnih, Tom Ward, Yotam Doron, Vlad Firoiu, Tim Harley, Iain Dunning, Shane Legg, and Koray Kavukcuoglu. IMPALA: scalable distributed deep-rl with importance weighted actor-learner architectures. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1406–1415. PMLR, 2018.

[Henderson *et al.*, 2018] Peter Henderson, Riashat Islam, Philip Bachman, Joelle Pineau, Doina Precup, and David Meger. Deep reinforcement learning that matters. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 3207–3214. AAAI Press, 2018.

[Islam *et al.*, 2017] Riashat Islam, Peter Henderson, Maziar Gomrokchi, and Doina Precup. Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. *CoRR*, abs/1708.04133, 2017.

[Kakade, 2002] Sham M Kakade. A natural policy gradient. In *Advances in Neural Information Processing Systems 14*, pages 1531–1538. MIT Press, 2002.

[Kingma and Ba, 2015] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, 2015.

[Mania *et al.*, 2018] Horia Mania, Aurelia Guy, and Benjamin Recht. Simple random search of static linear policies is competitive for reinforcement learning. In *Advances in Neural Information Processing Systems 31*, pages 1800–1809. Curran Associates, Inc., 2018.

[Mnih *et al.*, 2015] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529 EP –, 02 2015.

[Peters and Schaal, 2008] Jan Peters and Stefan Schaal. Reinforcement learning of motor skills with policy gradients. *Neural Networks*, 21(4):682 – 697, 2008. Robotics and Neuroscience.

[Rajeswaran *et al.*, 2017] Aravind Rajeswaran, Kendall Lowrey, Emanuel V. Todorov, and Sham M Kakade. Towards generalization and simplicity in continuous control. In *Advances in Neural Information Processing Systems 30*, pages 6550–6561. Curran Associates, Inc., 2017.

[Reda *et al.*, 2020] Daniele Reda, Tianxin Tao, and Michiel van de Panne. Learning to locomote: Understanding how environment design matters for deep reinforcement learning. In *Motion, Interaction and Games*, MIG '20, New York, NY, USA, 2020. Association for Computing Machinery.

[Rissanen, 1978] J. Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465 – 471, 1978.

[Schulman *et al.*, 2015] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pages 1889–1897. PMLR, 2015.

[Schulman *et al.*, 2016] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. In *Proceedings of the International Conference on Learning Representations*, 2016.

[Schulman *et al.*, 2017] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *CoRR*, abs/1707.06347, 2017.

[Sigaud and Stulp, 2019] Olivier Sigaud and Freek Stulp. Policy search in continuous action domains: An overview. *Neural Networks*, 113:28 – 40, 2019.

[Silver *et al.*, 2016] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Vedavyas Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy P. Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.

[Tucker *et al.*, 2018] George Tucker, Surya Bhupatiraju, Shixiang Gu, Richard Turner, Zoubin Ghahramani, and Sergey Levine. The mirage of action-dependent baselines in reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 5015–5024. PMLR, 2018.

The Successful Ingredients of Policy Gradient Algorithms

Author: Sven Gronauer et al.
Proceedings: 30th International Joint Conference on Artificial Intelligence
Publisher: International Joint Conferences on Artificial Intelligence

Reprint Permission

International Joint Conferences on Artificial Intelligence, Inc. (IJCAI) grants to the above authors, and the employers for whom the work was performed, royalty-free permission to:

1. retain all proprietary rights (such as patent rights) other than copyright and the publication rights transferred to IJCAI;
2. personally reuse all or portions of the paper in other works of their own authorship;
3. make oral presentation of the material in any forum;
4. reproduce, or have reproduced, the above paper for the author's personal use, or for company use provided that IJCAI copyright and the source are indicated, and that the copies are not used in a way that implies IJCAI endorsement of a product or service of an employer, and that the copies per se are not offered for sale. The foregoing right shall not permit the posting of the paper in electronic or digital form on any computer network, except by the author or the author's employer, and then only on the author's or the employer's own World Wide Web page or ftp site. Such Web page or ftp site, in addition to the aforementioned requirements of this Paragraph, must provide an electronic reference or link back to the IJCAI electronic server (`http://www.ijcai.org`), and shall not post other IJCAI copyrighted materials not of the author's or the employer's creation (including tables of contents with links to other papers) without IJCAI's written permission;
5. make limited distribution of all or portions of the above paper prior to publication.
6. In the case of work performed under U.S. Government contract, IJCAI grants the U.S. Government royalty-free permission to reproduce all or portions of the above paper, and to authorize others to do so, for U.S. Government purposes.

In the event the above paper is not accepted and published by IJCAI, or is withdrawn by the author(s) before acceptance by IJCAI, this agreement becomes null and void.

Paper II

Using Simulation Optimization to Improve Zero-Shot Policy Transfer of Quadrotors

Sven Gronauer, Matthias Kissel, Luca Sacchetto, Mathias Korte and Klaus Diepold

Abstract

In this work, we propose a data-driven approach to optimize the parameters of a simulation such that control policies can be directly transferred from simulation to a real-world quadrotor. Our neural network-based policies take only onboard sensor data as input and run entirely on the embedded hardware. In real-world experiments, we compare low-level Pulse-Width Modulated control with higher-level control structures such as Attitude Rate and Attitude, which utilize Proportional-Integral-Derivative controllers to output motor commands. Our experiments show that low-level controllers trained with Reinforcement Learning require a more accurate simulation than higher-level control policies at the expense of being less robust towards parameter uncertainties.

S. Gronauer, M. Kissel, L. Sacchetto, M. Korte, and K. Diepold. "Using Simulation Optimization to Improve Zero-Shot Policy Transfer of Quadrotors". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2022, pp. 10170–10176. DOI: 10.1109/IROS47612.2022.9981229

Using Simulation Optimization to Improve Zero-shot Policy Transfer of Quadrotors

Sven Gronauer, Matthias Kissel, Luca Sacchetto, Mathias Korte and Klaus Diepold

***Abstract*— In this work, we propose a data-driven approach to optimize the parameters of a simulation such that control policies can be directly transferred from simulation to a real-world quadrotor. Our neural network-based policies take only onboard sensor data as input and run entirely on the embedded hardware. In real-world experiments, we compare low-level Pulse-Width Modulated control with higher-level control structures such as Attitude Rate and Attitude, which utilize Proportional-Integral-Derivative controllers to output motor commands. Our experiments show that low-level controllers trained with Reinforcement Learning require a more accurate simulation than higher-level control policies at the expense of being less robust towards parameter uncertainties.**

I. INTRODUCTION

Programming intelligent control strategies for complex robot systems is a challenging task. Reinforcement learning (RL) promises the automated generation of control strategies through a data-driven approach instead of explicitly designing hand-crafted solutions through expert knowledge. In recent years, the field of RL has witnessed outstanding successes and raised a surge of interest in the control of dynamical systems through such a trial-and-error paradigm. The combination of RL and deep learning methods excel at problems that can be quickly simulated like robotics [13] and video games [19] or in domains where the exact model is known but long-horizon planning is not computationally tractable, e.g. board games like Go and Chess [26].

Despite the significant advances in recent years, the applicability of RL algorithms is still limited when the data at test time differ from those seen during training [10]. Since many real-world systems cannot afford to learn policies from scratch due to the expense of data, simulations are the preferred approach to build data-driven control policies in the RL community. Naturally, a gap between the simulation and the real world exists because an accurate model either requires in-depth expert knowledge or is simply not desirable, e.g. calculating all aero-dynamical effects can significantly increase simulation time. A successful policy transfer thus requires the reality gap to be small.

In this paper, we address the sim-to-real gap in the domain of quadrotor control and investigate three different structures for quadrotor control. Our contributions are:

1) We apply simulation optimization as a data-driven approach to narrow the reality gap and demonstrate that zero-shot policy transfers become feasible with minimal tuning effort. The data used for optimization were collected for approximately one hour on the real-world quadrotor with Proportional-Integral-Derivative (PID) controllers.
2) We deploy low-level control policies based on Pulse-Width Modulated (PWM) thrust commands trained entirely in simulation on the quadrotor without fine-tuning the policy. To the best of our knowledge, this is the first work using RL as framework for low-level decision making that accomplishes successful zero-shot transfers to a real-world quadrotor using only onboard sensing and computation.
3) We study three control structures that differ in their level of abstraction: PWM, Attitude Rate and Attitude control. Through real-world experiments, we investigate the required fidelity of the simulation with respect to the deployed control structure.

Throughout this work, we focus on the context of zero-shot policy transfer, i.e. we train the agent entirely in simulation and then deploy the controller on a quadrotor robot without using real-world data to fine-tune the policy.

II. RELATED WORK

A. Quadrotor Control

For attitude control and set-point tracking of quadrotors, a common approach is a hierarchical control that consists of nested PID controllers [17]. Approaches like the linear quadratic regulator are also suitable methods for the stabilization around the hover conditions under reasonably small roll and pitch angles. However, when more dynamic flight behaviors are desired, more complex controllers are required [18]. Due to their highly dynamic movement capabilities, quadrotors depict an interesting platform to test maneuvers such as landing [11] and perform acrobatic maneuvers like loopings and rolls [9], multi-flips [16] or flying through narrow vertical gaps [18]. While most of the aforementioned works rely on highly accurate state estimation, only a few papers have considered quadrotor control based on pure onboard sensing. In [12], model-based RL was used to train a policy from real-world data while relying on onboard sensor measurements to run control with direct motor PWM signals. The authors in [15] showed that a quadrotor can be navigated through a variety of cluttered real-world scenarios like forests or buildings with onboard sensing and computation.

B. Bridging the Reality Gap

To reduce the gap between simulation and real-world, *domain randomization* has been proposed to augment the diversity of data by randomization. One way is to randomize

The authors are with Department of Electrical and Computer Engineering, Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany. Contact author: sven.gronauer@tum.de

the dynamics where simulation parameters responsible for the realization of the system are re-sampled at the beginning of every trajectory [22]. Another approach is to randomize the rendering of image-based observations in the simulation [27]. In [14] it was shown that agile drone racing is possible with convolutional neural networks when trained on an abundance of image-based data. Besides domain randomization, another method to overcome the reality gap is *simulation optimization* [1] which is a data-driven approach to identify simulation parameters based on real-world data. In [2] a simulation parameter distribution was learned based on collected data from a physical manipulator robot.

C. Zero-shot Policy Transfer

In the area of RL, there have been only a few works that study the zero-shot policy transfer to real-world quadrotors. The work of [9] showed that Attitude Rate control based on image-based and onboard data could perform highly agile maneuvers such as rolls and loopings. Another work [15] performed zero-shot transfers into unknown and cluttered real-world environments while using onboard sensing and computation only. The policy computed collision-free trajectories based on cameras which are then tracked by a model-predictive controller. Most similar to our paper is [20] where low-level PWM-based controllers were able to stabilize and generalize to multiple sizes of quadrotors. The authors used an external tracking system to accurately estimate the drone state and showed that a zero-shot transfer is possbile. In contrast to the aforementioned works, we study the zero-shot policy transfer with onboard sensing and computation while using RL as the only framework for low-level decision making.

III. PRELIMINARIES

A. Reinforcement Learning

A Markov Decision Process (MDP) is formalized by a tuple $(\mathbb{S}, \mathbb{A}, \mathcal{P}, r, \mu)$, where $\mathbb{S}$ and $\mathbb{A}$ denote the state and action spaces respectively. $\mathcal{P} : \mathbb{S} \times \mathbb{A} \to P(\mathbb{S})$ describes the system transition probability, μ denotes the initial state distribution and $r : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ is the reward function. Let $\tau = (\boldsymbol{s}_0, \boldsymbol{a}_0, \boldsymbol{s}_1, \boldsymbol{a}_1, \dots)$ be a trajectory generated under policy π with $\boldsymbol{s}_{t+1} \sim \mathcal{P}(\cdot|\boldsymbol{s}_t, \boldsymbol{a}_t), \boldsymbol{a}_t \sim \pi(\cdot|\boldsymbol{s}_t)$ and $\boldsymbol{s}_0 \sim \mu$. We apply the shortcut $\tau \sim \pi$ when trajectories are generated under π and denote the trajectory return by $R(\tau) = \sum_{t=0}^{\infty} \gamma^t r(\boldsymbol{s}_t, \boldsymbol{a}_t)$ with the discount factor $\gamma \in (0, 1)$. In RL, the goal of the agent is to learn a control policy $\pi : \mathbb{S} \to \mathcal{P}(\mathbb{A})$ that maximizes the expected return $J(\pi) = \mathbb{E}_{\tau\sim\pi}[R(\tau)]$. We use π_θ to denote that the policy is parametrized by a neural network with weights $\boldsymbol{\theta}$.

In this work, we sample system parameters $\boldsymbol{\xi}$ from a distribution Ξ at the beginning of each trajectory. This results in the system transition probability being dependent on the system parameters $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t, \xi)$ which is known as domain randomization. Thus, we seek policy parameters that maximize the expected return

$$\max_{\boldsymbol{\theta}} J(\pi_\theta) = \mathbb{E}_{\boldsymbol{\xi}\sim\Xi}\left[\mathbb{E}_{\tau\sim\pi_\theta}[R(\tau)]\right] \tag{1}$$

over the dynamics induced by the distribution of simulation parameters.

B. Quadrotor

1) Dynamics Model: The dynamics of the quadrotor are modeled by the differential equation

$$\dot{\boldsymbol{x}} = f(\boldsymbol{x}, \boldsymbol{u}) \tag{2}$$

where the drone state $\boldsymbol{x} = [\boldsymbol{p}^T, \dot{\boldsymbol{p}}^T, \boldsymbol{\varphi}^T, \boldsymbol{\omega}^T]^T \in \mathbb{R}^{13}$ encompasses the position $\boldsymbol{p}$, the linear velocity $\dot{\boldsymbol{p}}$, the body angle $\boldsymbol{\varphi}$ in quaternions and the angular speed $\boldsymbol{\omega}$. The acceleration of the drone's center of mass is described by Newton's equation

$$m\ddot{\boldsymbol{p}} = \begin{bmatrix} 0 \\ 0 \\ -mg \end{bmatrix} + R \begin{bmatrix} 0 \\ 0 \\ \sum F_i \end{bmatrix} \tag{3}$$

in the inertial frame $\mathcal{O}$ with gravity g. The quadrotor mass is m and $\sum F_i$ is the sum of the vertical forces acting on the rotors. R is the rotation matrix from the body frame $\mathcal{B}$ to the inertial frame. The angular acceleration governed by Euler's rotation equations in $\mathcal{B}$ is

$$I\dot{\boldsymbol{\omega}} = \boldsymbol{\eta} - \boldsymbol{\omega} \times (I\boldsymbol{\omega}) \tag{4}$$

with the inertia matrix I. The torques $\boldsymbol{\eta}$ acting in the body frame are determined by

$$\boldsymbol{\eta} = \begin{bmatrix} \frac{1}{\sqrt{2}} L(-F_1 - F_2 + F_3 + F_4) \\ \frac{1}{\sqrt{2}} L(-F_1 + F_2 + F_3 - F_4) \\ -M_1 + M_2 - M_3 + M_4 \end{bmatrix} \tag{5}$$

with the rotor forces F_i, the corresponding motor torques M_i and the drone's arm length L. An overview of the quadrotor setup and the coordinate frames is depicted in Fig. 1 (Left).

2) Motor Model: The angular speed ν_i of rotor i produces a vertical force

$$F_i = \frac{mg}{4} k_F \nu_i^2 \tag{6}$$

that lifts the quadrotor. We use $k_F \in \mathbb{R}$ to denote the thrust-to-weight ratio. The rotors also produce a moment according to

$$M_i = k_{M_1} F_i + k_{M_2} \tag{7}$$

with k_{M_1} and k_{M_2} being scalars. The motor speeds are normalized $\nu_i \in [0, 1]$ and are modeled $\boldsymbol{\nu} = \sqrt{\boldsymbol{u}}$ as the square root of normalized commanded thrusts $u_i \in [0, 1]$. For PWM control, the relationship between the action $\boldsymbol{a}$ taken by the agent and the normalized commanded thrust is $u_i = \frac{1}{2}\left[\min(\max(a_i, -1), 1) + 1\right]$. We model the motor dynamics with a differential equation of first order

$$T_m \dot{\nu}_i = -\nu_i + \sqrt{u_i}, \tag{8}$$

where $T_m \in \mathbb{R}$ is the motor time constant. A common approach in related work is to neglect the motor dynamics and assume an instantaneous thrust acting on the rotors. However, as we observed in our experiments, an accurate actuator model is crucial for a successful sim-to-real transfer. Additionally, we model the overall latency Δ of the system

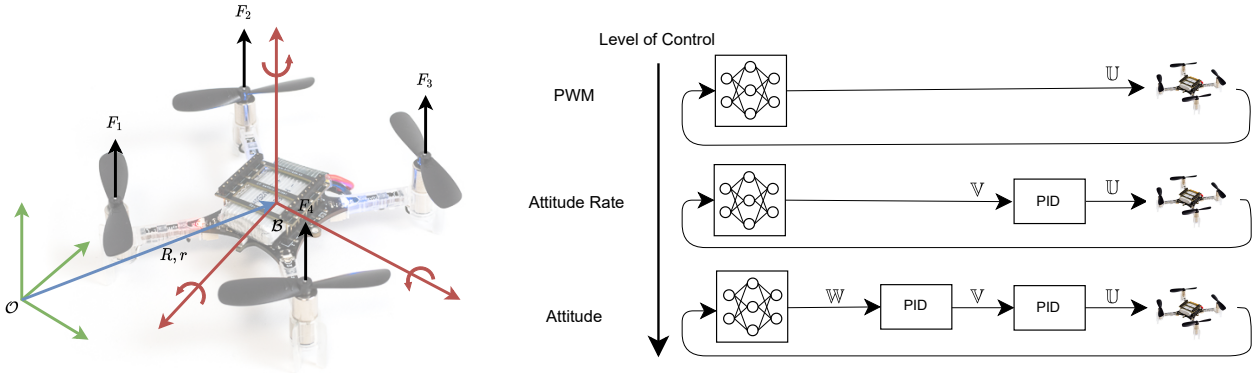


Fig. 1: The CrazyFlie Quadrotor and the investigated control structures. (Left) Overview and coordinates frames of the *CrazyFlie* quadrotor. (Right) Applied control structures from low-level PWM to high-level Attitude.

which should capture all delays emerging in the hardware. Typically, latency arises in the state estimator, in the actuation of the motors and by propagating data through the neural network policy.

IV. METHODS

We aim to find a policy that reliably transfers from the simulation to the quadrotor robot without using real-world data to fine-tune the policy. To render a zero-shot transfer feasible, we optimize the simulation parameters based on collected real-world measurements.

Besides simulation optimization, we want to find an answer to our hypothesis that *low-level control structures demand a higher simulation fidelity*. The intuition behind this assumption is that low-level control requires the policy to capture the mapping from drone state to motor commands and thus to understand to underlying quadrotor dynamics. High-level control, on the other hand, encourages the agent to learn an abstract understanding of the task since only the mapping from the drone state to an intermediate space is required. The mapping from the policy output space to the motor commands is subsequently handled by PID controllers.

The remainder of this section describes the details of our simulation environment including the observation and action space, the procedure of the simulation optimization using Bayesian optimization and an introduction to the three control structures that were deployed and evaluated on the real-world quadrotor.

A. Simulation Environment

We use the physics simulator PyBullet [3] to calculate the dynamics of the quadrotor akin to the work of [21]. We run the simulator and the motor dynamics with 200Hz while the agent receives noisy observations with $\leq$ 100Hz depending on the selected control structure. The implementation is publicly available at: `https://github.com/SvenGronauer/phoenix-drone-simulation`.

a) Observations: In every time step t, the agent receives the observation $\boldsymbol{s}_t = [\boldsymbol{x}_t^T, \boldsymbol{e}_t^T, \boldsymbol{a}_{t-1}^T]^T \in \mathbb{R}^{20}$ containing the drone state vector $\boldsymbol{x} \in \mathbb{R}^{13}$, the difference vector $\boldsymbol{e} = \boldsymbol{p} - \boldsymbol{p}_{ref} \in \mathbb{R}^3$ between the drone's current position and the set-point position, and the action $\boldsymbol{a}_{t-1}^T \in \mathbb{R}^4$ taken previously. Note that the agent has not access to all information of the drone state because the rotor speeds cannot be directly measured. To cope with partial observability, the agent is equipped with a history of observations with size $H \geq 1$, i.e. $(\boldsymbol{s}_{t-H+1}, \dots, \boldsymbol{s}_t) \in \mathbb{R}^{20H}$. We conducted a study about the optimal history size $H \in [1, 2, 4, 6, 8]$ in simulation and found that $H = 2$ performed best.

b) Actions: Actions represent abstract, possibly high-level commands which are transformed by the control structure to the corresponding motor command $\boldsymbol{u} \in \mathbb{U}$. The implementation differs for each control structure and is explained in Section IV-C.

c) Domain Randomization: Throughout training, we randomize the following system parameters uniformly in the range $\pm 10\%$ of the default value: Thrust-to-weight ratio k_F, physics time-step, quadrotor mass m, diagonal of inertia matrix I, motor time constant T_m and yaw-torque factors k_{m_1} and k_{m_2}.

d) Noise: As sensor noise, we add Gaussian and uniform noise for positions $\boldsymbol{p}$, velocities $\dot{\boldsymbol{p}}$ and angles $\boldsymbol{\varphi}$. For the angle rates, we apply the sensor model proposed in [5] which adds a Gaussian noise and a time-varying bias to $\boldsymbol{\omega}$. In a similar vain to [20], we use a discretized Ornstein-Uhlenbeck process to model actuator noise which we add to $\boldsymbol{u}$.

B. Simulation Optimization

Simulation optimization is an approach to tune the parameters of a simulation through data [1]. The aim is to bring the simulation as close as possible to the real-world and ideally close the reality gap. The objective function which we want to minimize is

$$\min_{\boldsymbol{\xi} \in \Xi} F(\boldsymbol{\xi}, \mathcal{D}) = \sum_t^T \delta^t \| W \left(\boldsymbol{x}_t^{sim}(\boldsymbol{\xi}) - \boldsymbol{x}_t^{real} \right) \|_1 + \sum_t^T \delta^t \| W \left(\boldsymbol{x}_t^{sim}(\boldsymbol{\xi}) - \boldsymbol{x}_t^{real} \right) \|_2 \quad (9)$$

where T is the mini-trajectory length, $\mathcal{D}$ is the data set and $\delta \in (0, 1)$ is a factor to discount later stages of the trajectory where the divergence increases due to discrepancies between simulation and real-world. We weight the drone state difference $\boldsymbol{x}_t^{sim}(\boldsymbol{\xi}) - \boldsymbol{x}_t^{real}$ according to the matrix W with the purpose to scale angle errors and angle rate errors differently. Similar to [2], we use the sum of L1 and L2 norm.

Prior sim-to-real work reported that the accurate modeling of the actuators including dynamics, noise and delays is crucial for a successful transfer. In [22] it was pointed out that randomizing the latency of the controller and injecting sensor noise to the simulation are key components for a high success rate. Similarly, it was demonstrated in [8] that learned actuator dynamics significantly helped to reduce the reality gap. Guided by these results, we decided to optimize those simulation parameters $\boldsymbol{\xi} = [k_F, T_m, \Delta]^T$ that describe the motor behavior as well as the system latency.

Bayesian Optimization (BO) is our method of choice, as it is regarded as a good option for the optimization in continuous domains with less than 20 dimensions and is robust against stochastic function evaluations [4]. BO is particularly suited for objective functions that have long evaluation times and can cope with low sample complexity. In contrast to gradient-based methods, BO is not prone to converge to local optima.

C. Control Structures

In this work, we investigate the following three control structures, ordered from low to high-level:

1) *PWM Control:* The agent is supposed to learn a mapping $\pi : \mathbb{S} \rightarrow \mathbb{U}$ from observation space to thrust commands. The agent receives noisy observations from the Kalman state estimator with 100Hz.
2) *Attitude Rate Control:* The policy outputs lie in the space $[c, \boldsymbol{\omega}_d^T]^T \in \mathbb{V} \subseteq \mathbb{R}^4$ which consists of the mass-normalized collective thrust c and the desired body angle rate $\boldsymbol{\omega}_d$. The PID controller calculates the thrust commands $\boldsymbol{u} \in \mathbb{U}$ based on c and the error between the actual $\boldsymbol{\omega}$ and the desired $\boldsymbol{\omega}_d$. The agent receives noisy observations with 50Hz and is supposed to learn the mapping $\pi : \mathbb{S} \rightarrow \mathbb{V}$.
3) *Attitude Control:* The agent is incentivized to learn the mapping $\pi : \mathbb{S} \rightarrow \mathbb{W}$ where the policy output space $[c, \boldsymbol{\varphi}_d^T] \in \mathbb{W} \subseteq \mathbb{R}^4$ encompasses the mass-normalized collective thrust c and the desired body angle $\boldsymbol{\varphi}_d$. Subsequently, the mapping $\mathbb{W} \rightarrow \mathbb{V} \rightarrow \mathbb{U}$ is handled by two cascaded PID controllers. Noisy observations are fed into the policy with 25Hz.

These three control structures are illustrated in Fig. 1. We test our hypothesis about the fidelity of the simulator based on the deployed control method in the next section.

V. RESULTS

In this section, we describe our experiments and discuss the results. First, we elaborate on the setup of the hardware and the learning task. Second, we explain our simulation optimization experiments with BO and present our found simulation parameters. Third and last, we describe our zero-shot transfer experiments that were conducted on the real-world quadrotor. After training control policies in simulation, we studied the three control structures PWM, Attitude Rate and Attitude in order to test our control level hypothesis of Section IV.

A. Experimental Setup

As quadrotor robot, we used the *CrazyFlie 2.1*. Due to its small size with a motor-to-motor diameter of 13cm and light weight of 30g, the *CrazyFlie* is very agile and demands high control frequencies. Furthermore, due to its low-cost design, the *CrazyFlie* exhibits high parameter uncertainties in its building parts which additionally requires the controller to be robust over a large system parameter range. The drone is equipped with a 3-axis gyroscope, a 3-axis accelerometer and a z-axis LIDAR. The sensor data were fed into an extended Kalman filter that runs with 100Hz for drone state estimation. For our experiments, we only used onboard sensor measurements and did not use an external tracking system. For evaluation purposes, we transmitted the flight information via *CrazyRadio* to a host computer where we analyzed the logged data.

As learning task, we want the quadrotor to fly a circle figure with a diameter of 0.5m and with a period of 3s. Each trajectory starts on a random point of the reference circle at the height of 1m going in clock-wise direction. The trajectory length in simulation is 500 time steps. To incentivize the drone following the set-point trajectory $\boldsymbol{p}_{ref}$, the reward function is designed as

$$\begin{aligned} r(\boldsymbol{s}, \boldsymbol{a}, t) = &- \|\boldsymbol{e}\|_2 - 0.0001\|0.5(\boldsymbol{a}_t + 1)\|_2 \\ &- 0.001\|\boldsymbol{a}_{t-1} - \boldsymbol{a}_t\|_2 - 0.001\|\boldsymbol{\omega}\|_2 + r_f \end{aligned} \quad (10)$$

with $r_f = -100$ being the terminal reward if $\|\boldsymbol{e}\|_2 > 0.25$ else $r_f = 0$. The actions are clipped into $[-1, 1]$ to regard actuator limits.

B. Simulation Optimization

We used the pre-implemented cascaded PID position controller from the *CrazyFlie* platform to collect data tuples $(\boldsymbol{x}, \boldsymbol{u})_t$ while the drone was flying the circle figure as described in Section V-A. Data were logged with 100Hz and we collected approximately one hour of real-world data. We think that less data would be also sufficient but we did not consider the amount of samples in this work. For building the data set $\mathcal{D}$, we used every tenth data-point from the collected real-world data as starting state for a mini-trajectory of length T. For instance, for $T = 50$ we obtained a data set of size $\mathcal{D} \in \mathbb{R}^{37673 \times 50 \times 13}$.

With BO, we opted to find the global optimum over the bounded set of system parameters $k_F \in [1.5, 2.5]$, $T_m \in [0.01, 0.50]$ and $\Delta \in [0.00, 0.05]$ with respect to $\mathcal{D}$. We ran the experiment for 250 evaluations on the objective from (9) and averaged the results over three trials. We tested different mini-trajectory lengths $T \in \{10, 20, 30, 40, 50\}$. As hyper-parameters, we used $\delta = 0.95$ as the discount factor and as function approximator we used a Gaussian process

TABLE I: Simulation optimization results found by BO. The mean and standard deviation was calculated over three trials.

Parameter	$T = 10$	$T = 20$	$T = 30$	$T = 40$	$T = 50$
Thrust-weight k_F	1.746 ± 0.0050	1.733 ± 0.0093	1.725 ± 0.0018	1.725 ± 0.0081	1.722 ± 0.0027
Time Constant T_m	0.102 ± 0.0110	0.087 ± 0.0087	0.088 ± 0.0006	0.096 ± 0.0035	0.104 ± 0.0012
Latency Δ	0.006 ± 0.0039	0.014 ± 0.0038	0.018 ± 0.0003	0.018 ± 0.0007	0.018 ± 0.0004

with the acquisition function uniformly chosen from lower confidence bound, expected improvement and probability of improvement.

The results obtained by the simulation optimization are shown in Table I. Contrary to our expectation, the mini-trajectory length T had only a minor impact on the obtained results. Except for the results for $T = 10$, the found parameters $\boldsymbol{\xi}$ lie in a similar range.

C. Zero-Shot Experiments

We conducted zero-shot experiments on the real quadrotor where we first trained policy networks with different simulation parameters and thereafter measured the transfer performance. We decided on such experimental setup for three reasons. First, we wanted to evaluate the performance of the parameters found by simulation optimization for an induced reality gap. Second, we intended to study the robustness of the policies trained in different scenarios. Third and last, we wanted to test our control level hypothesis as stated in Section IV.

We aligned the hyper-parameters to the ones suggested in [6] and [7]. We applied a distributed learner setup where the policy gradients were computed and averaged across 64 workers with a total batch size of 64000. We trained with the Proximal Policy Optimization [24] algorithm over 500 iterations and applied as discount factor $\gamma = 0.99$. For the neural network architecture, we used multi-layer perceptrons with two hidden layers. The critic network used 64 neurons each followed by tanh non-linearities whereas the actor had 50 neurons in each hidden layer with ReLU as activations. Since the policy network is supposed to run on the drone micro-controller, we reduced the number of hidden neurons to achieve an inference time of $<$ 1ms for one forward pass. The weights were set according to Kaiming Uniform and biases were initialized as zero vectors. We did not apply parameter sharing between the actor and the critic. Both networks were optimized with Adam where the learning rate of the value network was 0.001 and 0.0003 for the policy. To reduce the variance of critic estimates, we applied Generalized Advantage Estimation (GAE) [23] with the weighting factor $\lambda = 0.95$. Over the training, we used stochastic policies where the policy output is the mean of a multi-variate Gaussian distribution $\boldsymbol{a} \sim \mathcal{N}(\pi(\boldsymbol{s}), \epsilon I)$ with I being the identity matrix and $\epsilon \in \mathbb{R}$ the exploration noise that was linearly annealed from 0.5 to 0.01. Fig. 2 shows in the top row the learning curves of the RL training. Note that the return generated by PWM is not comparable to the returns of Attitude and Attitude Rate since the reward function prefers negative actions over zero actions. Attitude and Attitude Rate control typically produce zero outputs for angle stabilization. Returns larger than the terminal reward r_f indicate that the quadrotor does not crash and tracks the set point. For the three control structures, all trained policies were capable of tracking the set-point in simulation.

After training three policies for each combination of motor time constant $T_m \in \{0.04, 0.08, 0.12\}$s and latency $\Delta \in \{0, 0.015, 0.02\}$s in simulation, we tested the policies on the real quadrotor robot. We applied this evaluation procedure for each of the three control structures and determined the flight time of policies over three real-world trials. Overall, this resulted in $3^4 = 81$ trained neural networks and thus 243 flights. If a policy was able to fly longer than 20s, we manually stopped the execution of the neural network. If the policy was not able to stabilize the quadrotor, the trial was terminated by a stabilizing backup PID controller, which intervened when the roll or pitch angle was greater than $30°$ or when roll and pitch rates exceeded $800°$/s. We observed that the rollouts also failed when the distance between the drone and the set-point exceeded the episode termination criterion of $\|\boldsymbol{e}\|_2 > 0.25$m. The results are displayed in the bottom row of Fig. 2.

The PWM control structure showed good zero-shot performance when trained on the motor time constant $T_m = 120$ms with latency $\Delta \geq 15$ms. The parameter setting $T_m = 80$ms, $\Delta = 20$ms closest to the parameters suggested by the simulation optimization resulted in a median flight time of 20s and two outlier flights. However, outside this parameter setting, PWM fastly dropped in performance and often failed. When the latency was set to $\Delta = 0$ms, PWM yielded the worst performance compared to the other two control approaches.

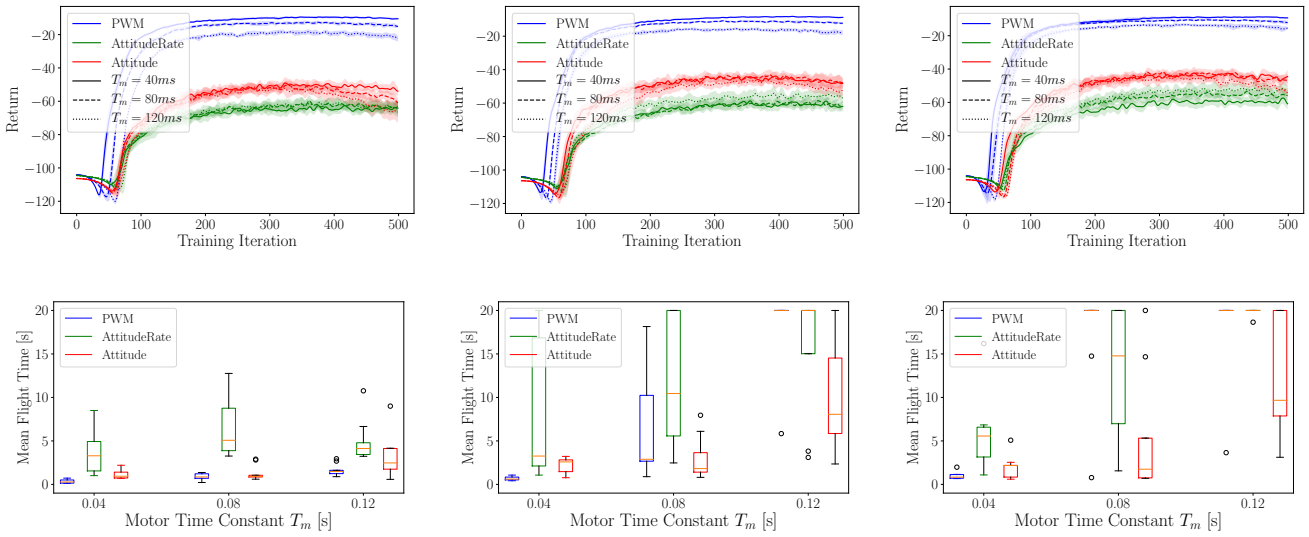
Overall, the Attitude Rate control structure showed the most robust results over the tested system parameters. The best performance was observed for the setting $T_m = 120$ms and $\Delta = 20$ms. In contrast to the other control structures, Attitude Rate was also able to stabilize the quadrotor when trained on latency $\Delta = 0$ms, although most flights terminated early due to the $\|\boldsymbol{e}\|_2 > 0.25$m termination criterion.

Surprisingly, Attitude control showed the worst performance despite being the highest control level structure. Only the policies that were trained on settings with $T_m = 120$ms and latency $\Delta \geq 15$ms could achieve a flight duration over 10s.

VI. DISCUSSION

In this section, we discuss our results from the experiments and draw connections to related work. Further, we point out important observations that we made during our experiments.

a) Simulation Optimization: The system parameters closest to the ones found by the simulation optimization showed a reasonable transfer success for the PWM and



(a) Latency $\Delta = 0$ms (b) Latency $\Delta = 15$ms (c) Latency $\Delta = 20$ms

Fig. 2: Training curves in simulation and evaluation of real-world experiments. (Top) The learning curves of the training in simulation are shown for the three different control structures and different values for (T_m, Δ). Each curve is the avarage over three independent runs while the shaded area denotes the standard deviation. The difference in return is due to the action penalties that prefer negative actions over zero actions. All trained policies were able to track the set-point trajectory in simulation. (Bottom) The flight times in the zero-shot experiments were evaluated for different system parameters and control structures. Each box represents nine real-world flights with a maximum flight time of 20s after which we manually stopped the policy execution.

Attitude Rate control. This underlines the potential of data-driven parameter estimation compared to classical methods known from system identification. A benefit is the ease of use since a handful of system parameters can be estimated simultaneously and no isolated measurements of individual physical properties must be conducted.

b) Control-level Hypothesis: For low-level PWM control, we found that good zero-shot performance can only be achieved when the dynamics of the actuator and the latency are accurately estimated. However, if the reality gap is too large, low-level policy control is prone to fail. The higher-level Attitude Rate control, in contrast, shows high robustness towards the choice of system parameters. This confirms our control level hypothesis that low-level control requires higher simulation fidelity than higher-level control structures. Merely Attitude control showed disappointing results in the zero-shot experiments, which we think is due to the small control frequency of 25Hz.

c) Actuator Model: Related work from quadrupedal robots [8] and robotic manipulators [22] suggested that accurate modeling of the actuators is a crucial component for reliable zero-shot policy transfers. Our experiments confirmed that this statement is also valid for quadrotor robots. In addition to that, we observed that adding latency to the simulation acts as a kind of regularization which helped to improve the zero-shot results.

d) Robustness: Due to crashes, we frequently changed spare hardware parts like propellers and motors. Although facing a variety of drone parameters, the zero-shot transfers worked reliably. Moreover, we tested some of the trained policies also on other *CrazyFlie* drones which showed similar flight behavior. This indicates that the trained policies are robust over a large parameter distribution and different drone systems.

e) Bang-bang behavior: Bang-bang behavior is a known issue in RL where policies prefer an action selection towards the boundaries of the action space [25]. Without adding penalties to the actions, we observed that agents produced large changes in the control outputs which caused a severe performance drop in the transfers. By adding penalties for actions $\|\boldsymbol{a}_t\|_2$ and action rates $\|\boldsymbol{a}_{t-1} - \boldsymbol{a}_t\|_2$, we were able to conduct reliable transfers to the real robot. Similar conclusions were made in sim-to-sim experiments where penalties for action and action rate improved both the transfer and the robustness towards disturbances [25]. Further, we noticed that the transfers failed when the action range was selected too high, e.g. an Attitude rate control with the range $[-360, 360]°$/s did not work whereas the range $[-60, 60]°$/s showed the desired results.

VII. CONCLUSION

In this paper, we demonstrated that low-level control policies trained with Reinforcement Learning entirely in simulation can be successfully transferred to a quadrotor robot when the reality gap is small. To render such zero-shot policy transfers feasible, we narrowed the sim-to-real gap by collecting real-world data with a pre-implemented PID controller and applyed simulation optimization. Our neural

network-based policies used onboard sensing and computation only. Finally, we conducted real-world experiments and compared three different control structures ranging from low-level PWM motor commands to high-level Attitude control based on cascaded PID controllers. The experiments confirm our hypothesis that low-level control policies require a higher simulation fidelity, which suggests the use of higher-level action structures like Attitude Rate control when the robot system parameters cannot be accurately estimated or are partially unknown.

APPENDIX

TABLE II: Used drone parameters.

Parameter	Value	Physical Unit
Gravitational acceleration g	9.81	$\mathrm{m{\cdot}s^{-2}}$
Inertial I_{xx}	$1.33 \cdot 10^{-5}$	$\mathrm{kg{\cdot}m^2}$
Inertial I_{yy}	$1.33 \cdot 10^{-5}$	$\mathrm{kg{\cdot}m^2}$
Inertial I_{zz}	$2.64 \cdot 10^{-5}$	$\mathrm{kg{\cdot}m^2}$
Length L	0.0396	m
Mass m	0.030	kg
Torque-to-weight ratio k_{M_1}	$5.96 \cdot 10^{-3}$	m
Torque-to-weight ratio k_{M_2}	$1.56 \cdot 10^{-5}$	N·m

ACKNOWLEDGMENT

We thank Matthias Emde for his support with the implementation of the drone firmware. The project was supported by the German Federal Ministry of Education and Research under grant number 01IS17049. The authors are responsible for the content of this publication.

REFERENCES

[1] Y. Carson and A. Maria, "Simulation optimization: Methods and applications," in *Proceedings of the 29th Conference on Winter Simulation*, ser. WSC '97. USA: IEEE Computer Society, 1997, pp. 118–126.

[2] Y. Chebotar, A. Handa, V. Makoviychuk, M. Macklin, J. Issac, N. Ratliff, and D. Fox, "Closing the sim-to-real loop: Adapting simulation randomization with real world experience," in *2019 International Conference on Robotics and Automation (ICRA)*, 2019, pp. 8973–8979.

[3] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," http://pybullet.org, 2016–2021, accessed: 2021-12-19.

[4] P. I. Frazier, "A tutorial on bayesian optimization," *CoRR*, vol. abs/1807.02811, 2018.

[5] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *RotorS—A Modular Gazebo MAV Simulator Framework*. Springer International Publishing, 2016, pp. 595–625.

[6] S. Gronauer, M. Gottwald, and K. Diepold, "The successful ingredients of policy gradient algorithms," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, 8 2021, pp. 2455–2461.

[7] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 3207–3214.

[8] J. Hwangbo, J. Lee, A. Dosovitskiy, D. Bellicoso, V. Tsounis, V. Koltun, and M. Hutter, "Learning agile and dynamic motor skills for legged robots," *Science Robotics*, vol. 4, 2019.

[9] E. Kaufmann, A. Loquercio, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Deep drone acrobatics," in *Proceedings of Robotics: Science and Systems*, Corvalis, Oregon, USA, July 2020.

[10] R. Kirk, A. Zhang, E. Grefenstette, and T. Rocktäschel, "A survey of generalisation in deep reinforcement learning," *CoRR*, vol. abs/2111.09794, 2021.

[11] J. E. Kooi and R. Babuska, "Inclined quadrotor landing using deep reinforcement learning," in *IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2021, Prague, Czech Republic, September 27 - Oct. 1, 2021*. IEEE, 2021, pp. 2361–2368.

[12] N. O. Lambert, D. S. Drew, J. Yaconelli, S. Levine, R. Calandra, and K. S. J. Pister, "Low-level control of a quadrotor with deep model-based reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 4, no. 4, pp. 4224–4230, 2019.

[13] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *4th International Conference on Learning Representations*, 2016.

[14] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, 2019.

[15] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, 2021.

[16] S. Lupashin, A. Schöllig, M. Sherback, and R. D'Andrea, "A simple learning strategy for high-speed quadrocopter multi-flips," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA) 2010*, Zurich, 2009.

[17] R. Mahony, V. Kumar, and P. Corke, "Multirotor aerial vehicles: Modeling, estimation, and control of quadrotor," *IEEE Robotics Automation Magazine*, vol. 19, no. 3, pp. 20–32, 2012.

[18] D. Mellinger and V. Kumar, "Minimum snap trajectory generation and control for quadrotors," in *2011 IEEE International Conference on Robotics and Automation*, 2011, pp. 2520–2525.

[19] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529 EP –, 02 2015.

[20] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," *CoRR*, vol. abs/1903.04628, 2019.

[21] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok, and A. P. Schoellig, "Learning to fly—a gym environment with pybullet physics for reinforcement learning of multi-agent quadcopter control," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021.

[22] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *2018 IEEE International Conference on Robotics and Automation (ICRA)*, 2018, pp. 3803–3810.

[23] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the International Conference on Learning Representations*, 2016.

[24] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *CoRR*, vol. abs/1707.06347, 2017.

[25] T. Seyde, I. Gilitschenski, W. Schwarting, B. Stellato, M. Riedmiller, M. Wulfmeier, and D. Rus, "Is bang-bang control all you need? solving continuous control with bernoulli policies," in *Thirty-Fifth Conference on Neural Information Processing Systems*, 2021.

[26] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2021/12/10 2018.

[27] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," in *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2017, pp. 23–30.

Using Simulation Optimization to Improve Zero-shot Policy Transfer of Quadrotors

Conference Proceedings: 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Author: Sven Gronauer
Publisher: IEEE
Date: October 23, 2022

Reprint Permission

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:
Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1. In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.

2. In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.

3. If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1. The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]

2. Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.

3. In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to `http://www.ieee.org/publications_standards/publications/rights/rights_link.html` to learn how to obtain a License from RightsLink.

Paper III

Comparing Quadrotor Control Policies for Zero-Shot Reinforcement Learning under Uncertainty and Partial Observability

Sven Gronauer, Daniel Stümke and Klaus Diepold

Abstract

To alleviate the sample complexity of reinforcement learning algorithms, simulations are a common approach to train control policies before deploying the policy on a real-world robot. However, a gap between simulation and reality generally persists, which endorses the aim to train robust policies already in simulation such that those can be transferred to a real robot at a high success rate. In this paper, we investigate history-dependent policies for drone control in the context of zero-shot transfer learning, where the training is conducted exclusively in simulation. We compare policies represented by feed-forward neural networks with recurrent neural networks and assess both performance and robustness on a real-world quadrotor. Furthermore, we study if an end-to-end learned representation can control a quadrotor based on raw onboard-sensor information only, rendering accurate state estimation from a Kalman filter obsolete. Our results show that recurrent control policies achieve similar performance and robustness as feed-forward policies when acting on state estimates. With raw sensory data, however, recurrent networks offer higher success rates for sim-to-real transfer than feed-forward networks. We also find that recurrent architectures are advantageous when system parameters such as latency are uncertain.

S. Gronauer, D. Stümke, and K. Diepold. "Comparing Quadrotor Control Policies for Zero-Shot Reinforcement Learning under Uncertainty and Partial Observability". In: *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. 2023, pp. 7508–7514. DOI: 10.1109/IROS55552.2023.10341941

Comparing Quadrotor Control Policies for Zero-Shot Reinforcement Learning under Uncertainty and Partial Observability

Sven Gronauer*, Daniel Stümke* and Klaus Diepold

***Abstract*— To alleviate the sample complexity of reinforcement learning algorithms, simulations are a common approach to train control policies before deploying the policy on a real-world robot. However, a gap between simulation and reality generally persists, which endorses the aim to train robust policies already in simulation such that those can be transferred to a real robot at a high success rate. In this paper, we investigate history-dependent policies for drone control in the context of zero-shot transfer learning, where the training is conducted exclusively in simulation. We compare policies represented by feed-forward neural networks with recurrent neural networks and assess both performance and robustness on a real-world quadrotor. Furthermore, we study if an end-to-end learned representation can control a quadrotor based on raw onboard-sensor information only, rendering accurate state estimation from a Kalman filter obsolete. Our results show that recurrent control policies achieve similar performance and robustness as feed-forward policies when acting on state estimates. With raw sensory data, however, recurrent networks offer higher success rates for sim-to-real transfer than feed-forward networks. We also find that recurrent architectures are advantageous when system parameters such as latency are uncertain.**

I. Introduction

The design and synthesis of control policies for robot systems such as drones traditionally require a wide range of expert knowledge. Expertise might be required to formulate an accurate system description, construct a feedback controller, and implement high-level algorithms for planning and decision-making. In cases where neither a system model is available nor the optimization objective can be formulated a-priori (*e.g.*, flying through a cluttered environment [17]), reinforcement learning (RL) promises an automated approach to learn control strategies and expressive representations in an end-to-end fashion through data.

Despite the tremendous success record of RL algorithms [18], [23], the deployment to real-world robots remains a challenge [5]. One major problem is the abundance of data required to train a control policy. A common approach to elude the sample inefficiency is to pre-train policies in simulation and thereafter transfer the policies to the real-world robot. However, a gap between simulation and reality inherently persists, which demands a certain degree of robustness in the controller design so that unmodeled effects and parameter uncertainties can be handled on the real robot. Although prior work on the subject of RL-based control has shown potential in real-world robotics, challenges like partial observability, non-stationarity and delays in acting and sensing lack of practical investigation. In support of that, recent theoretical work emphasized the benefits of using history-dependent policies under randomized dynamics [2].

This paper investigates history-dependent control policies for a tracking task with a quadrotor robot and addresses challenges like partial observability, non-stationarity, and delays in acting and sensing. We focus on the setting of zero-shot transfer learning, where the control policies are exclusively trained in simulation and deployed on a real quadrotor afterward. We use policies that map inputs directly to motor commands and realize history-dependent inputs either by a feed-forward neural network (FNN) with a stacked observation history or by using recurrent neural networks (RNNs). Furthermore, we consider two different observation levels as policy inputs. The first provides state estimates through an extended Kalman filter (EKF), whereas the second leverages only raw sensory data. We investigate if an end-to-end learned representation is able to control a quadrotor based on raw sensor information only, rendering accurate state estimation from a Kalman filter superfluous.

Our results show that recurrent control policies achieve comparable results to FNN policies in terms of performance and robustness when having access to state estimates. On raw sensory data, however, RNNs show higher success rates for sim-to-real transfers than their feed-forward counterparts. By randomizing system parameters, RNNs are capable to adapt to the characteristics of the real world at deployment and, hence, can be beneficial when system parameters like latency are uncertain.

II. Related Work

Learning-based and autonomous drone control is an active research topic [9]. In this section, we focus on related work that addresses the sim-to-real transfer with RL.

A. High-level Control

Based on synthetic data generated with simulators like *Flightmare* [24] or *RotorS* [6], the task of drone racing through moving gates was tackled in [16] and [25]. Both papers proposed a combination of model-predictive control (MPC) with RL to navigate the quadrotor through a racing course based on vision inputs. A policy able to navigate a drone through real-world settings such as forests or snow-covered terrains at high speed was presented in [17]. The authors proposed an end-to-end architecture that maps depth images and state estimates to receding-horizon trajectories.

*Authors with equal contribution

The authors are with TUM School of Computation, Information and Technology, Technical University of Munich, Arcisstr. 21, 80333 Munich, Germany. Contact author: sven.gronauer@tum.de

The authors concluded that accounting for the task's multi-modality and the use of an abstract input representation enables transfer to real-world environments without the need to fine-tune with real data.

B. Low-level Control

Similar to our paper, a substantial body of work considers the data-driven synthesis of control policies that use sensor readings and state estimation to output either attitude targets or motor thrust commands [14]. Both works in [8] and [19] applied proximal policy optimization (PPO) combined with domain randomization (DR) to learn policies for tracking tasks that generalize to many out-of-distribution states and work robustly even when real physical effects were ignored during training. In the former, the authors also aimed to find simulation parameters that fit best to the real world using simulation optimization.

Model-based RL and latent state space models were used in [1] to learn a take-off and tracking policy. Similar to our work, the authors applied RNNs and considered sensor-level observations, but they required a stabilizing proportional–integral–derivative (PID) controller and did not investigate the zero-shot setting.

Other works showed that zero-shot transfers are also possible without DR if simulation parameters are chosen accurately enough [12], [15] or policies are robustified [3]. In contrast to our work, the partially observable case with only raw sensory perception was not investigated in the works presented so far.

III. Preliminaries

A. Quadrotor Dynamics

The dynamics of the quadrotor are modeled by the differential equation

$$\dot{x} = f(x, u) \tag{1}$$

where the drone state $x = [p^T, \dot{p}^T, \varphi^T, \omega^T]^T \in \mathbb{R}^{13}$ contains the position p, the linear velocity $\dot{p}$, the body angle φ in quaternions and the angular speed ω. The rate of change in linear and angular velocities are described by the Newton-Euler equations

$$\ddot{p} = \frac{R}{m}\begin{bmatrix} 0 \\ 0 \\ \sum F_i \end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} \quad \text{and} \quad I\dot{\omega} = \eta - \omega \times (I\omega) \tag{2}$$

in the inertial frame $\mathcal{O}$ with gravity g and the inertia matrix I in the body frame $\mathcal{B}$, respectively. The quadrotor mass is m and $\sum F_i$ is the sum of the up-lifting forces. The rotation matrix R describes the map from $\mathcal{B}$ to $\mathcal{O}$. The torques η acting in the body frame are given by

$$\eta = \begin{bmatrix} \frac{1}{\sqrt{2}} L(-F_1 - F_2 + F_3 + F_4) \\ \frac{1}{\sqrt{2}} L(-F_1 + F_2 + F_3 - F_4) \\ -M_1 + M_2 - M_3 + M_4 \end{bmatrix} \tag{3}$$

with the rotor forces F_i, the corresponding motor torques M_i and the drone's arm length L. An overview of the quadrotor setup and the coordinate frames is depicted in Fig. 1.

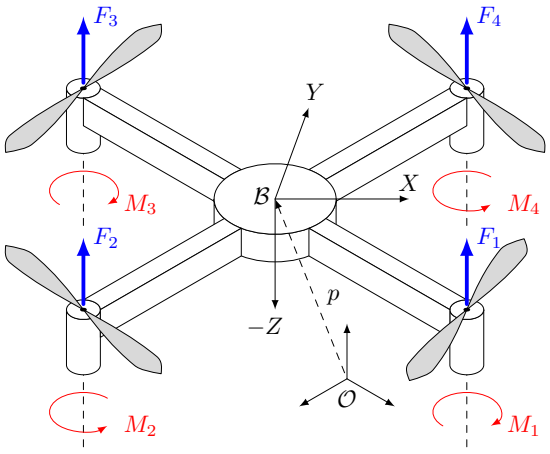


Fig. 1: Illustration and coordinate frames of the quadrotor.

The up-lifting forces $F_i = \frac{mg}{4} k_F \nu_i^2$ are produced by the angular speed ν_i of each rotor i with thrust-to-weight ratio $k_F \in \mathbb{R}$. The rotors also produce a moment according to $M_i = k_{M_1} F_i + k_{M_2}$ with k_{M_1} and k_{M_2} being scalars.

In many works, the motor dynamics are neglected. However, we reported in a previous work that an accurate actuator model is crucial for successful zero-shot transfers [8]. Thus, we regard the motor lag by $T_m \dot{\nu}_i = -\nu_i + \sqrt{u_i}$ with the motor time constant $T_m \in \mathbb{R}$ and the normalized commanded thrust $u_i \in [0, 1]$. An agent action a is converted to a thrust command by $u_i = \frac{1}{2}[\min(\max(a_i, -1), 1) + 1]$. Additionally, we simulate system latency Δ which shall capture all delays emerging in the hardware, *e.g.*, the time between sensor readouts and motor voltage adjustments.

B. Reinforcement Learning

The standard framework to describe RL problems is the Markov decision process (MDP) which is formalized by the tuple $(\mathbb{S}, \mathbb{A}, \mathcal{P}, r, \mu)$, where $\mathbb{S}$ and $\mathbb{A}$ are the state and action space, $\mathcal{P}$ describes the system transition probability, μ denotes the initial state distribution and $r : \mathbb{S} \times \mathbb{A} \to \mathbb{R}$ is the reward function. The goal is to find a policy $\pi : \mathbb{S} \to \mathbb{A}$ such that the expectation over trajectory returns

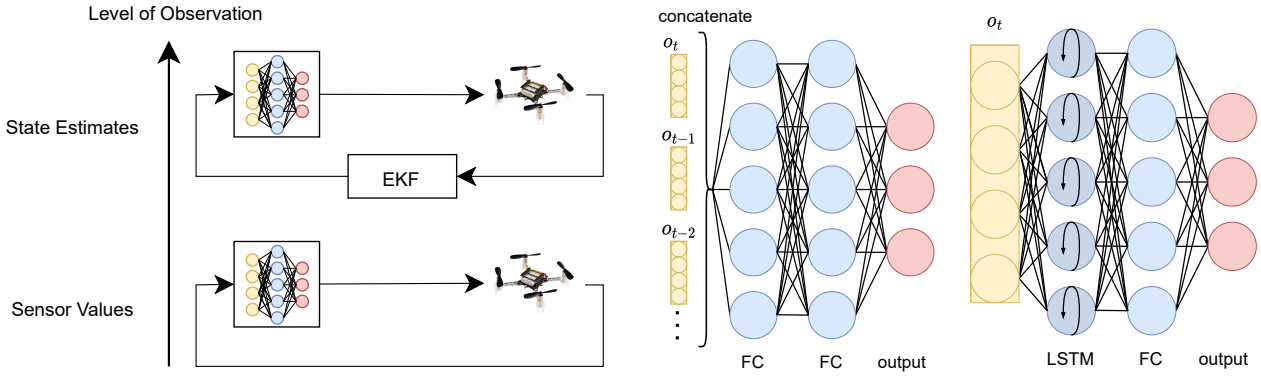
$$\mathbb{E}_{\tau \sim \pi}\left[R(\tau)\right] = \mathbb{E}_{\tau \sim \pi}\left[\sum_{t=0}^{T} \gamma^t r(s_t, a_t)\right] \tag{4}$$

is maximized with the discount factor $\gamma \in (0, 1)$ at time step t. For a trajectory $\tau = (s_0, a_0, s_1, a_1, \dots)$ produced under π with states $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t)$ and $s_0 \sim \mu$, we use the notation $\tau \sim \pi$ as shortcut. We use π_θ to denote that the policy is parametrized by a neural network with weights θ.

Note that we use $s_t = [x_t^T, e_t^T, a_{t-1}^T]^T \in \mathbb{R}^{20}$ instead of x_t to denote that states include additional task information such as the difference vector $e = p - p_{ref} \in \mathbb{R}^3$ to the set-point p_{ref} and the action $a_{t-1}^T \in \mathbb{R}^4$ taken previously.

C. Domain Randomization

During the training in simulation, we sample system parameters ξ from a distribution Ξ at the beginning of each



(a) Hierarchy of observation levels.

(b) Feed-forward architecture.

(c) RNN architecture.

Fig. 2: Methods overview. (a) For the sensor level, raw and noisy sensor readings are used as policy inputs. The network is supposed to learn a meaningful state representation in an end-to-end fashion. For the state estimates, the sensor readings are fused with an EKF to provide accurate state estimates as inputs to the policy. (b-c) Feed-forward and recurrent actor-critic architectures that are compared in our experiments.

trajectory. This results in the system transition probability $s_{t+1} \sim \mathcal{P}(\cdot|s_t, a_t, \xi)$ being dependent on the system parameters. Thus, we search for policy parameters θ that maximize the expected return

$$\max_{\theta} J(\pi_\theta) = \mathbb{E}_{\xi\sim\Xi}\left[\mathbb{E}_{\tau\sim\pi_\theta}\left[R(\tau)\right]\right] \tag{5}$$

over the dynamics induced by the distribution of simulation parameters. This method is known as domain randomization.

D. Partial Observations

In real-world applications, the Markov property rarely holds because the agent can only perceive noisy and partial state information. Therefore, the MDP is extended with a set of observations $\mathbb{O}$ and an observation function that is defined by $\mathcal{O} : \mathbb{X} \times \mathbb{U} \times \mathbb{O} \rightarrow [0, 1]$. The resulting tuple $(\mathbb{X}, \mathbb{U}, \mathbb{O}, \mathcal{P}, \mathcal{O}, r, \mu)$ describes the partially observable Markov decision processes (POMDP), where policies take history-dependent actions $a_t \sim \pi_\theta(\ \cdot \mid o_t, o_{t-1}, ..., o_{t-H+1})$. The parameter $H \in \mathbb{Z}_{\geq 0}$ indicates how many previous observations are considered.

IV. Methods

In this paper, we compare history-dependent drone control policies for a tracking task. Sensing and computation run entirely onboard and the policies directly map input signals to motor commands. The history dependence is realized either by (i) stacking subsequent observations and feeding them into a multi-layer perceptron (MLP) or by (ii) implementing the actor-critic as RNN processing one observation at a time. In addition to that, we compare two observation levels with each other, where one provides accurate state estimates through an EKF, and the other leverages only raw sensory data as policy input. An overview of our methodology can be seen in Figure 2.

A. Recurrent Actor-Critic

Since sample efficiency is not an issue in simulation, we selected PPO [21] as learning algorithm due to its learning stability and simplicity. While a parameterization with FNNs allows sampling experience (s_t, a_t) randomly from time steps $t \in [0, T)$ from a trajectory τ, recurrent networks require updates with consecutive sequences $\tau_{t_s:t_e} = (s_{t_s}, a_{t_s}, s_{t_{s+1}}, a_{t_{s+1}}, ..., s_{t_e}, a_{t_e})$ starting and ending at time step t_s and t_e of trajectory τ respectively. The sequence length of sub-trajectories is going to be denoted as $L = t_e - t_s + 1$. The mini-batches $\mathcal{B}$ are composed of sequences that are allowed to overlap with $K \in [0, L - 1)$ steps, *i.e.*, $\mathcal{B} = \{\tau_{0:L-1}, \tau_{L-K:2L-K-1}, ...\}$. Recurrent PPO intends is to improve performance by inferring useful information like system parameters or hidden state values from the temporal evolution of a trajectory.

Several methods for the initialization of the RNN's memory exist [10], [13]. We apply the *zero-state* initialization, *i.e.*, the internal memory is set to zero at the beginning of each forward pass in the update routine. We also tested more sophisticated initialization methods like the *burn-in* method, which did not yield policy performance improvements.

B. Neural Network Architectures

We use the same architecture for the value and policy networks but do not share parameters. The dimension of a single observation o_t depends on the considered observation level (see Section IV-C). For the FNN architecture (Figure 2b), the input is the history of the last H observations concatenated into one vector. After each fully-connected (FC) hidden layer, ReLU is applied as activation in the policy and tanh in the critic network. The recurrent architecture (Figure 2c) has long short-term memory (LSTM) cells in the first hidden layer followed by one FC layer. After the FC layers, we use ReLU in the policy and tanh in the value network as activation. Since the LSTM already contains

non-linearities, no further activations are applied thereafter. The recurrent architecture is not provided with a history of previous observations since we expect an RNN to infer useful information from the temporal progression of observations.

C. Observation Levels

Besides comparing actor-critic architectures, we are also interested in the information level provided to the agent. As an EKF requires knowledge about the system dynamics, which contradicts the classical model-free RL assumption that no system knowledge is necessary, we study an end-to-end learning approach with raw sensory data as inputs.

We suppose that incomplete state information requires recurrent structures or a large history size for FNNs. Therefore the following observation levels and their implication on the sim-to-real performance are compared:

1) *State Estimates.* Estimated values of the position, linear velocity, angular orientation, and velocities are used as policy inputs. On the real drone, these values are computed by an EKF fusing on-board sensor readings.
2) *Raw Sensor Data.* Linear accelerations and angular velocities measured by an inertial measurement unit and absolute height measurements from a LiDAR are fed into the policy.

Both perception models are visualized in Figure 2a.

V. Experiments

In our experiments, we want to find answers to the following research questions:

1) Is it beneficial to use RNNs in terms of performance and robustness compared to FNNs for zero-shot RL?
2) Can we learn a policy that directly maps raw sensory data to motor commands in an end-to-end fashion?
3) Does the memory state encode meaningful information in recurrent architectures?

The first question attempts to get an answer if RNNs are superior to FNNs for the sim-to-real transfer of low-level drone control policies. We hypothesize that recurrent architectures improve sim-to-real performance because they enable adaptive control by encoding the underlying system parameters in the memory state at deployment.

In the second question, we hypothesize that recurrent networks are essential for low-level sensor-based observations since they can infer long-term dependencies from the inputs. Because a control policy needs to capture unknown system states from noisy sensor data, an FNN policy can require a large window of previous observations, which exceeds the computational resources of the hardware platform.

In the third and last question, we seek to find evidence that an RNN encodes information about the system's properties in its latent memory state.

A. Experimental Setup

We conducted experiments for both sim-to-sim and sim-to-real scenarios to compare history-dependent actor-critic network architectures for their applicability to low-level drone control. The real-world experiments were conducted indoors in a laboratory setting. The experimental setup is depicted in Figure 3.

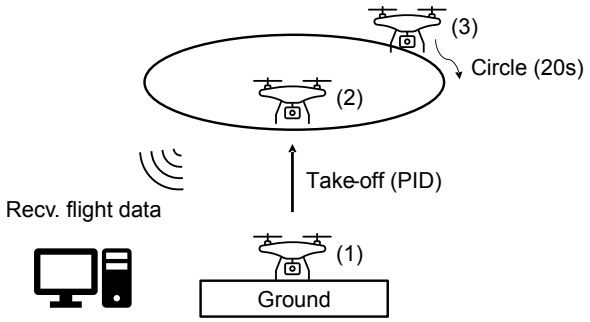


Fig. 3: We bring the drone from the ground (1) to the start position (2) with a PID controller. At the beginning of each episode, the motor control is given to the learned policy.

a) Hardware Platform: As robot, we used the *CrazyFlie 2.1* quadrotor, which is very agile due to its small size and light weight. The drone is equipped with a 3-axis gyroscope, a 3-axis accelerometer, and a z-axis LiDAR. In our experiments, we only used onboard sensor measurements and no external tracking system. For evaluation purposes, we transmitted the flight information via *CrazyRadio* to a host computer where the logged data were analyzed.

b) Learning Task: The *phoenix-drone-simulation*[1] based on the *PyBullet* [4] physics engine is used as training environment. The learning task is to follow a trajectory described by a circle of $0.25\,\mathrm{m}$ radius in clock-wise direction with a period of $3\,\mathrm{s}$. An episode terminates early if the drone becomes unstable or gets too far off the reference. The maximum flight time is set to $5\,\mathrm{s}$ in simulation, after which the environment is reset. At the beginning of each episode, the drone gets randomly initialized at a point ($\pm 0.05\,\mathrm{m}$) near the reference trajectory at a height of $1\,\mathrm{m}$. Additionally, the velocities and orientation are drawn from uniform distributions. To incentivize the drone for set-point tracking, the reward function is designed as

$$r(s_t, a_t) = r_f - \|e\|_2 - 10^{-4}\|\frac{1}{2}(a_t + 1)\|_2 - 10^{-3}\|\omega_t\|_2 - 10^{-3}\|a_{t-1} - a_t\|_2 \quad (6)$$

where e is the distance from the drone to the reference. The terminal reward is $r_f = -100$ if $\|e\|_2 > 0.25$ else $r_f = 0$.

c) Domain Randomization: Numerous system parameters can be randomized within the simulation to incentivize robustly trained policies. We resampled the drone mass m, diagonal entries of the inertia matrix I, first-order motor time constants T_m, both force-torque-factors k_{M_1} and k_{M_2}, thrust-to-weight ratio k_F and the physics sampling period T_{phys} at the beginning of each episode. We draw their value from a uniform distribution around their nominal value, *i.e.*, $\xi^\star + \xi^\star \mathcal{U}(-\mathrm{DR}, \mathrm{DR})$. Also, the system latency Δ can be

[1] https://github.com/SvenGronauer/phoenix-drone-simulation

TABLE I: **Sim-to-sim** results. The median cumulative reward is obtained over 500 episodes in simulation. The best value in each column is highlighted with bold font. (a) Results with policies that take as inputs raw sensory data. (b) State-level observations obtained from an EKF as policy inputs.

(a) **Sensor**-level observations, $\Delta = 20$ms

	$\mathbf{DR_S}$	0		0.1	
Actor	$\mathbf{DR_T}$	0.1	0.2	0.1	0.2
Recurrent	H=1	-105.8	-105.8	**-39.9**	-104.7
Forward	H=2	-111.6	-107.2	-103.9	-103.9
	H=4	-44.6	-105.2	-49.2	-105.3
	H=8	**-40.3**	**-104.9**	-41.0	**-103.7**

(b) **State**-level observations, $\Delta = 20$ms

	$\mathbf{DR_S}$	0		0.1	
Actor	$\mathbf{DR_T}$	0.1	0.2	0.1	0.2
Recurrent	H=1	**-15.7**	-30.5	**-12.9**	**-22.5**
Forward	H=2	-15.9	**-28.2**	-13.8	-22.7
	H=4	-18.9	-33.6	-13.7	-23.8
	H=8	-22.8	-53.4	-15.5	-29.3

randomly drawn from a discrete set of values. The physics simulation runs at $200\,\mathrm{Hz}$ while the control loop has an update frequency of $100\,\mathrm{Hz}$.

d) Actions: The actions a_t are limited to $[-1, 1]$ to account for actuator limits. Depending on the selected system latency Δ, actions are applied instantaneously or with a delay of Δ / T_{phys} time steps.

B. Hyperparameters

a) Training: We aligned the hyper-parameters to the ones suggested in [7] and [11]. A distributed learner setup was applied where the policy gradients were computed and averaged across 64 workers with a total batch size of 64000. We trained with the PPO algorithm over 500 iterations with the discount factor $\gamma = 0.99$ and clipping factor 0.2. Observations were standardized using first and second-order statistics, which were estimated with Welford's algorithm. We stopped updating the mean and variance after 125 iterations since this led to better and more consistent results.

b) Actor-Critic: Neural networks were optimized with Adam, where the learning rate of the critic was 0.001 and 0.0003 for the actor. To reduce the variance of critic estimates, we applied generalized advantage estimation [20] with the weighting factor $\lambda = 0.95$. Over the training, we used stochastic policies where the policy output is the mean of a multi-variate Gaussian distribution $a \sim \mathcal{N}(\pi(s), \epsilon I)$ with I being the identity matrix and $\epsilon \in \mathbb{R}$ being the exploration noise that was linearly annealed from 0.5 to 0.01. Due to the computational limitations of the microcontroller on the *CrazyFlie 2.1*, we set the number of hidden units to 32 for FC and 16 for LSTM layers. Further, we used two hidden layers for all networks. The networks were trained with 32-bit floats but were quantized to 16-bit on the microcontroller. As a result, the execution time of a forward pass on the target hardware is faster than $1.5\,\mathrm{ms}$ for all policies. Since the critic networks were only used during training, we increased their network size, *i.e.*, our value networks had 300 units in each FC and 128 neurons in the LSTM layers.

VI. Results and Discussion

A. Sim-to-Sim Results

We first ran transfer learning experiments from simulation to simulation to assess the performance and the impact of the history size H on the learned policies. For this matter, policies trained with source domain randomization $\mathrm{DR}_S \in \{0.0, 0.1\}$ were transferred to environments with target domain randomization $\mathrm{DR}_T \in \{0.1, 0.2\}$. We used a fixed latency of $\Delta = 20$ms for this experiment. The results are reported as the median cumulative reward in Table I. Each value was computed using 500 episodes with randomly sampled parameters and initial conditions.

When sensor-level observations are used, the performance of feed-forward architectures significantly improves with increasing history size. With $H = 8$ the FNN architecture achieves comparable results to the recurrent architecture. At state level, a smaller history size of $H = 2$ is sufficient for the feed-forward architecture to be on par with the RNNs. A larger value of H deteriorates the cumulative reward, indicating the importance of tuning this parameter. Further, recurrent policies improve performance when domain randomization is used during training, *i.e.*, $\mathrm{DR}_S > 0$.

B. Sim-to-Real Results

Based on the sim-to-sim results, we selected the FNN policies $H = 2$ for state-level observations and $H = 8$ for sensor-level observations for the deployment on the real drone. For recording real-world flight data, we trained five policies with different random seeds for each architecture configuration. To get the drone into an initial state for the *Circle* task, a PID controller was used to take off the ground. After $20\,\mathrm{s}$ of successful flying, the task was manually stopped. We repeated this procedure three times for each policy. The resulting flight times and success rates are presented in Table II.

Almost all policies were able to fly the quadrotor for $20\,\mathrm{s}$ when the state-level observations were used. In cases where no domain randomization has been applied during training, the policies produced smoother trajectories but flew at a lower altitude than 1m. This can be explained by the overfitting of the agent to the fixed system parameters in the non-randomized setup. In contrast, policies in randomized settings learned to compensate the altitude error by varying rotor speeds accordingly since the policies have seen different thrust-to-weight ratios and other changing parameters during the training.

With raw sensory data as observations, the success rates were lower than with state estimates. Feed-forward architectures led to unstable behavior and, thus, terminated early in most of the trials, resulting in success rates less than 7%. The best result in terms of flight time was achieved with the recurrent policy that has been trained with domain

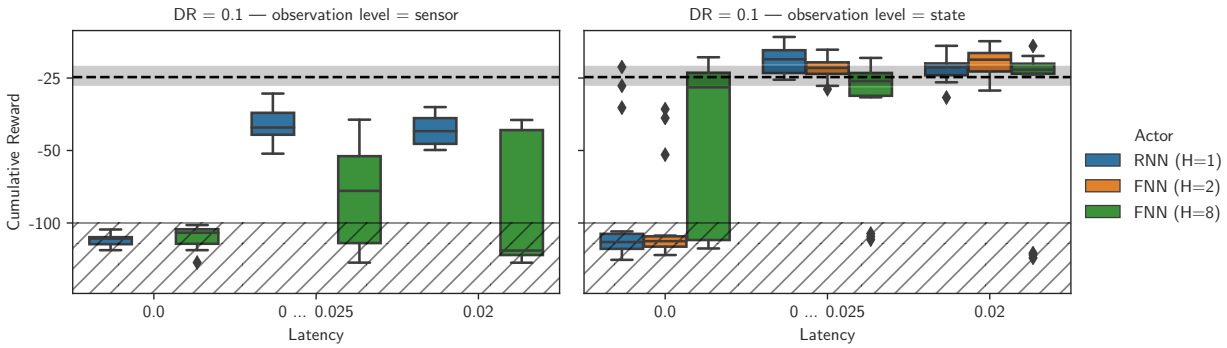


Fig. 4: Real-world cumulative reward for the sensor-level and state-level observations. The hatched area marks the reward range where the drone crashed and the gray area is the baseline performance achieved by a PID controller. (Left) Raw sensory data as policy inputs. (Right) State estimates as policy inputs.

TABLE II: **Sim-to-real** results. Real-world flight time and success rates are shown with a fixed latency $\Delta = 20\,\mathrm{ms}$.

Obs.	DR	Actor	Success [%]	Flight Time [s]		
				min	max	median
State	0	FNN (H=2)	100.0	20.0	20.0	20.0
		RNN (H=1)	100.0	20.0	20.0	20.0
	0.1	FNN (H=2)	100.0	20.0	20.0	20.0
		RNN (H=1)	93.3	5.3	20.0	20.0
Sensor	0	FNN (H=8)	6.7	0.7	20.0	2.1
		RNN (H=1)	0.0	3.2	12.1	4.9
	0.1	FNN (H=8)	6.7	2.5	20.0	4.3
		RNN (H=1)	66.7	7.1	20.0	20.0

randomization. It had a success rate of 66.7% and a median flight time of 20s.

The results show that a quadrotor control policy can indeed be learned in an end-to-end fashion. However, the average performance is worse than using state-based estimates from a Kalman filter. The performance improvements of RNNs over FNNs on sensor level can be explained by the RNN being able to reconstruct the state vector, whereas at state level the RNN cannot take advantage of this because the EKF already reconstructs the state. Also, all emerging lag behaviors seem to be sufficiently captured by an FNN with $H \geq 2$. The results underpin the importance of domain randomization combined with a recurrent architecture when only raw sensory data are available.

C. Impact of Latency

As we reported in [8], the correct choice of system latency Δ is crucial for a successful zero-shot policy transfer on quadrotors. While our previous experiment showed that $\Delta = 20\,\mathrm{ms}$ is a suitable choice for modeling the hardware latency and yields successful policy transfers, we also want to investigate if history-dependent policies can handle an uncertain latency. Hence, we train policies with latency values drawn from the set $\{0, 5, 10, 15, 20, 25\}\mathrm{ms}$ at the start of each episode. As an ablation study, we also evaluated policies that were trained with no latency, *i.e.*, $\Delta = 0\,\mathrm{ms}$.

The results from the real-world flights are depicted in Figure 4, which shows the performance evaluation in terms of the cumulative reward over 500 time steps. Besides the learned policies, we used a position PID controller specifically tuned on the reward function (6) and lag compensation as a benchmark baseline.

The results demonstrate that randomizing the latency still leads to transferable policies. For feed-forward architectures, the cumulative reward decreases, especially when sensor-based observations are provided as policy inputs. The recurrent architectures benefit from latency randomization and surpass the previous results under fixed latency. When zero latency is considered during the training in simulation, none of the policy architectures achieved satisfying results.

The results confirm our hypothesis that recurrent architectures can improve sim-to-real performance when using DR during training. We suppose that is due to the ability of RNNs to infer the actual system parameters from the sequence of observations and encode the estimate in the latent state. This results in a policy that can adapt to varying system parameters such as latency. Congruent results were shown for bipedal robots, where RNNs outperformed FNNs when trained using DR to prevent overfitting [22]. However, we did not find evidence that the use of RNNs yields robustness benefits over FNNs when facing uncertain system parameters.

D. Decoding of Recurrent Memory

Motivated by the former experimental results, we investigated if RNN memory states encode information about the system latency. Therefore, we trained an MLP classifier that predicts the simulated system latency from latent memory states. The train and test data sets were obtained by running RNN policies in simulation with latency $\Delta \in \{5, 15, 25\}\mathrm{ms}$, and sampled 75000 data points for both observation levels. We also trained a baseline classifier for comparison, which maps single observations to the unknown latency.

With raw sensory observations, the validation accuracy was $65\,\%$ and $59.8\,\%$ for the memory and baseline classifier respectively. This result indicates that the RNN policy extracts and encodes useful information about the latency from the sequence of partial observations in the latent memory.

In contrast, we obtained only a minor difference of $1\,\%$ in favor of the memory classifier, with $72.8\,\%$ against $71.8\,\%$ for state-based observations. Obviously, a strong prior on the latency can be found with full state information in combination with the last selected action. This conclusion is in accordance with [26], who reported that relevant dynamics parameters can be recovered from a state-action history.

VII. Conclusion

We demonstrated in this paper that the synthesis of history-dependent drone control policies could be achieved in a zero-shot transfer learning scenario, using onboard sensor information only and mapping observations directly to motor commands. Our results confirm the common practice of using DR to reduce the reality gap. Without dynamics randomization, an over-fitting to the simulation parameters can be observed.

An important hyper-parameter for FNN policies is the history size of previous observation-action inputs, decreasing sim-to-real performance if not chosen properly. To discard this sensitive parameter, recurrent architectures can be used to process only the most recent observation at the cost of a higher training time.

Our experiments showed that learning-based drone policies achieved better performance results than the PID controller baseline. In addition to that, we demonstrated that it is even possible to control the quadrotor robot with an end-to-end learning approach, where the agent maps raw sensory data to low-level motor commands. Albeit our experiments did not perform as good as the PID baseline in this setup, it is worth conducting further research in this direction since the amount of expert knowledge used in the problem design can be reduced significantly.

Additional experiments with altered system latency values showed that an artificially introduced reality gap can render the sim-to-real transfer impossible. When applying DR, including the latency parameter during training, the tested recurrent architectures improved the sim-to-real performance and outperformed a carefully chosen constant value. We suppose that the ability of RNNs to infer the actual system parameters from the sequence of observations and encode them in the latent state is responsible for superior results.

References

[1] P. Becker-Ehmck, M. Karl, J. Peters, and P. van der Smagt, "Learning to fly via deep model-based reinforcement learning," *CoRR*, vol. abs/2003.08876, 2020.

[2] X. Chen, J. Hu, C. Jin, L. Li, and L. Wang, "Understanding domain randomization for sim-to-real transfer," in *Proceedings of the International Conference on Learning Representations, ICLR*, 2022.

[3] Y. Cheng, P. Zhao, F. Wang, D. J. Block, and N. Hovakimyan, "Improving the robustness of reinforcement learning policies with $\updownarrow_1$ adaptive control," *IEEE Robotics and Automation Letters*, vol. 7, no. 3, pp. 6574–6581, 2022.

[4] E. Coumans and Y. Bai, "Pybullet, a python module for physics simulation for games, robotics and machine learning," 2021.

[5] G. Dulac-Arnold, N. Levine, D. J. Mankowitz, J. Li, C. Paduraru, S. Gowal, and T. Hester, "Challenges of real-world reinforcement learning: definitions, benchmarks and analysis," *Machine Learning*, vol. 110, no. 9, pp. 2419–2468, 2021.

[6] F. Furrer, M. Burri, M. Achtelik, and R. Siegwart, *Robot Operating System (ROS): The Complete Reference (Volume 1)*. Springer International Publishing, 2016, ch. RotorS—A Modular Gazebo MAV Simulator Framework, pp. 595–625.

[7] S. Gronauer, M. Gottwald, and K. Diepold, "The successful ingredients of policy gradient algorithms," in *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI*, 2021, pp. 2455–2461.

[8] S. Gronauer, M. Kissel, L. Sacchetto, M. Korte, and K. Diepold, "Using simulation optimization to improve zero-shot policy transfer of quadrotors," in *2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2022, pp. 10 170–10 176.

[9] D. Hanover, A. Loquercio, L. Bauersfeld, A. Romero, R. Penicka, Y. Song, G. Cioffi, E. Kaufmann, and D. Scaramuzza, "Autonomous drone racing: A survey," 2023.

[10] M. J. Hausknecht and P. Stone, "Deep recurrent q-learning for partially observable mdps," *CoRR*, vol. abs/1507.06527, 2015.

[11] P. Henderson, R. Islam, P. Bachman, J. Pineau, D. Precup, and D. Meger, "Deep reinforcement learning that matters," in *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*. AAAI Press, 2018, pp. 3207–3214.

[12] J. Hwangbo, I. Sa, R. Siegwart, and M. Hutter, "Control of a quadrotor with reinforcement learning," *IEEE Robotics and Automation Letters*, vol. 2, no. 4, pp. 2096–2103, 2017.

[13] S. Kapturowski, G. Ostrovski, J. Quan, R. Munos, and W. Dabney, "Recurrent experience replay in distributed reinforcement learning," in *Proceedings of the International Conference on Learning Representations, ICLR*, 2019.

[14] E. Kaufmann, L. Bauersfeld, and D. Scaramuzza, "A benchmark comparison of learned control policies for agile quadrotor flight," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 10 504–10 510.

[15] J. E. Kooi and R. Babuška, "Inclined quadrotor landing using deep reinforcement learning," in *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2021, pp. 2361–2368.

[16] A. Loquercio, E. Kaufmann, R. Ranftl, A. Dosovitskiy, V. Koltun, and D. Scaramuzza, "Deep drone racing: From simulation to reality with domain randomization," *IEEE Transactions on Robotics*, vol. 36, no. 1, pp. 1–14, 2020.

[17] A. Loquercio, E. Kaufmann, R. Ranftl, M. Müller, V. Koltun, and D. Scaramuzza, "Learning high-speed flight in the wild," *Science Robotics*, vol. 6, no. 59, 2021.

[18] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, pp. 529 EP –, 02 2015.

[19] A. Molchanov, T. Chen, W. Hönig, J. A. Preiss, N. Ayanian, and G. S. Sukhatme, "Sim-to-(multi)-real: Transfer of low-level robust control policies to multiple quadrotors," *CoRR*, vol. abs/1903.04628, 2019.

[20] J. Schulman, P. Moritz, S. Levine, M. Jordan, and P. Abbeel, "High-dimensional continuous control using generalized advantage estimation," in *Proceedings of the International Conference on Learning Representations, ICLR*, 2016.

[21] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," 2017.

[22] J. Siekmann, S. Valluri, J. Dao, L. Bermillo, H. Duan, A. Fern, and J. Hurst, "Learning memory-based control for human-scale bipedal locomotion," in *Robotics: Science and Systems*, 2020.

[23] D. Silver, T. Hubert, J. Schrittwieser, I. Antonoglou, M. Lai, A. Guez, M. Lanctot, L. Sifre, D. Kumaran, T. Graepel, T. Lillicrap, K. Simonyan, and D. Hassabis, "A general reinforcement learning algorithm that masters chess, shogi, and go through self-play," *Science*, vol. 362, no. 6419, pp. 1140–1144, 2021/12/10 2018.

[24] Y. Song, S. Naji, E. Kaufmann, A. Loquercio, and D. Scaramuzza, "Flightmare: A flexible quadrotor simulator," in *Conference on Robot Learning*, 2020.

[25] Y. Song and D. Scaramuzza, "Policy search for model predictive control with application to agile drone flight," *IEEE Transactions on Robotics*, vol. 38, no. 4, pp. 2114–2130, 2022.

[26] D. Zhang, A. Loquercio, X. Wu, A. Kumar, J. Malik, and M. W. Mueller, "Learning a single near-hover position controller for vastly different quadcopters," *arXiv preprint arXiv:2209.09232*, 2023.

Comparing Quadrotor Control Policies for Zero-Shot Reinforcement Learning under Uncertainty and Partial Observability

Conference Proceedings: 2023 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
Author: Sven Gronauer
Publisher: IEEE
Date: October 01, 2023

Reprint Permission

The IEEE does not require individuals working on a thesis to obtain a formal reuse license, however, you may print out this statement to be used as a permission grant:
Requirements to be followed when using any portion (e.g., figure, graph, table, or textual material) of an IEEE copyrighted paper in a thesis:

1. In the case of textual material (e.g., using short quotes or referring to the work within these papers) users must give full credit to the original source (author, paper, publication) followed by the IEEE copyright line © 2011 IEEE.

2. In the case of illustrations or tabular material, we require that the copyright line © [Year of original publication] IEEE appear prominently with each reprinted figure and/or table.

3. If a substantial portion of the original paper is to be used, and if you are not the senior author, also obtain the senior author's approval.

Requirements to be followed when using an entire IEEE copyrighted paper in a thesis:

1. The following IEEE copyright/ credit notice should be placed prominently in the references: © [year of original publication] IEEE. Reprinted, with permission, from [author names, paper title, IEEE publication title, and month/year of publication]

2. Only the accepted version of an IEEE copyrighted paper can be used when posting the paper or your thesis on-line.

3. In placing the thesis on the author's university website, please display the following message in a prominent place on the website: In reference to IEEE copyrighted material which is used with permission in this thesis, the IEEE does not endorse any of [university/educational entity's name goes here]'s products or services. Internal or personal use of this material is permitted. If interested in reprinting/republishing IEEE copyrighted material for advertising or promotional purposes or for creating new collective works for resale or redistribution, please go to `http://www.ieee.org/publications_standards/publications/rights/rights_link.html` to learn how to obtain a License from RightsLink.

Paper IV

Reinforcement Learning with Ensemble Model Predictive Safety Certification

Sven Gronauer, Tom Haider, Felippe Schmoeller Roza and Klaus Diepold

Abstract

Reinforcement learning algorithms need exploration to learn. However, unsupervised exploration prevents the deployment of such algorithms on safety-critical tasks and limits real-world deployment. In this paper, we propose a new algorithm called *Ensemble Model Predictive Safety Certification* that combines model-based deep reinforcement learning with tube-based model predictive control to correct the actions taken by a learning agent, keeping safety constraint violations at a minimum through planning. Our approach aims to reduce the amount of prior knowledge about the actual system by requiring only offline data generated by a safe controller. Our results show that we can achieve significantly fewer constraint violations than comparable reinforcement learning methods.

S. Gronauer, T. Haider, F. Schmoeller da Roza, and K. Diepold. "Reinforcement Learning with Ensemble Model Predictive Safety Certification". In: *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*. 2024

Reinforcement Learning with Ensemble Model Predictive Safety Certification

Sven Gronauer
Technical University of Munich (TUM)
Munich, Germany
sven.gronauer@tum.de

Tom Haider
Fraunhofer IKS
Munich, Germany
tom.haider@iks.fraunhofer.de

Felippe Schmoeller da Roza
Fraunhofer IKS
Munich, Germany
felippe.schmoeller.da.roza@iks.fraunhofer.de

Klaus Diepold
Technical University of Munich (TUM)
Munich, Germany
kldi@tum.de

ABSTRACT

Reinforcement learning algorithms need exploration to learn. However, unsupervised exploration prevents the deployment of such algorithms on safety-critical tasks and limits real-world deployment. In this paper, we propose a new algorithm called *Ensemble Model Predictive Safety Certification* that combines model-based deep reinforcement learning with tube-based model predictive control to correct the actions taken by a learning agent, keeping safety constraint violations at a minimum through planning. Our approach aims to reduce the amount of prior knowledge about the actual system by requiring only offline data generated by a safe controller. Our results show that we can achieve significantly fewer constraint violations than comparable reinforcement learning methods.

KEYWORDS

Reinforcement Learning, Safe Reinforcement Learning, Safe Exploration, Predictive Safety Filter, Model-based Learning,

ACM Reference Format:
Sven Gronauer, Tom Haider, Felippe Schmoeller da Roza, and Klaus Diepold. 2024. Reinforcement Learning with Ensemble Model Predictive Safety Certification. In *Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), Auckland, New Zealand, May 6 – 10, 2024*, IFAAMAS, 9 pages.

1 INTRODUCTION

Deep reinforcement learning (RL) is a powerful data-driven paradigm for learning control strategies and has recently achieved remarkable results in various domains [23, 32]. RL is beneficial in situations where the system dynamics are not known or only partially available, but data can be generated through interaction with the environment. However, gathering data in safety-critical tasks and real-world systems is not trivial since the agent is required to act safely at all times, *e.g.*, the actions taken by a robot are supposed to satisfy a series of state and control input constraints to avoid harm to itself or its environment. On top of the difficulty of providing safety, other self-imposed challenges, such as the high sample complexity and the lack of interpretability, impede the adoption of deep RL in real-world applications that go beyond simulations and fail-safe academic environments [12].

The majority of safe RL algorithms are designed to merely incentivize safety instead of ensuring hard safety constraint satisfaction [6]. Policies are encouraged to maximize the task performance while constraint violations are in expectation less or equal to a predefined safety threshold, resulting in safety constraint satisfaction at the end but not throughout training. In contrast, methods derived from model predictive control (MPC) provide formal safety guarantees by making rather strong assumptions and, thus, are a popular choice for designing safe controllers. Yet, their applicability is often limited to low-dimensional systems, as evidenced in [5, 16]. Deep RL, however, has the potential of scaling to high-dimensional problems [1, 34].

In this paper, we introduce a novel algorithm called ensemble model predictive safety certification (X-MPSC) that extends model-based deep RL with tubed-based MPC to certify potentially unsafe actions taken by a learning agent. The result is an algorithm that combines the best of both frameworks by leveraging an ensemble of probabilistic neural networks (NNs) to approximate the system dynamics trained on data sampled from the environment. To provide safe exploration, MPC is used to plan multiple tube-based trajectories that enforce all given safety constraints based on the NN ensemble. The actions of an RL-based agent are modified to safe ones if necessary. For initialization, our method only requires offline data collected by a low-performing but safe controller. The experimental results demonstrate that our algorithm can significantly reduce the number of constraint violations compared to alternative constrained RL algorithms. Constraint violations can even be reduced to zero when a coarse prior system dynamics model is incorporated into the learning loop.

2 RELATED WORK

The artificial intelligence community has different notions of what constitutes a safe system [4, 18]. For RL, safety can be achieved by preventing error states, *i.e.*, undesirable states from which the system's original state cannot be recovered. Error states are tightly coupled to the concepts of reachability [24] and set invariance [3]. Another perspective is to define an RL system to be safe when it

Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024), N. Alechina, V. Dignum, M. Dastani, J.S. Sichman (eds.), May 6 – 10, 2024, Auckland, New Zealand. © 2024 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org).

is able to maximize a performance measure while fulfilling or encouraging safety constraints during both learning and deployment phases [13, 26, 29]. In this paper, we adopt the latter safety definition as an instance of the constrained Markov decision process (CMDP) framework [2].

In a recent survey, Brunke et al. [6] unify the perspectives from both control theory and RL and provide a systematic overview of safe learning-based control in the context of robotics. The authors classify the task of achieving safe learning-based control into three main categories: (1) the formal certification of safety, (2) RL approaches that encourage safety, and (3) the improvement of system performance by safely learning the uncertain dynamics. Because our proposed algorithm addresses safety certification but compares to safety-encouraging RL methods, we limit the following literature review to these two categories.

Formal Certification. Methods from this category utilize prior system dynamics knowledge to provide rigorous safety through hard constraint satisfaction. One way to achieve this is to use *safety filters* such as model predictive safety certification (MPSC) that adapt input actions as minimal as possible to fulfill safety constraints [37, 38]. Another kind of safety filter is a control barrier function (CBF), which is a mechanism to prevent the system from entering unsafe regions. A CBF maps the state space to a scalar value and is defined to change signs when the system enters an unsafe region of the state space [3]. Cheng et al. [7] showed that CBFs could be integrated into model-free RL to achieve safe exploration for continuous tasks, while Robey et al. [30] learned a CBF from expert demonstrations. Luo and Ma [21] used barrier certificates to certify the stability of a closed dynamical system. By iteratively learning a dynamics model and a barrier certificate alongside a policy, they can ensure that no safety violations occur during training. However, their experimental evaluation was conducted on very low-dimensional state spaces. Similar to our work, Koller et al. [16] used learning-based stochastic MPC for multi-step look ahead predictions to correct any potentially unsafe actions based on a single probabilistic Gaussian process (GP) model. Another work by Pfrommer et al. [27] proposed a chance-constrained MPC approach that uses a safety penalty term in the objective to guide policy gradient updates.

Encouraging Safety. The standard formulation in safe RL is to model the environment as a CMDP [29] to learn how to respect safety thresholds while learning the task, leading to soft constraint satisfaction in most cases. One instance is *Lagrangian relaxation methods*, which add a penalty term for constraint violations to the objective such that an unconstrained optimization problem is solved instead [8, 20, 34]. Another common approach is to perform a *constrained policy search* where typically the cost objective function is linearized around the current policy iterate [1, 39]. Lastly, *action projection methods* are applied to correct actions taken by an agent and turn the actions into safe ones, *e.g.*, via Lyapunov functions [9], in closed form with linearized cost models [11] or by evaluating risk-aware Q-functions [33]. Thananjeyan et al. [35] simultaneously learned a task policy, focused solely on task performance, and a recovery policy, activated when constraint violation is likely, which guides the agent back to a safe state. By separating task performance and constraint satisfaction into two separate policies, safety and reward maximization are balanced more efficiently.

Our algorithm extends previous methods based on MPC by relaxing the requirement of prior knowledge about the system dynamics or knowing the terminal set *a priori*. To the best of our knowledge, this is the first work that integrates an NN ensemble into the tube-based MPC framework. Luo and Ma [21] utilized an NN ensemble together with barrier certificates, while Lütjens et al. [22] deployed an ensemble of recurrent NNs for predictive uncertainty estimates realized by Monte Carlo dropout and bootstrapping.

3 PRELIMINARIES

Throughout this work, we consider the dynamics of a system in discrete time described by

$$x_{t+1} = f(x_t, u_t, w_t), \tag{1}$$

with states $x \in \mathbb{X}$, actions $u \in \mathbb{U}$, and disturbances $w \in \mathbb{W}$ at time step t. We assume that the disturbances are bounded and that the system dynamics are Lipschitz continuous. Further, we assume that the system is subject to polytopic constraints in the states and actions, *i.e.*, $x \in \mathbb{X} = \{x \in \mathbb{R}^{n_x} \mid H_x x \leq d_x\}$ and $u \in \mathbb{U} = \{u \in \mathbb{R}^{n_u} \mid H_u u \leq d_u\}$.

3.1 Deep Reinforcement Learning

The standard framework for RL problems is the Markov decision process (MDP) which is formalized by the tuple $(\mathbb{X}, \mathbb{U}, f, r, \mathbb{X}_0)$, where the system f underlies a random disturbance w_t with the transition probability distribution given by $x_{t+1} \sim p(\cdot|x_t, u_t)$. The set $\mathbb{X}_0$ denotes the initial state distribution, and $r : \mathbb{X} \times \mathbb{U} \to \mathbb{R}$ is the reward function. Note that the dynamics model from Equation (1) is equivalently described by the state probability function $p(x_{t+1}|x_t, u_t)$. The optimization objective of RL is given by

$$\underset{\theta}{\text{maximize}}\ J_{\text{RL}}(\pi_\theta) = \mathbb{E}_{\tau\sim\pi_\theta}\left[\sum_{k=0}^{\infty} \gamma^k r(x_t, u_t)\right],$$

where the expected return along the trajectories $\tau = (x_0, u_0, x_1, \dots)$ produced under the policy π_θ is optimized. A policy $\pi_\theta : \mathbb{X} \to P(\mathbb{U})$ describes the mapping from states to a distribution over actions, where the vector θ parameterizes the NN representing the policy. The shortcut $\tau \sim \pi_\theta$ describes trajectories generated under policy π_θ given $x_{t+1} \sim p(\cdot|x_t, u_t)$, $u_t \sim \pi_\theta(\cdot|x_t)$, and $x_0 \sim \mathbb{X}_0$. Finally, $\gamma \in (0, 1)$ denotes the discount factor.

3.2 Model Predictive Safety Certification

The nominal MPSC problem as introduced in [36] seeks a control input v_0 that changes the learning input u_t as minimal as possible by solving the objective

$$\begin{aligned}
\underset{v_0,\dots,v_{N-1}}{\text{minimize}}\quad & \|u_t - v_0\|_2^2 & \\
\text{subject to}\quad & z_0 = x_t & \\
& z_{k+1} = f_{\text{prior}}(z_k, v_k) & \forall\, k = [0, N-1] \\
& v_k \in \mathbb{U} & \forall\, k = [0, N-1] \\
& z_k \in \mathbb{X} & \forall\, k = [0, N] \\
& z_N \in \mathbb{X}_{\text{term}} &
\end{aligned} \tag{2}$$

over a finite horizon N such that the nominal state-action sequence $(z_0, v_0, \dots, v_{N-1}, z_N)$ lies within the given state-action constraints.

Note that we use t as the time index for state measurements and actions that are applied to (1), whereas k indicates states and actions used for planning such that the predicted states z_k are k stages ahead of time step t. The terminal set $\mathbb{X}_{\text{term}}$ acts as a constraint that must be reached from x_t within N stages. MPSC assumes access to a model f_{prior}, which is usually derived from first principles. MPSC was initially proposed for linear systems [37] and extended to systems with nonlinear dynamics in later works [36, 38].

3.3 Tube-based Model Predictive Control

If uncertainties and disturbances exist in the system under control, feedback control is superior to open loop control. While conventional MPC finds a nominal action sequence as a solution for the open loop control problem, robust MPC returns a sequence of feedback policies. In this paper, we focus on *tube-based* MPC [28] as implementation to approach robustness. Tube-based MPC utilizes a model f_{prior} to plan a nominal state trajectory $(z_0, \dots, z_N)$ associated with the action sequence $(v_0, \dots, v_{N-1})$ based on the latest measurement x_t from (1). In presence of uncertainty, it is assumed that the tube contains all possible realizations of the actual system, where each realization implements a series of disturbances. Since tubes can grow large under uncertainty, a closed loop feedback

$$u_{t+k} = v_k + K(x_{t+k} - z_k) \tag{3}$$

is used to track the state trajectory $(x_t, x_{t+1}, \dots)$ of the actual system toward the nominal trajectory. The matrix $K \in \mathbb{R}^{n_u \times n_x}$ implements the feedback and is chosen such that the error system $e_k = x_{t+k} - z_k$ is stable.

3.4 Ellipsoidal Calculus

An ellipsoid $\mathcal{E}(c, S) = \left\{x \mid (x-c)^T S^{-1}(x-c) \leq 1\right\}$ describes an affine transformation of the unit ball with center $c \in R^{n_x}$ and positive definite shape matrix $S \in \mathbb{R}^{n_x \times n_x}$. Ellipsoids are preserved under affine transformations since $A\mathcal{E}(c, S) = \mathcal{E}(Ac, ASA^T)$. Although the result of a set addition of two ellipsoids is, in general, not ellipsoidal, we can outer approximate the operation [17]. The over-approximated ellipsoid can be computed through

$$\mathcal{E}(c_1, S_1) \oplus \mathcal{E}(c_2, S_2) \subseteq \mathcal{E}(c, S_+) \tag{4}$$

with shape matrix $S_+ = (1 + \alpha^{-1})S_1 + (1 + \alpha)S_2$ and center $c_+ = c_1 + c_2$ given by $\alpha = \sqrt{\text{Tr}(S_1)/\text{Tr}(S_2)}$. Another useful expression is to check whether the ellipsoid $\mathcal{E}(c, S)$ is contained in the polytope $\mathcal{P} = \{x \mid Hx \leq d\}$, where $Hx \leq d$ describes a system of linear inequalities. The inscription is evaluated by

$$h_j^T c - d_j + \sqrt{h_j^T S h_j} \leq 0 \quad \forall\, j, \tag{5}$$

where h_j is the j-th row of H and d_j is the j-th vector component.

Ellipsoids have favorable geometrical properties compared to polytopes, *e.g.*, under uncertainty the computational complexity is linear over the predictive horizon. Also, the analytical expressions of (4) can be exploited to maintain differentiability along the predicted trajectory.

4 ENSEMBLE MODEL PREDICTIVE SAFETY CERTIFICATION

Our proposed algorithm integrates an ensemble of NNs and tube-based MPC into (2) to certify the actions of a model-based RL agent. Trained on trajectory data from the environment, the ensemble of dynamics models is leveraged for both RL policy optimization and planning with tube-based MPC. The ensemble is represented by probabilistic NNs and parametrizes state-action dependent ellipsoidal predictions, propagated over multiple time steps. An action is certified as safe when the planned tubes satisfy the given state-action constraints and capture the trajectory of the actual system.

In the remainder of this section, we first describe our approach for a single dynamics model in Sections 4.1–4.2 and then extend to ensembles in Section 4.3. We finally present the X-MPSC optimization problem and our algorithm in Sections 4.4–4.5.

4.1 Neural Network Parametrization

To approximate the system dynamics, we use probabilistic NNs

$$f_\phi(x_t, u_t) = \mathcal{N}(m_\phi(x, u), S_\phi(x, u)) \tag{6}$$

parametrized by Gaussian probability distribution functions with the mean $m_\phi(x, u)$ and the diagonal covariance matrix $S_\phi(x, u)$. The predicted uncertainties are state- and action-dependent and are determined by the vector ϕ that holds the flattened weights and biases of the NN. Similar to [10], we train a dynamics model by optimizing the maximum-likelihood

$$\underset{\phi}{\text{minimize}} \quad (m_\phi - x_{t+1})^T S_\phi^{-1}(m_\phi - x_{t+1}) + \log\det S_\phi, \tag{7}$$

over trajectory data sampled from (1). An apparent advantage of using probabilistic over deterministic NNs is that aleatoric uncertainty can be captured, *i.e.*, the stochasticity inherent to a system.

4.2 Single-Model Uncertainty Propagation

To predict the future evolution of (1), we use a probabilistic dynamics model f_ϕ for multi-step look ahead rollouts. The nominal state trajectory described by $(z_0, z_1, \dots, z_N)$ is produced by the action sequence $(v_0, v_1, \dots, v_{N-1})$ with respect to the nonlinear model $z_{k+1} = m_\phi(z_k, v_k)$ for all $k \in [0, N-1]$. In addition to the mean, the probabilistic model provides an ellipsoidal uncertainty estimate S_ϕ that is propagated over multiple time steps. However, the resulting tube can grow large when an open loop action sequence rather than a closed loop feedback is used for planning [28]. Thus, we employ the affine control law from (3) to keep the actual system close to the nominal trajectory.

We now consider the uncertainty tube along the nominal trajectory $(z_0, v_0, \dots, v_{N-1}, z_N)$. For a one-step error prediction, we make use of the first-order Taylor-series expansion

$$x_{t+k+1} \approx m_\phi(z_k, v_k) + A_k(x_{t+k} - z_k) + B_k(u_{t+k} - v_k),$$

around a fixed point (z_k, v_k) given $A_k = \nabla_x\, m_\phi(x, u)^T \,|_{x=z_k, u=v_k}$ and $B_k = \nabla_u\, m_\phi(x, u)^T \,|_{x=z_k, u=v_k}$ as the Jacobians. The predicted error $e_{k+1} = x_{t+k+1} - z_{k+1}$ between the nominal state and the

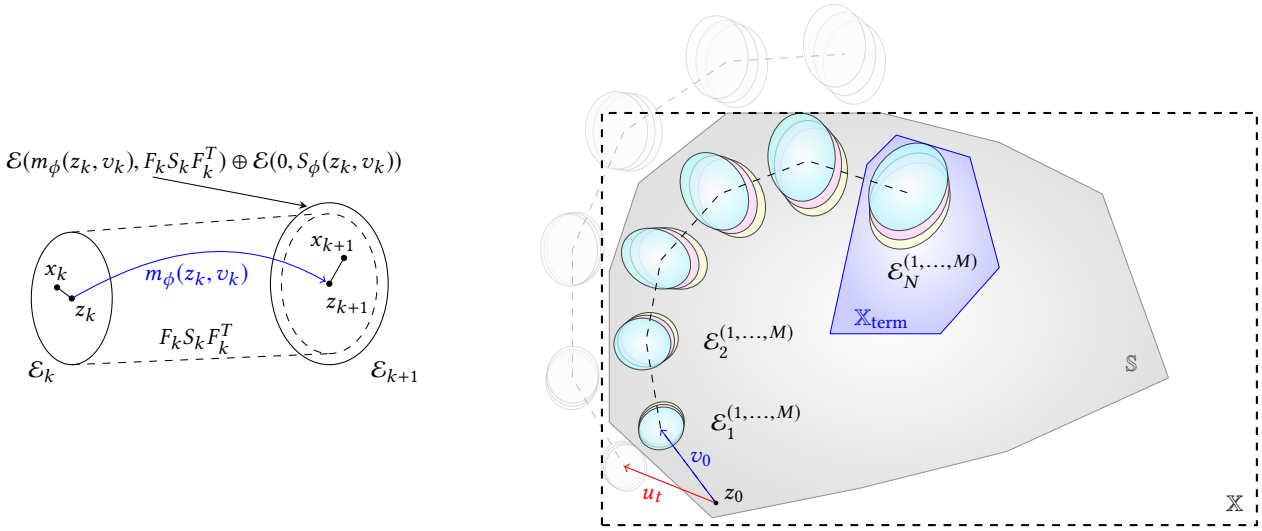


Figure 1: X-MPSC uses multi-step planning with ellipsoidal uncertainty estimates. (Left) Ellipsoidal uncertainty propagation with a single model. (Right) Tube-based predictions generated by an ensemble of NN models. By utilizing multiple models, an unsafe action u_t (red) is corrected to v_0 (blue) that keeps the system within the safety constraints over the horizon N.

actual system state satisfies the error difference equation

$$\begin{aligned} e_{k+1} &\approx A_k(x_{t+k} - z_k) + B_k(u_{t+k} - v_k) \\ &= A_k(x_{t+k} - z_k) + B_k K(x_{t+k} - z_k) \\ &= (A_k + B_k K)(x_{t+k} - z_k) \\ &= F_k e_k \end{aligned}$$

that is accurate to the first order, given $z_{k+1} = m_\phi(z_k, v_k)$. The matrix $F_k = A_k + B_k K$ describes the closed loop error system. To account for state-action dependent uncertainties S_ϕ, we combine the nonlinear nominal trajectory with the linearized error dynamics. A one-step ellipsoidal forward propagation is computed by

$$\mathcal{E}_{k+1} = g_\phi(\mathcal{E}_k, v_k)$$

with the non-linear mapping

$$g_\phi(\mathcal{E}_k, v_k) = \mathcal{E}\left(m_\phi(z_k, v_k), F_k S_k F_k^T\right) \oplus \mathcal{E}\left(0, S_\phi(z_k, v_k)\right),$$

where the evolution of $\mathcal{E}_k$'s only depends on the action sequence $(v_0, \dots, v_{N-1})$ and the NN parameters ϕ. Note that $\mathcal{E}(z_k, S_k)$ is abbreviated to $\mathcal{E}_k$ to improve readability. An illustration of the one-step uncertainty propagation is depicted in Figure 1 (Left).

4.3 Ensemble Uncertainty Propagation

A frequent problem in model-based RL is that NN predictions exhibit inaccuracies that grow with the length of the predictive horizon, limiting the applicability to short rollouts [25]. Ensembles, however, demonstrated their effectiveness in preventing the exploitation of inaccuracies during planning [10]. Therefore, we adopt an NN ensemble $\tilde{f}_\phi = \{f_{\phi_1}, \dots, f_{\phi_M}\}$ composed of M models. The ellipsoidal uncertainty propagation for each model i is described by

$$\mathcal{E}_{k+1}^{(i)} = g_{\phi_i}\left(\mathcal{E}_k^{(i)}, v_k\right),$$

resulting in M tubes used for planning.

4.4 Safety Certification

In this section, we introduce the optimization problem that is solved by X-MPSC. More formally, X-MPSC extends nominal MPSC from (2) with tube-based MPC and a probabilistic ensemble of NNs (the modified parts are highlighted in blue color). The goal of X-MPSC is to certify an action u_t proposed by an RL-based policy in each time step t by solving the optimization problem

$$\begin{aligned} \underset{v_0,\dots,v_{N-1}}{\text{minimize}} \quad & \|u_t - v_0\|_2^2 & \\ \text{subject to} \quad & \mathcal{E}_0^{(i)} = \mathcal{E}(x_t, 0) & \forall\, i \\ & \mathcal{E}_{k+1}^{(i)} = g_{\phi_i}(\mathcal{E}_k^{(i)}, v_k) & \forall\, k \in [0, N-1], i \\ & v_k \in \tilde{\mathbb{U}}(\mathcal{E}_k^{(i)}) & \forall\, k \in [0, N-1], i \\ & \mathcal{E}_k^{(i)} \subseteq \mathbb{X} & \forall\, k \in [0, N], i \\ & \mathcal{E}_N^{(i)} \subseteq \mathbb{X}_{\text{term}} & \forall\, i \end{aligned} \tag{8}$$

over the horizon N. The objective is solved in a receding horizon fashion, where a safe action v_0 that deviates as minimally as possible from the learning input is obtained. For (8) to be feasible, a series of constraints must be satisfied. All propagated ellipsoids must be contained in the polytopic state space, which is checked through (5). Further, every member of the ensemble is subject to the constraints. If a single member violates a constraint, the optimization problem becomes infeasible. The final ellipsoid of each tube is required to be contained in the terminal set $\mathbb{X}_{\text{term}}$. Depending on the ellipsoidal uncertainty estimates, the set of feasible actions shrinks $v_k \in \tilde{\mathbb{U}}(\mathcal{E}_k) = \mathbb{U} \ominus \mathcal{E}(0, K S_k K^T)$.

When (8) is feasible, we do not only obtain a safe action but also a sequence of feedback policies $\Pi_t = (\pi_t, \pi_{t+1}, \dots, \pi_{t+N-1})$ that can steer the system back to $\mathbb{X}_{\text{term}}$ within N steps. Here, each $\pi_{t+k}(x_{t+k}) = v_k + K(x_{t+k} - z_k)$ is an affine controller that tracks the actual system toward the nominal trajectory. In case of infeasibility, the solution of the former solver iteration Π_{t-1} is reused. An

illustration of the optimization problem and the safety certification can be seen in Figure 1 (Right).

4.5 Proposed Algorithm

The solution to the optimization problem in (8) provides a safe action in each step. However, to provide safe exploration for an RL agent throughout the training, we rely on certain assumptions.

Assumption 4.1. There exists a safe backup policy $\pi_b \in \Pi_b$ such that when following π_b the states

$$x_t \in \mathbb{S} \Rightarrow x_{t+k} \in \mathbb{X} \quad \forall k > 0$$

are contained in $\mathbb{X}$. The set $\mathbb{S}$ is called safe set.

The safe set $\mathbb{S}$ is a control-forward invariant set that allows us to gather safely offline data when having access to a safe backup controller. We utilize offline data collected by such a safe backup controller for pre-training the ensemble of dynamics model $\tilde{f}_\phi$ and the initial policy π_θ before starting the RL training. In practice, we can satisfy this assumption through a simple stabilizing local controller, which has low task performance but keeps the system safe within a small region of the state space.

Assumption 4.2. The set of initial states is inscribed in the safe set, *i.e.*, $\mathbb{X}_0 \subseteq \mathbb{S}$. Also, the terminal set is a subset of the safe set, *i.e.*, $\mathbb{X}_{\text{term}} \subseteq \mathbb{S}$. Both sets are convex.

Since all initial states lie in the safe set, a safe backup policy is able to keep the system safe for all future time steps.

Assumption 4.3. The actual system (1) underlies bounded disturbances and is Lipschitz continuous.

The predictions of the dynamics models (6) can be also bounded by this assumption.

Assumption 4.4. The ensemble of dynamics models $\tilde{f}_\phi$ is sufficiently accurate such that it captures the trajectories of the actual system (1).

A trajectory is captured by the ensemble when all states of the actual system are contained in any of the predicted ellipsoids, *i.e.*,

$$\forall\, k \in [0, N] \;\; \exists i \quad \text{s. t.} \quad x_{t+k} \in \mathcal{E}_k^{(i)}.$$

Even though a state x_{t+k} is not contained in any of the predicted ellipsoids but lies in between the tube-based rollouts, *i.e.*,

$$x_{t+k} \in \operatorname{conv}\left(\bigcup_i \mathcal{E}_k^{(i)}\right),$$

safety constraints can be still enforced. We consider the ensemble of dynamics to be sufficiently accurate in such a scenario. The assumption of an accurate dynamics model is very strong since the model can only be a good approximation in regions where data is available. Since the terminal set acts as a natural regularizer to the exploration in (8), *i.e.*, the agent must reach $\mathbb{X}_{\text{term}}$ within N steps, exploration relies on the terminal set growing throughout training to acquire novel samples. However, the growth must happen at an appropriate speed so that new data is informative (*i.e.*, from regions where prediction uncertainty is high) and safe (the system can be steered back to the terminal set).

Algorithm 1 Safe Reinforcement Learning with X-MPSC

Input: Initial data $\mathbb{D}_0$ collected by a safe policy π_b (and optionally a prior model f_{prior})
Pre-train π_θ on $\mathbb{D}_0$ and set $\mathbb{D} \leftarrow \mathbb{D}_0$
for epoch $j = 1, \dots$ **do**
  Train actor-critic and ensemble model $\tilde{f}_\phi$ via (7) on $\mathbb{D}$
  Estimate $\tilde{\mathbb{S}}_j$ based on (9) and set $\mathbb{X}_{\text{term}} \leftarrow \tilde{\mathbb{S}}_{j-\delta}$
  for time step $t = 1, \dots$ **do**
    Sample (possibly unsafe) $u_t \sim \pi_\theta(x_t)$ from RL policy
    Obtain (feasible, Π) by solving *X-MPSC* problem (8)
    Retrieve sequence of controllers

$$\Pi_t \leftarrow \begin{cases} \Pi = (\pi_t, \pi_{t+1}, \dots, \pi_{t+N-1}) & \textbf{if feasible} \\ \Pi_{t-1} = (\pi_t, \dots, \pi_{t+N-2}) & \textbf{otherwise} \end{cases}$$

    Get safe action $v_t = \pi_t(x_t)$ and apply v_t to system (1)
    Store $\mathbb{D} \leftarrow \mathbb{D} \cup (x_t, v_t, x_{t+1}, \text{feasible})$
  end for
end for

We claim that when the Assumptions 4.1–4.4 are fulfilled, then Algorithm 1 becomes itself a safe backup policy due to its recursive feasibility and, hence, can safely certify the actions of an RL-based agent. In the next paragraph, we will give an intuition of this claim and show empirical evidence with the experiments conducted in Section 6. Our method is summarized in Algorithm 1.

At the beginning of each episode, we obtain $x_0 \in \mathbb{X}_0$. By Assumptions 4.1 and 4.2, there always exists a solution $\Pi_0 = \{\pi_b, \dots, \pi_b\}$ at $t = 0$ since $x_0 \in \mathbb{X}_0 \subseteq \mathbb{S} \Rightarrow x_t \in \mathbb{X} \quad \forall t > 0$ when following a safe backup controller $\pi_b \in \Pi_b$. We can now show by induction that the system can be kept safe $\forall t \geq 0$. Let the previous step $t-1$ have the feasible solution $\Pi_{t-1} = (\pi_{t-1}, \pi_t, \pi_{t+1}, \dots)$ that holds the system safe for all future time steps $t \geq 0$. Then, there exists also a solution at step t because either X-MPSC will find it by solving (8) or the solution Π_{t-1} from the former step gives a sequence of policies $(\pi_t, \pi_{t+1}, \dots, \pi_{t+N-2})$ as a fallback solution that leads to a safe state $x_N \in \mathbb{X}_{\text{term}} \subseteq \mathbb{S}$ within $N-1$ steps, from where a safe backup controller π_b can keep the system safe.

5 PRACTICAL IMPLEMENTATION

In the previous section, we did not specify how the terminal set $\mathbb{X}_{\text{term}}$ can be obtained nor how the safe set $\mathbb{S}$ can be determined. In this section, we elaborate on the estimation of both sets and introduce prior system models that we used to improve the accuracy of the NN dynamics models.

Estimation of the Safe Set. The safe set $\mathbb{S}$ is difficult to compute in practice. However, from the data $\mathbb{D}$ collected so far over the training, we can build the outer approximation

$$\tilde{\mathbb{S}} = \operatorname{conv}\{x_t \mid (x_t, \cdot, \cdot, \text{feasible} = \text{true}) \in \mathbb{D}\} \tag{9}$$

as convex hull over all states where X-MPSC found a solution to (8), which is polytopic and convex. Here we use the fact that all states x_t kept the system within $\mathbb{X}$ under the model $\tilde{f}_\phi$. Note that $\tilde{\mathbb{S}}$ is just an approximation of $\mathbb{S}$ and changes as soon as new data samples are collected in the training process.

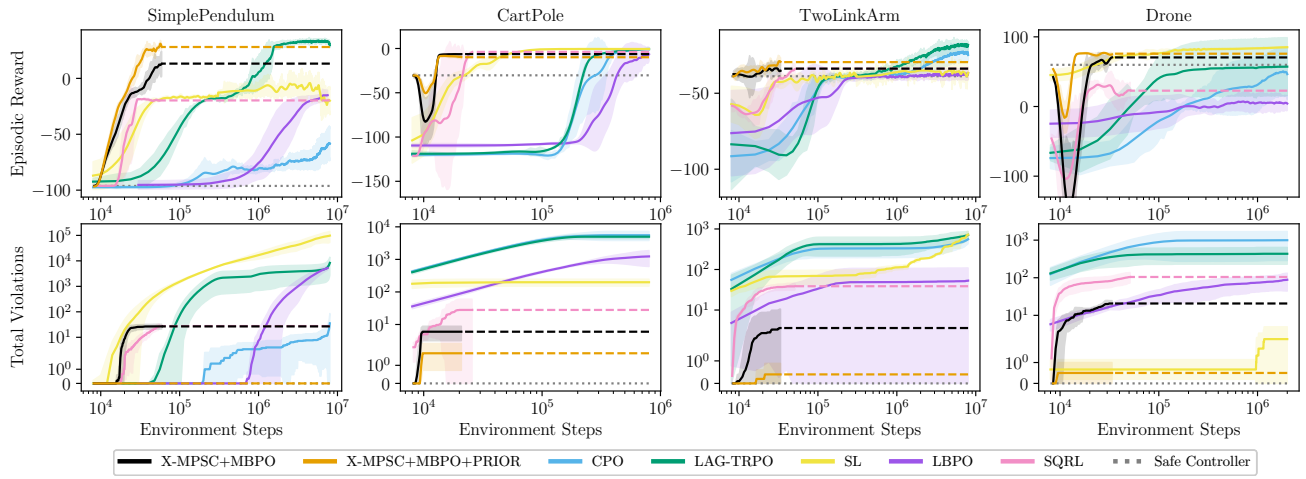


Figure 2: Experimental results. Thick lines show the average over five independent seeds and the shaded area denotes the standard deviation. (Top) The cumulative reward of one episode reported over the total environment steps. (Bottom) Total constraint violations over the whole training.

Terminal Set. Since the terminal set naturally limits exploration, the terminal set is supposed to grow throughout the training to learn about the areas of the state space that have not been visited yet. Due to Assumption 4.2, we can set $\mathbb{X}_{\text{term}} \leftarrow \mathbb{X}_0$ as the first choice for the terminal set. The initial state distribution can be estimated by building the convex hull of all initial states from $\mathbb{D}_0$. As new samples are collected, the safe set $\tilde{\mathbb{S}}_j$ is re-estimated at each epoch j. In order to prevent model exploitation, we use a safe set estimation of a former epoch, *i.e.*, $\mathbb{X}_{\text{term}} \leftarrow \tilde{\mathbb{S}}_{j-\delta}$ such that the estimated safe set is delayed by δ epochs.

Handling of Infeasibility. Due to divergence in the model ensemble predictions or large uncertainty ellipsoids, the X-MPSC problem might not always be feasible, *i.e.*, the solver cannot find an action sequence that keeps all tubes within the constraints. In such cases, the solution Π_{t-1} found in the former solver iteration returns a policy sequence that steers the system back to $\mathbb{X}_{\text{term}}$ in $N-1$ steps. Safety can still be guaranteed due to recursive feasibility. In practice, however, a single failure event can lead to a series of infeasibility events. When the solver fails to find a solution to (8) in N consecutive steps, we transform the hard constraints to soft constraints with high penalty terms. The number of infeasibility events depends on the selected hyper-parameters and varied between 0.0 % (for the best seeds) and 2.0 % (worst case) of the time steps. However, in the TwoLinkArm task, we could also observe a worst-case failure rate of approximately 39 %.

Prior Model. To improve the validity of Assumption 4.4, we tested the use of a prior model. Thus, we extended each ensemble member

$$f_{\phi_i}(x_t, u_t) = \mathcal{N}(m_{\phi_i}(x,u), S_{\phi_i}(x,u)) + f_{\text{prior}}$$

with an additive component f_{prior} that is derived from first principles. We set the system parameters of the prior model with an error of 20% (offset) compared to the actual system's parameters of (1).

6 EXPERIMENTS

Our experimental evaluation is intended to give empirical evidence that X-MPSC can certify the actions taken by an RL agent and that the Assumptions 4.1–4.4 apply to typical RL problem settings. The software implementation is published on GitHub and can be found at: https://github.com/SvenGronauer/x-mpsc.

6.1 Environments

We tested X-MPSC on four tasks that differ in complexity and dynamics. (1) *Simple Pendulum* ($\mathbb{X} \subset \mathbb{R}^2, \mathbb{U} \subset \mathbb{R}$) describes a swing-up task with restricted angle and input constraints. (2) In *Cart Pole* ($\mathbb{X} \subset \mathbb{R}^4, \mathbb{U} \subset \mathbb{R}$), the agent is supposed to balance the pole in an upright position without violating cart position and pole angle constraints. (3) *Two-Link-Arm* ($\mathbb{X} \subset \mathbb{R}^8, \mathbb{U} \subset \mathbb{R}^2$) is a two-joint manipulator where a target point should be reached with end-effector position limits. (4) The *Drone* ($\mathbb{X} \subset \mathbb{R}^{12}, \mathbb{U} \subset \mathbb{R}^4$) environment describes the task to take off the ground and fly to position $[0, 0, 1]^T$ with a restricted body angle. All environments terminated early when state constraint violations occurred. We denote the episodic reward as task performance while we refer to the total number of constraint violations as safety performance.

6.2 Algorithm Setup

In general, X-MPSC can be combined with any RL algorithm. For our experiments, we chose model-based policy optimization (MBPO) proposed by Janner et al. [15] for two reasons. First, we can use the NN ensemble for both planning with X-MPSC and for generating short model-based rollouts to train the policy. Second, MBPO has been shown to be more sample efficient than other RL algorithms, which reduces the wall clock time since the computational bottleneck is finding a solution to (8). For training, we used an ensemble size of $M = 5$, where each NN was implemented as a multi-layer

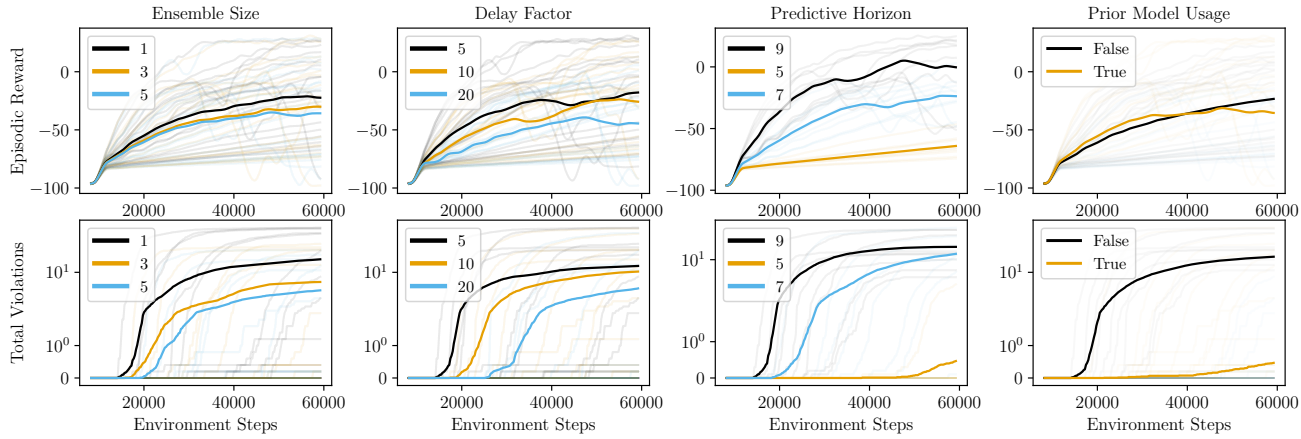


Figure 3: Impact of X-MPSC hyper-parameters on safety and performance in Simple Pendulum.

perceptron (MLP) with two hidden layers. The actor-critics were also MLPs with two hidden layers.

The optimization problem in (8) was solved with a primal-dual interior point method, for which we used *CasADi* and the *IPOPT* software library. Two relevant hyper-parameters of X-MPSC are the delay factor δ and the horizon N, which depend on the characteristics of the environment. We adjusted both hyper-parameters for each environment individually. The feedback matrix K was hand-tuned, and all matrix entries were set to 0.5 for all environments. We expect that tuning this hyper-parameter individually for each environment will reduce the number of constraint violations but did not test this. An overview of all selected hyper-parameters can be found in the Appendix at: https://arxiv.org/abs/2402.04182.

We collected for each task $|\mathbb{D}_0| = 8000$ initial samples, which equals less than three minutes of real-world experience on systems with a 50 Hz sampling rate. The safe backup controller used for initial data collection is specific to the environment dynamics and was implemented by a low-performing but stabilizing linear quadratic regulator (LQR) or proportional–integral–derivative (PID) controller. We compare our results with several baselines, namely constrained policy optimization (CPO) [1], safety Q-functions for RL (SQRL) [33], safety layer (SL) [11], Lyapunov barrier policy optimization (LBPO) [31], and Lagrangian trust-region policy optimization (Lag-TRPO). For SL and SQRL, we collected initial data samples that deliberately contained safety violations to pre-train their safety-aware functions.

6.3 Results

Figure 2 depicts the episodic reward as well as the total number of constraint violations over the course of training for each algorithm and environment. For each algorithm, we identified the best hyper-parameter choice via a coarse grid search and report only the best configuration. Each experiment was averaged over five independent random seeds. Additionally, we report the performance of the safe controller that was used to collect the initial data for X-MPSC. Note that MBPO and SQRL converge faster due to their off-policy nature.

We observe that, even without using a prior model, X-MPSC can significantly reduce constraint violations compared to the other algorithms while achieving only a slightly worse final reward than Lag-TRPO, the strongest baseline in terms of task performance. The on-policy algorithms CPO, SL, and LBPO show less consistent results across the experiments, with SL demonstrating comparable performance to X-MPSC only in the Drone task. The off-policy SQRL displays a good performance-safety ratio by learning policies with fewer violations than the other algorithms in most cases, excluding X-MPSC. However, when an inaccurate prior model is added to the NN ensemble, the total constraint violations with the X-MPSC can be reduced by approximately an order of magnitude without performance losses in terms of episodic reward.

6.4 Impact of Hyper-Parameters

We studied the impact of selected X-MPSC hyper-parameters on performance and constraint violations in the Simple Pendulum task. Since hyper-parameter settings can cross-correlate and influence each other, we deliberately do not fix all hyper-parameters except the one of interest to avoid cherry-picking good hyper-parameter settings and distorting the general impact of the selected hyper-parameter. Instead, we are interested in the average effect of the hyper-parameter choice and, thus, measure the impact over multiple random seeds and various hyper-parameter settings. To this end, we tested 54 different hyper-parameter configurations, where each configuration was averaged over five independent trials.

Figure 3 shows each configuration in pale color, while the average of the selected hyper-parameter is shown as a solid line. The plots indicate that a larger ensemble size results in fewer total constraint violations. Further, the usage of a coarse prior model can reduce the number of constraint violations on average by one order of magnitude. The delay factor regulates the speed of the terminal set updates, with a higher factor diminishing the number of violations at the expense of fewer cumulative rewards. Finally, a longer predictive horizon N can accelerate the learning progress but produces more costs.

7 DISCUSSION

The results show that our proposed X-MPSC algorithm is able to provide safe exploration when certain assumptions are fulfilled. We discuss our experimental results based on those assumptions first and then give reasons for the success or failure of X-MPSC. After that, we discuss the most critical limitations of our method.

7.1 Discussion of Results

Safety. As shown in Figure 2, our method offers a better safety operation in terms of constraint satisfaction, with the violations kept at zero in the Simple Pendulum and nearly zero in the Drone environment when prior knowledge is used. Note that the Cart Pole task is particularly challenging since the starting pole upward position is an unstable equilibrium point, and expanding the safe set will rapidly reach states where the controller is not yet able to stabilize it. X-MPSC was able to learn a safe policy faster, but collecting the data necessary to approximate the dynamics model around unstable regions is a challenge that remains unsolved.

Performance. A safety-performance trade-off is reflected in the results. Slowly expanding the safe set and using ensembles based on tube-based MPC, which ensures that all tubes are contained in the safety constraints, result in a conservative system. X-MPSC's performance without prior model is relatively close to the best baseline algorithms in the Cart Pole and Drone environments but presents a more significant difference to Lag-TRPO in the other two environments. When adding a prior dynamics model, we do not only see a significant decrease in constraint violations, but also slight improvements in terms of task reward in the Drone and Two-Link-Arm environments. In the Simple Pendulum environment this increase was even more substantial. We argue that the usage of a prior model leads to more accurate nominal trajectories, which facilitates learning in terms of reward performance as well as improved safety satisfaction capabilities.

7.2 Limitations of Our Method

In the remainder of this section, we list the limitations of the X-MPSC algorithm, which are ordered from weak to strong. We see these limitations as a starting point to be addressed in future work.

Conservatism. With a larger ensemble size, the safety certification can lead to more conservative behavior since the constraints imposed by every single dynamics model must be satisfied. As soon as a single model deviates from the rest of the ensemble, the agent is enforced to satisfy wrong constraints, and, hence, the set of feasible actions is reduced, which can result in diminished reward performance. With an increasing number of models, the safety certification becomes safer but also more conservative.

Safe Backup Controller. Our algorithm only requires offline data to be collected within the safe set to pre-train the ensemble and estimate the initial safe and terminal sets. In contrast to our work, related methods require offline data to contain mixed safe and unsafe trajectories (*e.g.*, SL [11], SQRL [33] and Recovery RL [35]), which can be a limiting factor for the deployment on real-world robots. In practice, the safe backup controller can be implemented as a local control law that has low task performance but keeps the system close to the initial state and within the safe set. The time required to develop a safe backup controller largely depends on the task and the robot dynamics. For CartPole and Simple Pendulum, we used an LQR, while we used P-controllers for Drone and Two-Link-Arm. Because a safe backup controller does not aim for good task performance but only for stabilizing within a small region of the state space, the design can be achieved with a few trial-and-error attempts. Conversely, the design of a safe backup controller with high reward performance can take considerably more time and requires accurate prior knowledge about the robot system. X-MPSC can be used, however, with relatively little prior knowledge about the system and can thus offer an approach to safely learn about the system while being able to improve task performance.

Accurate Dynamics Model. The access to an accurate model is a strong assumption since the model is only a good approximation in regions where data samples are available. Through the terminal set constraint, we naturally limit the exploration of the state space since the terminal set must be reached within N steps and, hence, force the agent to stay close to regions where data exists. Incorporating prior models helped significantly reduce the number of constraint violations, although we only used inaccurate models with parameters deviating by 20% from the ground-truth.

Computation Time. The bottleneck in the training loop is finding a solution to (8). The X-MPSC problem has cubic computational complexity, *i.e.*, $O\left(N^3(n_x + n_u + n_c)^3\right)$ with N being the predictive horizon and n_x, n_u, n_c being the dimensions of the state, action spaces and number of constraints, respectively. Note that the computational complexity grows linearly with the ensemble size M and quadratically with the number of neurons in each layer. Thus, we use at most 20 neurons in the hidden layers of the NNs. Furthermore, the estimation of the safe set involves the computation of the convex hull, which also grows with the number of state constraints. Thus, we limited the number of constrained state space variables to two, *e.g.*, position and velocity.

8 CONCLUSION

We proposed X-MPSC, a novel algorithm that integrates an ensemble of NNs and tube-based MPC into nominal MPSC to correct the actions taken by an RL agent. To provide safe exploration, we utilized an ensemble of probabilistic NNs trained on sampled environment data to plan multiple tube-based trajectories that satisfy *a priori* defined safety constraints. The experimental results demonstrate that our method can achieve significantly fewer constraint violations than comparable RL methods, requiring only offline data with safe trajectories. When an inaccurate prior dynamics model is added to the NN ensemble, the constraint violations can be reduced by an order of magnitude without forfeiting reward performance.

Although the results indicate an improvement over other state-of-the-art algorithms, the scalability to higher dimensional state-action spaces and larger NN models remains a research frontier. Future work may improve upon the computational speed of solving the X-MPSC problem such that it can be implemented on a real robot or applied to high-dimensional tasks, as presented in the *Safety Gym* [29] or *Bullet Safety Gym* [14].

REFERENCES

[1] Joshua Achiam, David Held, Aviv Tamar, and Pieter Abbeel. 2017. Constrained Policy Optimization. In *Proceedings of the 34th International Conference on Machine Learning (ICML)*, Doina Precup and Yee Whye Teh (Eds.). 22–31. http://proceedings.mlr.press/v70/achiam17a.html

[2] Eitan Altman. 1999. *Constrained Markov decision processes*. CRC Press.

[3] Aaron D Ames, Samuel Coogan, Magnus Egerstedt, Gennaro Notomista, Koushil Sreenath, and Paulo Tabuada. 2019. Control barrier functions: Theory and applications. In *European Control Conference (ECC)*. 3420–3431.

[4] Dario Amodei, Chris Olah, Jacob Steinhardt, Paul F. Christiano, John Schulman, and Dan Mané. 2016. Concrete Problems in AI Safety. Preprint. https://doi.org/10.48550/arXiv.1606.06565

[5] Felix Berkenkamp, Matteo Turchetta, Angela Schoellig, and Andreas Krause. 2017. Safe Model-based Reinforcement Learning with Stability Guarantees. In *Advances in Neural Information Processing Systems 30 (NeurIPS)*. 908–918.

[6] Lukas Brunke, Melissa Greeff, Adam W. Hall, Zhaocong Yuan, Siqi Zhou, Jacopo Panerati, and Angela P. Schoellig. 2022. Safe Learning in Robotics: From Learning-Based Control to Safe Reinforcement Learning. *Annual Review of Control, Robotics, and Autonomous Systems* 5, 1 (2022), 411–444. https://doi.org/10.1146/annurev-control-042920-020211 arXiv:https://doi.org/10.1146/annurev-control-042920-020211

[7] Richard Cheng, Gábor Orosz, Richard M. Murray, and Joel W. Burdick. 2019. End-to-End Safe Reinforcement Learning through Barrier Functions for Safety-Critical Continuous Control Tasks. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI, Vol. 33)*. 3387–3395. https://doi.org/10.1609/aaai.v33i01.33013387

[8] Yinlam Chow, Mohammad Ghavamzadeh, Lucas Janson, and Marco Pavone. 2017. Risk-constrained reinforcement learning with percentile risk criteria. *The Journal of Machine Learning Research* 18, 1 (2017), 6070–6120.

[9] Yinlam Chow, Ofir Nachum, Aleksandra Faust, Edgar Duenez-Guzman, and Mohammad Ghavamzadeh. 2019. Lyapunov-based safe policy optimization for continuous control. *arXiv preprint arXiv:1901.10031* (2019).

[10] Kurtland Chua, Roberto Calandra, Rowan McAllister, and Sergey Levine. 2018. Deep Reinforcement Learning in a Handful of Trials Using Probabilistic Dynamics Models. In *Advances in Neural Information Processing Systems 31 (NeurIPS)*. 4759–4770.

[11] Gal Dalal, Krishnamurthy Dvijotham, Matej Vecerik, Todd Hester, Cosmin Paduraru, and Yuval Tassa. 2018. Safe Exploration in Continuous Action Spaces. Preprint. https://doi.org/10.48550/arXiv.1801.08757 arXiv:1801.08757

[12] Gabriel Dulac-Arnold, Nir Levine, Daniel J. Mankowitz, Jerry Li, Cosmin Paduraru, Sven Gowal, and Todd Hester. 2021. Challenges of real-world reinforcement learning: definitions, benchmarks and analysis. *Machine Learning* 110, 9 (2021), 2419–2468. https://doi.org/10.1007/s10994-021-05961-4

[13] Javier García, Fern, and o Fernández. 2015. A Comprehensive Survey on Safe Reinforcement Learning. *Journal of Machine Learning Research* 16, 42 (2015), 1437–1480.

[14] Sven Gronauer. 2022. *Bullet-Safety-Gym: A Framework for Constrained Reinforcement Learning*. Technical Report. mediaTUM. https://doi.org/10.14459/2022md1639974

[15] Michael Janner, Justin Fu, Marvin Zhang, and Sergey Levine. 2019. When to Trust Your Model: Model-Based Policy Optimization. In *Advances in Neural Information Processing Systems 32 (NeurIPS)*, H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett (Eds.). 12519–12530.

[16] Torsten Koller, Felix Berkenkamp, Matteo Turchetta, and Andreas Krause. 2018. Learning-Based Model Predictive Control for Safe Exploration. In *IEEE Conference on Decision and Control (CDC)*. 6059–6066. https://doi.org/10.1109/CDC.2018.8619572

[17] Alex A. Kurzhanskiy and Pravin Varaiya. 2006. Ellipsoidal Toolbox (ET). In *IEEE Conference on Decision and Control (CDC, Vol. 45)*. 1498–1503. https://doi.org/10.1109/CDC.2006.377036

[18] Jan Leike, Miljan Martic, Victoria Krakovna, Pedro A. Ortega, Tom Everitt, Andrew Lefrancq, Laurent Orseau, and Shane Legg. 2017. AI Safety Gridworlds. Preprint. https://doi.org/10.48550/arXiv.1711.09883 arXiv:1711.09883

[19] Weiwei Li and Emanuel Todorov. 2004. Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems. In *International Conference on Informatics in Control, Automation and Robotics*.

[20] Qingkai Liang, Fanyu Que, and Eytan Modiano. 2018. Accelerated primal-dual policy optimization for safe reinforcement learning. *arXiv:1802.06480* (2018).

[21] Yuping Luo and Tengyu Ma. 2021. Learning Barrier Certificates: Towards Safe Reinforcement Learning with Zero Training-time Violations. In *Advances in Neural Information Processing Systems 34 (NeurIPS)*, M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (Eds.). 25621–25632. https://proceedings.neurips.cc/paper_files/paper/2021/file/d71fa38b648d86602d14ac610f2e6194-Paper.pdf

[22] Björn Lütjens, Michael Everett, and Jonathan P. How. 2019. Safe Reinforcement Learning With Model Uncertainty Estimates. In *International Conference on Robotics and Automation (ICRA)*. 8662–8668. https://doi.org/10.1109/ICRA.2019.8793611

[23] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529 EP. https://doi.org/10.1038/nature14236

[24] Teodor Mihai Moldovan and Pieter Abbeel. 2012. Safe Exploration in Markov Decision Processes. In *Proceedings of the 29th International Coference on Machine Learning (ICML)*. 1451–1458.

[25] Anusha Nagabandi, Gregory Kahn, Ronald S. Fearing, and Sergey Levine. 2018. Neural Network Dynamics for Model-Based Deep Reinforcement Learning with Model-Free Fine-Tuning. In *IEEE International Conference on Robotics and Automation (ICRA)*. 7559–7566. https://doi.org/10.1109/ICRA.2018.8463189

[26] Martin Pecka and Tomas Svoboda. 2014. Safe Exploration Techniques for Reinforcement Learning – An Overview. In *Modelling and Simulation for Autonomous Systems (MESAS)*, Jan Hodicky (Ed.). Springer International Publishing, Cham, 357–375.

[27] Samuel Pfrommer, Tanmay Gautam, Alec Zhou, and Somayeh Sojoudi. 2022. Safe Reinforcement Learning with Chance-constrained Model Predictive Control. In *Proceedings of The 4th Annual Learning for Dynamics and Control Conference*, Vol. 168. PMLR, 291–303.

[28] James Blake Rawlings, David Q Mayne, and Moritz Diehl. 2017. *Model Predictive Control: Theory, Computation, and Design* (2nd edition ed.). Nob Hill Publishing, LLC.

[29] Alex Ray, Joshua Achiam, and Dario Amodei. 2019. Benchmarking Safe Exploration in Deep Reinforcement Learning. Preprint. https://cdn.openai.com/safexp-short.pdf

[30] Alexander Robey, Haimin Hu, Lars Lindemann, Hanwen Zhang, Dimos V Dimarogonas, Stephen Tu, and Nikolai Matni. 2020. Learning control barrier functions from expert demonstrations. In *2020 59th IEEE Conference on Decision and Control (CDC)*. IEEE, 3717–3724.

[31] Harshit Sikchi, Wenxuan Zhou, and David Held. 2021. Lyapunov Barrier Policy Optimization. *CoRR* abs/2103.09230 (2021). arXiv:2103.09230

[32] David Silver, Aja Huang, Chris J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel, and Demis Hassabis. 2016. Mastering the game of Go with deep neural networks and tree search. *Nature* 529, 7587 (2016), 484–489. https://doi.org/10.1038/nature16961

[33] Krishnan Srinivasan, Benjamin Eysenbach, Sehoon Ha, Jie Tan, and Chelsea Finn. 2020. Learning to be safe: Deep rl with a safety critic. Preprint. https://doi.org/10.48550/arXiv.2010.14603

[34] Chen Tessler, Daniel J. Mankowitz, and Shie Mannor. 2019. Reward Constrained Policy Optimization. In *7th International Conference on Learning Representations (ICLR)*.

[35] Brijen Thananjeyan, Ashwin Balakrishna, Suraj Nair, Michael Luo, Krishnan Srinivasan, Minho Hwang, Joseph E. Gonzalez, Julian Ibarz, Chelsea Finn, and Ken Goldberg. 2021. Recovery RL: Safe Reinforcement Learning With Learned Recovery Zones. *IEEE Robotics and Automation Letters* 6, 3 (2021), 4915–4922. https://doi.org/10.1109/LRA.2021.3070252

[36] Kim P. Wabersich, Lukas Hewing, Andrea Carron, and Melanie N. Zeilinger. 2022. Probabilistic Model Predictive Safety Certification for Learning-Based Control. *IEEE Trans. Automat. Control* 67, 1 (2022), 176–188. https://doi.org/10.1109/TAC.2021.3049335

[37] K. P. Wabersich and M. N. Zeilinger. 2018. Linear Model Predictive Safety Certification for Learning-Based Control. In *IEEE Conference on Decision and Control (CDC)*. 7130–7135. https://doi.org/10.1109/CDC.2018.8619829

[38] Kim Peter Wabersich and Melanie N. Zeilinger. 2021. A predictive safety filter for learning-based control of constrained nonlinear dynamical systems. *Automatica* 129 (2021), 109597. https://doi.org/doi.org/10.1016/j.automatica.2021.109597

[39] Tsung-Yen Yang, Justinian Rosca, Karthik Narasimhan, and Peter J. Ramadge. 2020. Projection-Based Constrained Policy Optimization. In *International Conference on Learning Representations*.

Reinforcement Learning with Ensemble Model Predictive Safety Certification

Author: Sven Gronauer et al.
Publication: Proc. of the 23rd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2024)
Publisher: International Foundation for Autonomous Agents and Multiagent Systems
Date: May 06, 2024

Reprint Permission

Part B

Multi-Agent Reinforcement Learning

Paper V

Multi-Agent Deep Reinforcement Learning: A Survey

Sven Gronauer and Klaus Diepold

Abstract

The advances in reinforcement learning have recorded sublime success in various domains. Although the multi-agent domain has been overshadowed by its single-agent counterpart during this progress, multi-agent reinforcement learning gains rapid traction, and the latest accomplishments address problems with real-world complexity. This article provides an overview of the current developments in the field of multi-agent deep reinforcement learning. We focus primarily on literature from recent years that combines deep reinforcement learning methods with a multi-agent scenario. To survey the works that constitute the contemporary landscape, the contents are divided into three parts. First, we analyze the structure of training schemes that are applied to train multiple agents. Second, we consider the emergent patterns of agent behavior in cooperative, competitive and mixed scenarios. Third, we systematically enumerate challenges that exclusively arise in the multi-agent domain and review methods that are leveraged to cope with these challenges. To conclude this survey, we discuss advances, identify trends, and outline possible directions for future work in this research area.

S. Gronauer and K. Diepold. "Multi-Agent Deep Reinforcement Learning: A Survey". In: *Artificial Intelligence Review* 55.2 (2022), pp. 895–943. DOI: 10.1007/s10462-021-09996-w

Artificial Intelligence Review (2022) 55:895–943
https://doi.org/10.1007/s10462-021-09996-w

Multi-agent deep reinforcement learning: a survey

Sven Gronauer[1] · **Klaus Diepold**[1]

Published online: 15 April 2021

Abstract
The advances in reinforcement learning have recorded sublime success in various domains. Although the multi-agent domain has been overshadowed by its single-agent counterpart during this progress, multi-agent reinforcement learning gains rapid traction, and the latest accomplishments address problems with real-world complexity. This article provides an overview of the current developments in the field of multi-agent deep reinforcement learning. We focus primarily on literature from recent years that combines deep reinforcement learning methods with a multi-agent scenario. To survey the works that constitute the contemporary landscape, the main contents are divided into three parts. First, we analyze the structure of training schemes that are applied to train multiple agents. Second, we consider the emergent patterns of agent behavior in cooperative, competitive and mixed scenarios. Third, we systematically enumerate challenges that exclusively arise in the multi-agent domain and review methods that are leveraged to cope with these challenges. To conclude this survey, we discuss advances, identify trends, and outline possible directions for future work in this research area.

Keywords Multi-agent systems · Multi-agent learning · Machine learning · Reinforcement learning · Deep learning · Survey

1 Introduction

A multi-agent system describes multiple distributed entities—so-called agents—which take decisions autonomously and interact within a shared environment (Weiss 1999). Each agent seeks to accomplish an assigned goal for which a broad set of skills might be required to build intelligent behavior. Depending on the task, an intricate interplay between agents can occur such that agents start to collaborate or act competitively to excel opponents. Specifying intelligent behavior a-priori through programming is a tough, if not impossible, task for complex systems. Therefore, agents require the ability to adapt and learn over time

✉ Sven Gronauer
sven.gronauer@tum.de

Klaus Diepold
kldi@tum.de

[1] Department of Electrical and Computer Engineering, Technical University of Munich (TUM), Arcisstr. 21, 80333 Munich, Germany

by themselves. The most common framework to address learning in an interactive environment is reinforcement learning (RL), which describes the change of behavior through a trial-and-error approach.

The field of reinforcement learning is currently thriving. Since the breakthrough of deep learning methods, works have been successful at mastering complex control tasks, e.g. in robotics (Levine et al. 2016; Lillicrap et al. 2016) and game playing (Mnih et al. 2015; Silver et al. 2016). The key to these results is based on learning techniques that employ neural networks as function approximators (Arulkumaran et al. 2017). Despite these achievements, the majority of works investigated single-agent settings only, although many real-world applications naturally comprise multiple decision-makers that interact at the same time. The areas of application encompass the coordination of distributed systems (Cao et al. 2013; Wang et al. 2016b) such as autonomous vehicles (Shalev-Shwartz et al. 2016) and multi-robot control (Matignon et al. 2012a), the networking of communication packages (Luong et al. 2019), or the trading on financial markets (Lux and Marchesi 1999). In these systems, each agent discovers a strategy alongside other entities in a common environment and adapts its policy in response to the behavioral changes of others. Carried by the advances of single-agent deep RL, the multi-agent reinforcement learning (MARL) community has been surged with new interest and a plethora of literature has emerged lately (Hernandez-Leal et al. 2019; Nguyen et al. 2020). The use of deep learning methods enabled the community to exceed the historically investigated tabular problems to challenging problems with real-world complexity (Baker et al. 2020; Berner et al. 2019; Jaderberg et al. 2019; Vinyals et al. 2019).

In this paper, we provide an extensive review of the recent advances in the area of multi-agent deep reinforcement learning (MADRL). Although multi-agent systems enjoy a rich history (Busoniu et al. 2008; Shoham et al. 2003; Stone and Veloso 2000; Tuyls and Weiss 2012), this survey aims to shed light on the contemporary landscape of the literature in MADRL.

1.1 Related work

The intersection of multi-agent systems and reinforcement learning holds a long record of active research. As one of the first surveys in the field, Stone and Veloso (2000) analyzed multi-agent systems from a machine learning perspective and classified the reviewed literature according to heterogeneous and homogeneous agent structures as well as communication skills. The authors discussed issues associated with each classification. Shoham et al. (2003) criticized the ill-posed problem statement of MARL which is in the authors' opinion unclear and called for more grounded research. They proposed a coherent research agenda which includes four directions for future research. Yang and Gu (2004) reviewed algorithms and pointed out that the main difficulty lies in the generalization to continuous action and state spaces and in the scaling to many agents. Similarly, Busoniu et al. (2008) presented selected algorithms and discussed benefits as well as challenges of MARL. Benefits include computational speed-ups and the possibility of experience sharing between agents. In contrast, drawbacks are the specification of meaningful goals, the non-stationarity of the environment, and the need for coherent coordination in cooperative games. In addition to that, they posed challenges such as the exponential increase of computational complexity with the number of agents and the alter-exploration problem where agents must gauge between the acquisition of new knowledge and the exploitation of current knowledge. More specifically, Matignon et al. (2012b) identified challenges for the coordination

of independent learners that arise in fully cooperative Markov Games such as non-stationarity, stochasticity, and shadowed equilibria. Further, they analyzed conditions under which algorithms can address such coordination issues. Another work by Tuyls and Weiss (2012) accounted for the historical developments of MARL and evoked non-technical challenges. They criticized that the intersection of RL techniques and game theory dominates multi-agent learning, which may render the scope of the field too narrow and investigations are limited to simplistic problems such as grid worlds. They claimed that the scalability to high numbers of agents and large and continuous spaces are the holy grail of this research domain.

Since the advent of deep learning methods and the breakthrough of deep RL, the field of MARL has attained new interest and a plethora of literature has emerged during the last years. Nguyen et al. (2020) presented five technical challenges including nonstationarity, partial observability, continuous spaces, training schemes, and transfer learning. They discussed possible solution approaches alongside their practical applications. Hernandez-Leal et al. (2019) concentrated on four categories including the analysis of emergent behaviors, learning communication, learning cooperation, and agent modeling. Further survey literature focuses on one particular sub-field of MADRL. Oroojlooyjadid and Hajinezhad (2019) reviewed recent works in the cooperative setting while Da Silva and Costa (2019) and Da Silva et al. (2019) focused on knowledge reuse. Lazaridou and Baroni (2020) reviewed the emergence of language and connected two perspectives, which comprise the conditions under which language evolves in communities and the ability to solve problems through dynamic communication. Based on theoretical analysis, Zhang et al. (2019) focused on MARL algorithms and presented challenges from a mathematical perspective.

1.2 Contribution and survey structure

The contribution of this paper is to present a comprehensive survey of the recent research directions pursued in the field of MADRL. We depict a holistic overview of current challenges that arise exclusively in the multi-agent domain of deep RL and discuss state-of-the-art solutions that were proposed to address these challenges. In contrast to the surveys of Hernandez-Leal et al. (2019) and Nguyen et al. (2020), which focus on a subset of topics, we aim to provide a widened and more comprehensive overview of the current investigations conducted in the field of MADRL while recapitulating what has already been accomplished. We identify contemporary challenges and discuss literature that addresses such. We see our work complementary to the theoretical survey of Zhang et al. (2019).

We dedicate this paper to an audience who wants an excursion to the realm of MADRL. Readers shall gain insights about the historical roots of this still young field and its current developments, but also understand the open problems to be faced by future research. The contents of this paper are organized as follows. We begin with a formal introduction to both single-agent and multi-agent RL and reveal pathologies that are present in MARL in Sect. 2. We then continue with the main contents, which are categorized according to the three-fold taxonomy as illustrated in Fig. 1.

We analyze training architectures in Sect. 3, where we categorize approaches according to a centralized or distributed training paradigm and additionally differentiate into execution schemes. Thereafter, we review literature that investigates emergent patterns of agent behavior in Sect. 4. We classify works in terms of the reward structure (Sect. 4.1), the language between multiple agents (Sect. 4.2), and the social context (Sect. 4.3). In Sect. 5, we enumerate current challenges of the multi-agent domain,

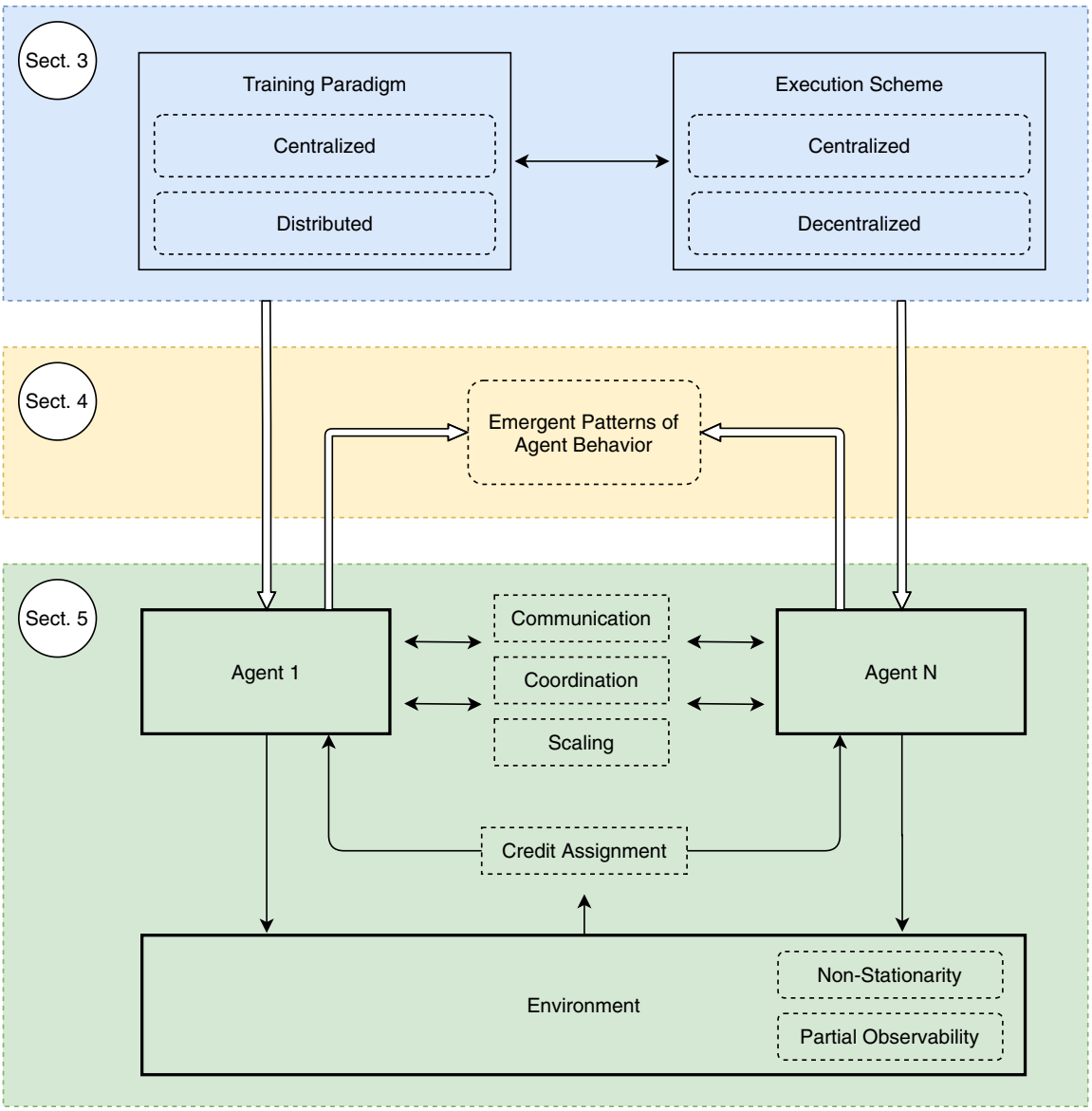


Fig. 1 Schematic structure of the main contents in this survey. In Sect. 3, we review schemes that are applied to train agent behavior in the multi-agent setting. The training of agents can be divided into two paradigms which are namely distributed (Sect. 3.1) and centralized (Sect. 3.2). In Sect. 4, we consider the emergent patterns of agent behavior with respect to the reward structure (Sect. 4.1), the language (Sect. 4.2) and the social context (Sect. 4.3). In Sect. 5, we enumerate current challenges of MADRL which include the non-stationarity of the environment due to co-adapting agents (Sect. 5.1), the learning of communication (Sect. 5.2), the need for a coherent coordination of actions (Sect. 5.3), the credit assignment problem (Sect. 5.4), the ability to scale to an arbitrary number of decision-makers (Sect. 5.5), and non-Markovian environments due to partial observations (Sect. 5.6)

which include the non-stationarity of the environment due to simultaneously adapting learners (Sect. 5.1), the learning of meaningful communication protocols in cooperative tasks (Sect. 5.2), the need for coherent coordination of agent actions (Sect. 5.3), the credit assignment problem (Sect. 5.4), the ability to scale to an arbitrary number of decision-makers (Sect. 5.5), and non-Markovian environments due to partial observations (Sect. 5.6). We discuss the matter of MADRL, pose trends that we identified in recent literature, and outline possible future work in Sect. 6. Finally, this survey concludes in Sect. 7.

2 Background

In this section, we provide a formal introduction into the concepts of RL. We start with the Markov decision process as a framework for single-agent learning in Sect. 2.1. We continue with the multi-agent case and introduce the Markov Game in Sect. 2.2. Finally, we pose pathologies that arise in the multi-agent domain such as the non-stationarity of the environment from the perspective of a single learner, relative over-generalization, and the credit assignment problem in Sect. 2.3. We provide the formal concepts behind these MARL pathologies in order to drive our discussion about the state-of-the-art approaches in Sect. 5. The scope of this background section is deliberately focusing on classical MARL works to reveal the roots of the domain and to give the reader insights into the early works on which modern MADRL approaches rest.

2.1 Single-agent reinforcement learning

The traditional reinforcement learning problem (Sutton and Barto 1998) is concerned with learning a control policy that optimizes a numerical performance by making decisions in stages. The decision-maker called agent interacts with an environment of unknown dynamics in a trial-and-error fashion and occasionally receives feedback upon which the agent wants to improve. The standard formulation for such sequential decision-making is the Markov decision process, which is defined as follows (Bellman 1957; Bertsekas 2012, 2017; Kaelbling et al. 1996).

Definition 1 *Markov decision process (MDP)* A Markov decision process is formalized by the tuple $(\mathscr{X}, \mathscr{U}, \mathscr{P}, R, \gamma)$ where $\mathscr{X}$ and $\mathscr{U}$ are the state and action space, respectively, $\mathscr{P} : \mathscr{X} \times \mathscr{U} \to P(\mathscr{X})$ is the transition function describing the probability of a state transition, $R : \mathscr{X} \times \mathscr{U} \times \mathscr{X} \to \mathbb{R}$ is the reward function providing an immediate feedback to the agent, and $\gamma \in [0, 1)$ describes the discount factor.

The agent's goal is to act in such a way as to maximize the expected performance on a long-term perspective with regard to an unknown transition function $\mathscr{P}$. Therefore, the agent learns a behavior policy $\pi : \mathscr{X} \to P(\mathscr{U})$ that optimizes the expected performance J throughout learning. The performance is defined as the expected value of discounted rewards

$$J = \mathbb{E}_{x_0 \sim \rho_0,\, x_{t+1} \sim \mathscr{P},\, u_t \sim \pi} \left[\sum_{t=0}^{\infty} \gamma^t R(x_t, u_t, x_{t+1}) \right] \tag{1}$$

over the initial state distribution ρ_0 while selected actions are governed by the policy π. Here, we regard the infinite-horizon problem where the interaction between agent and environment does not terminate after a countable number of steps. Note that the learning objective can also be formalized for finite-horizon problems (Bertsekas 2012, 2017). As an alternative to the policy performance, which describes the expected performance as a function of the policy, one can define the utility of being in a particular state in terms of a *value function*. The state-value function $V_\pi : \mathscr{X} \to \mathbb{R}$ describes the utility under policy π when starting from state x, i.e.

$$V_\pi(x) = \mathbb{E}_{x_{t+1}\sim\mathscr{P},\, u_t\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R(x_t, u_t, x_{t+1}) \mid x_0 = x\right]. \tag{2}$$

In a similar manner, the action-value function $Q_\pi : \mathscr{X}\times\mathscr{U}\to\mathbb{R}$ describes the utility of being in state x, performing action u, and following the policy π thereafter, that is

$$Q_\pi(x,u) = \mathbb{E}_{x_{t+1}\sim\mathscr{P},\, u_{t>0}\sim\pi}\left[\sum_{t=0}^{\infty}\gamma^t R(x_t, u_t, x_{t+1}) \mid x_0 = x, u_0 = u\right]. \tag{3}$$

In the context of deep reinforcement learning, either the policy, a value function or both are represented by neural networks.

2.2 Multi-agent reinforcement learning

When the sequential decision-making is extended to multiple agents, Markov Games[1] are commonly applied as framework. The Markov Game was originally introduced by Littman (1994) to generalize MDPs to multiple agents that simultaneously interact within a shared environment and possibly with each other. The definition is formalized in a discrete-time setting and is denoted as follows (Littman 1994).

Definition 2 *Markov Games (MG)* The Markov Game is an extension to the MDP and is formalized by the tuple $\left(\mathscr{N}, \mathscr{X}, \{\mathscr{U}^i\}, \mathscr{P}, \{R^i\}, \gamma\right)$, where $\mathscr{N} = \{1, \dots, N\}$ denotes the set of $N > 1$ interacting agents and $\mathscr{X}$ is the set of states observed by all agents. The joint action space is denoted by $\mathscr{U} = \mathscr{U}^1 \times \cdots \times \mathscr{U}^N$ which is the collection of individual action spaces from agents $i \in \mathscr{N}$. The transition probability function $\mathscr{P} : \mathscr{X}\times\mathscr{U}\to P(\mathscr{X})$ describes the chance of a state transition. Each agent owns an associated reward function $R^i : \mathscr{X}\times\mathscr{U}\times\mathscr{X}\to\mathbb{R}$ that provides an immediate feedback signal. Finally, $\gamma \in [0, 1)$ describes the discount factor.

At stage t, each agent $i \in \mathscr{N}$ selects and executes an action depending on the individual policy $\pi^i : \mathscr{X}\to P(\mathscr{U}^i)$. The system evolves from state x_t under the joint action u_t with respect to the transition probability function $\mathscr{P}$ to the next state x_{t+1} while each agent receives R^i as immediate feedback to the state transition. Akin to the single-agent problem, the aim of each agent is to change its policy in such a way as to optimize the received rewards on a long-term perspective.

A special case of the MG is the stateless setting $\mathscr{X} = \emptyset$ called strategic-form game[2]. Strategic-form games describe one-shot interactions where all agents simultaneously execute an action and receive a reward based on the joint action after which the game ends. Significant progress within the MARL community has been accomplished by studying this simplified stateless setting, which is still under active research to cope with several pathologies as discussed later in this section. These games are also known

[1] Markov games are also known as Stochastic Games (Shapley 1953), but we continue to use the term Markov Game to draw a clear distinction between deterministic Markov Games and stochastic Markov Games.

[2] The strategic-form game is also known as matrix game or normal-form game. The most commonly studied strategic-form game is the one with $N = 2$ players, the so-called bi-matrix game.

as matrix games because the reward function is represented by an $N \times N$ matrix. The formalism which extends to multi-step sequential stages is called extensive-form game.

In contrast to the single-agent case, the value function $V^i : \mathscr{X} \to \mathbb{R}$ does not only depend on the individual policy of agent i but also on the policies of other agents, i.e. the value function for agent i is the expected sum

$$V^i_{\pi^i, \boldsymbol{\pi}^{-i}}(x) = \mathbb{E}_{x_{t+1} \sim \mathscr{P}, u_t \sim \boldsymbol{\pi}} \left[\sum_{t=0}^{\infty} \gamma^t R^i(x_t, u_t, x_{t+1}) \mid x_0 = x \right] \tag{4}$$

when the agents behave according to the joint policy $\boldsymbol{\pi}$. We denote the joint policy $\boldsymbol{\pi} : \mathscr{X} \to P(\mathscr{U})$ as the collection of all individual policies, i.e. $\boldsymbol{\pi} = \{\pi^1, \ldots, \pi^N\}$. Further, we make use of the convention that $-i$ denotes all agents except i, meaning for policies that $\boldsymbol{\pi}^{-i} = \{\pi^1, \ldots, \pi^{i-1}, \pi^{i+1}, \ldots, \pi^N\}$.

The optimal policy is determined by the individual policy and the other agents' strategies. However, when other agents' policies are fixed, the agent i can maximize its own utility by finding the best response π^i_* with respect to the other agents' strategies.

Definition 3 *Best response* The agent's i best response $\pi^i_* \in \Pi^i$ to the joint policy $\boldsymbol{\pi}^{-i}$ of other agents is

$$V^i_{\pi^i_*, \boldsymbol{\pi}^{-i}}(x) \geq V^i_{\pi^i, \boldsymbol{\pi}^{-i}}(x)$$

for all states $x \in \mathscr{X}$ and policies $\pi^i \in \Pi^i$.

In general, when all agents learn simultaneously, the found best response may not be unique (Shoham and Leyton-Brown 2008). The concept of best response can be leveraged to describe the most influential solution concept from game theory: the *Nash equilibrium*.

Definition 4 *Nash equilibrium* A solution where each agent's policy π^*_i is the best response to the other agents' policy $\boldsymbol{\pi}^{-i}_*$ such that the following inequality

$$V^i_{\pi^i_*, \boldsymbol{\pi}^{-i}_*}(x) \;\geq\; V^i_{\pi^i, \boldsymbol{\pi}^{-i}_*}(x)$$

holds true for all states $x \in \mathscr{X}$ and all policies $\pi^i \in \Pi^i \; \forall i$ is called Nash equilibrium.

Intuitively spoken, a Nash equilibrium is a solution where one agent cannot improve when the policies of other agents are fixed, that is no agent can improve by unilaterally deviating from π^*. However, a Nash equilibrium may not be unique. Thus, the concept of Pareto-optimality might be useful (Matignon et al. 2012b).

Definition 5 *Pareto-optimality* A joint policy $\boldsymbol{\pi}$ Pareto-dominates a second joint policy $\hat{\boldsymbol{\pi}}$ if and only if

$$V^i_{\boldsymbol{\pi}}(x) \;\geq\; V^i_{\hat{\boldsymbol{\pi}}}(x) \quad \forall i, \forall x \in \mathscr{X} \quad \text{and} \quad V^j_{\boldsymbol{\pi}}(x) \;>\; V^j_{\hat{\boldsymbol{\pi}}}(x) \quad \exists j, \exists x \in \mathscr{X}.$$

A Nash equilibrium is regarded to be Pareto-optimal if no other has greater value and, thus, is not Pareto-dominated.

Classical MARL literature can be categorized according to different features, such as the type of task and the information available to agents. In the remainder of this section,

we introduce MARL concepts based on the taxonomy proposed in Busoniu et al. (2008). For one, the primary factor that influences the learned agent behavior is the type of task. Whether agents compete or cooperate is promoted by the designed reward structure.

(1) *Fully cooperative setting* All agents receive the same reward $R = R^i = \cdots = R^N$ for state transitions. In such an equally-shared reward setting, agents are motivated to collaborate and try to avoid the failure of an individual to maximize the performance of the team. More generally, we talk about *cooperative settings* when agents are encouraged to collaborate but do not own an equally-shared reward.

(2) *Fully competitive setting* Such problem is described as a *zero-sum* Markov Game where the sum of rewards equals zero for any state transition, i.e. $R = \sum_{i=1}^{N} R^i(x, u, x') = 0$. Agents are prudent to maximize their own individual reward while minimizing the reward of the others. In a loose sense, we refer to competitive games when agents are encouraged to excel against opponents, but the sum of rewards does not equal zero.

(3) *Mixed setting* Also known as *general-sum* game, the mixed setting is neither fully cooperative nor fully competitive and, thus, does not incorporate restrictions on agent goals.

Beside the reward structure, other taxonomy may be used to differentiate between the information available to the agents. Claus and Boutilier (1998) distinguished between two types of learning, namely independent learners and joint-action learners. The former ignores the existence of other agents and cannot observe the rewards and selected actions of others as considered in Bowling and Veloso (2002) and Lauer and Riedmiller (2000). Joint-action learners, however, observe the taken actions of all other actions a-posteriori as shown in Hu and Wellman (2003) and Littman (2001).

2.3 Formal introduction to multi-agent challenges

In the single-agent formalism, the agent is the only decision-instance that influences the state of the environment. State transitions can be clearly attributed to the agent, whereas everything outside the agent's field of impact is regarded as part of the underlying system dynamics. Even though the environment may be stochastic, the learning problem remains stationary.

On the contrary, one of the fundamental problems in the multi-agent domain is that agents update their policies during the learning process simultaneously, such that the environment appears non-stationary from the perspective of a single agent. Hence, the Markov assumption of an MDP no longer holds, and agents face—without further treatment—a moving target problem (Busoniu et al. 2008; Yang and Gu 2004).

Definition 6 *Non-stationarity* A single agent faces a moving target problem when the transition probability function changes

$$\mathscr{P}(x' \mid x, u, \pi^1, \ldots, \pi^N) \neq \mathscr{P}(x' \mid x, u, \bar{\pi}^1, \ldots, \bar{\pi}^N),$$

due to the co-adaption $\pi^i \neq \bar{\pi}^i \; \exists \; i \in \mathscr{N}$ of agents.

Above, we have introduced the Nash equilibrium as a solution concept where each agent's policy is the best response to the others. However, it has been shown that agents can converge, despite a high degree of randomness in action selection, to sub-optimal solutions

or can get stuck between different solutions (Wiegand 2004). Fulda and Ventura (2007) investigated such convergence to solutions and described a Pareto-selection problem called *shadowed equilibrium*.

Definition 7 *Shadowed equilibrium* A joint policy $\bar{\boldsymbol{\pi}}$ is shadowed by another joint policy $\hat{\boldsymbol{\pi}}$ in a state x if and only if

$$V_{\pi^i,\bar{\boldsymbol{\pi}}^{-i}}(x) < \min_{j,\pi_j} V_{\pi^j,\hat{\boldsymbol{\pi}}^{-j}}(x) \quad \exists\, i, \pi_i. \tag{5}$$

An equilibrium is shadowed by another when at least one agent exists who, when unilaterally deviating from $\bar{\boldsymbol{\pi}}$, will see no better improvement than for deviating from $\hat{\boldsymbol{\pi}}$ (Matignon et al. 2012b). As a form of shadowed equilibrium, the pathology of *relative overgeneralization* describes that a sub-optimal Nash equilibrium in the joint action space is preferred over an optimal solution. This phenomenon arises since each agent's policy performs relatively well when paired with arbitrary actions from other agents (Panait et al. 2006; Wei and Luke 2016; Wiegand 2004).

In a Markov Game, we assumed that each agent observes a state x, which encodes all necessary information about the world. However for complex systems, complete information might not be perceivable. In such partially observable settings, the agents do not observe the whole state space but merely a subset $\mathcal{O}^i \subset \mathcal{X}$. Hence, the agents are confronted to deal with sequential decision-making under uncertainty. The partially observable Markov Game (Hansen et al. 2004) is the generalization of both MG and MDP.

Definition 8 *Partially observable Markov Games (POMG)* The POMG is mathematically denoted by the tuple $\left(\mathcal{N}, \mathcal{X}, \{\mathcal{U}^i\}, \{\mathcal{O}^i\}, \mathcal{P}, \{R^i\}, \gamma\right)$, where $\mathcal{N} = \{1, \ldots, N\}$ denotes the set of $N > 1$ interacting agents, $\mathcal{X}$ is the set of global but unobserved system states, and $\mathcal{U}$ is the set of individual action spaces $\mathcal{U}_i$. The observation space $\mathcal{O}$ denotes the collection of individual observation spaces $\mathcal{O}^i$. The transition probability function is denoted by $\mathcal{P}$, the reward function associated with agent i by R^i, and the discount factor is γ.

When agents face a cooperative task with a shared reward function, the POMG is then known as *decentralized Partially Observable Markov decision process* (dec-POMDP) (Bernstein et al. 2002; Oliehoek and Amato 2016). In partially observable domains, the inference of good policies is extended in complexity since the history of interactions becomes meaningful. Hence, the agents usually incorporate history-dependent policies $\pi_t^i : \{\mathcal{O}^i\}_{t>0} \rightarrow P(\mathcal{U}^i)$, which map from a history of observations to a distribution over actions.

Definition 9 *Credit assignment problem* In the fully-cooperative setting with joint reward signals, an individual agent cannot conclude the impact of its own action towards the team's success and, thus, faces a credit assignment problem.

In cooperative games, agents are encouraged to maximize a common goal through a joint reward signal. However, agents cannot ascertain their contribution to the eventual reward when they do not experience the taken joint action or deal with partial observations. Associating rewards to agents is known as the *credit assignment problem* (Chang et al. 2004; Weiß 1995; Wolpert and Tumer 1999).

Some of the above-introduced pathologies occur in all cooperative, competitive, and mixed tasks, whereas some pathologies like relative over-generalization, credit assignment, and miss-coordination are predominant issues in cooperative settings. To cope with these pathologies, still commonly studied settings are tabular worlds such as variations of the climbing game where solutions are not yet found, e.g. when the environment exhibits reward stochasticity (Claus and Boutilier 1998). Thus, simple worlds remain a fertile ground for further research, especially for problems like shadowed equilibria, non-stationarity or alter-exploration problems[3] and continue to matter for modern deep learning approaches.

3 Analysis of training schemes

The training of multiple agents has long been a computational challenge (Becker et al. 2004; Nair et al. 2003). Since the complexity in the state and action space grows exponentially with the number of agents, even modern deep learning approaches may reach their limits. In this section, we describe training schemes that are used in practice for learning agent policies in the multi-agent setting similar to the ones described in Bono et al. (2019). We denote training as the process during which agents acquire data to build up experience and optimize their behavior with respect to the received reward signals. In contrast, we refer test time[4] to the step after the training when the learned policy is evaluated but is no further refined. The training of agents can be broadly divided into two paradigms, namely centralized and distributed (Weiß 1995). If the training of agents is applied in a centralized manner, policies are updated based on the mutual exchange of information during the training. This additional information is then usually removed at test time. In contrast to the centralized scheme, the training can also be handled in a distributed fashion where each agent performs updates on its own and develops an individual policy without utilizing foreign information.

In addition to the training paradigm, agents may deviate in the way of how they select actions. We recognize two execution schemes. Centralized execution describes that agents are guided from a centralized unit, which computes the joint actions for all agents. On the contrary, agents determine actions according to their individual policy for decentralized execution. An overview of the training schemes is depicted in Fig. 2 while Table 1 lists the reviewed literature of this section.

3.1 Distributed training

In distributed training schemes, agents learn independently of other agents and do not rely on explicit information exchange.

[3] The alter-exploration dilemma, also known as the exploration-exploitation problem, describes the trade-off an agent faces to decide whether to choose actions that extend experience or take decisions that are already optimal according to the current knowledge.

[4] Note that test and execution time are often used interchangeably in recent literature. For clarity, we use the term test for the post-training evaluation and the term execution for the action selection with respect to some policy.

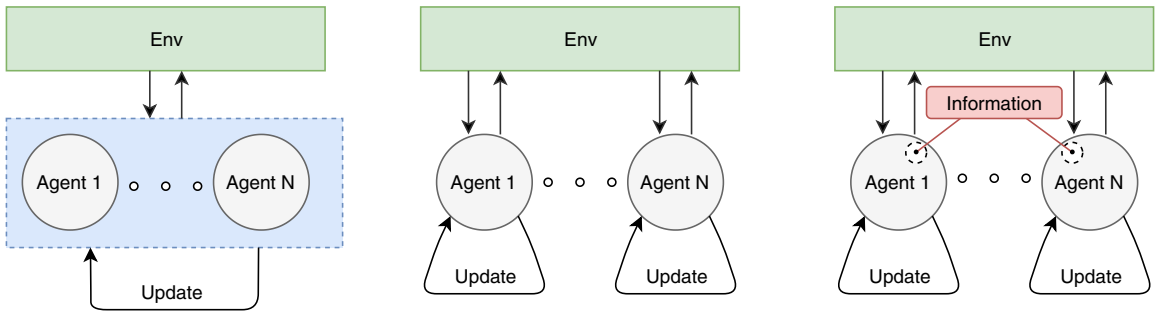


Fig. 2 Training schemes in the multi-agent setting. (Left) CTCE holds a joint policy for all agents. (Middle) Each agent updates its own individual policy in DTDE. (Right) CTDE enables agents to exchange additional information during training which is then discarded at test time

Definition 10 *Distributed training decentralized execution (DTDE)* Each agent i has an associated policy $\pi^i : \mathcal{O}^i \rightarrow P(\mathcal{U}^i)$ which maps local observations to a distribution over individual actions. No information is shared between agents such that each agent learns independently.

The fundamental drawback of the DTDE paradigm is that the environment appears non-stationary from a single agent's viewpoint because agents neither have access to the knowledge of others, nor do they perceive the joint action. The first approaches in this training scheme were studied in tabular worlds. The work by Tan (1993) investigated the question if independently learning agents can match with cooperating agents. The results showed that independent learners learn slower in tabular and deterministic worlds. Based on that, Claus and Boutilier (1998) examined both independent and joint-action learners in cooperative stochastic-form games and empirically showed that both types of learning can converge to an equilibrium in deterministic games. Subsequent works elaborated on the DTDE scheme in discretized worlds (Hu and Wellman 1998; Lauer and Riedmiller 2000).

More recent works report that distributed training schemes scale poorly with the number of agents due to the extra sample complexity, which is added to the learning problem. Gupta et al. (2017) showed that distributed methods have inferior performance compared to policies that are trained with a centralized training paradigm. Similarly, Foerster et al. (2018b) showed that the speed of independently learning actor-critic methods is slower than using centralized training. In further works, DTDE has been applied to cooperative navigation tasks (Chen et al. 2016; Strouse et al. 2018), to partially observable domains (Dobbe et al. 2017; Nguyen et al. 2017b; Srinivasan et al. 2018), and to social dilemmas (Leibo et al. 2017).

Due to limited information in the distributed setting, independent learners are confronted with several pathologies (Matignon et al. 2012b). Besides non-stationarity, environments may exhibit stochastic transitions or stochastic rewards, which further complicates learning. In addition to that, the search for an optimal policy influences the other agents' decision-making, which may lead to action shadowing and impacts the balance between exploration and knowledge exploitation.

A line of recent works expands independent learners with techniques to cope with the aforementioned MARL pathologies in cooperative domains. First, Omidshafiei et al. (2017) introduced a decentralized experience replay extension called Concurrent Experience Replay Trajectories (CERT) that enables independent learners to face a cooperative

Table 1 Overview of training schemes applied in recent MADRL works

Scheme	Approach	Literature
CTCE		Gupta et al. (2017) and Sunehag et al. (2018)
CTDE	Parameter sharing	Ahilan and Dayan (2019), Chu and Ye (2017), Gupta et al. (2017), Peng et al. (2017), Sukhbaatar et al. (2016) and Sunehag et al. (2018)
	Value-based	Castellini et al. (2019), Foerster et al. (2016), Jorge et al. (2016), Rashid et al. (2018), Son et al. (2019) and Sunehag et al. (2018)
	Centralized critic	Bono et al. (2019), Das et al. (2019), Foerster et al. (2018b), Lowe et al. (2017), Iqbal and Sha (2019), Wei et al. (2018) and Wu et al. (2018)
	Master-slave	Kong et al. (2017) and Kumar et al. (2017)
DTDE		Chen et al. (2016), Dobbe et al. (2017), Foerster et al. (2018b), Gupta et al. (2017), Jaderberg et al. (2019), Jaques et al. (2019), Leibo et al. (2017), Liu et al. (2019), Lyu and Amato (2020), Nguyen et al. (2017b), Omidshafiei et al. (2017), Palmer et al. (2018), Palmer et al. (2019), Srinivasan et al. (2018), Strouse et al. (2018) and Zheng et al. (2018a)

and partially observable setting by rendering samples more stable and efficient. Similarly, Palmer et al. (2018) extended the experience replay of Deep Q-Networks with leniency, which associates stored state-action pairs with decaying temperature values that govern the amount of applied leniency. They showed that this induces optimism in value function updates and can overcome relative over-generalization. Another work by Palmer et al. (2019) proposed negative update intervals double-DQN as an mechanism that identifies and removes generated data from the replay buffer that leads to mis-coordination. Alike, Lyu and Amato 2020 proposed decentralized quantile estimators which identify non-stationary transition samples based on the likelihood of returns. Another work that aims to improve upon independent learners can be found in Zheng et al. (2018a) who used two auxiliary mechanisms, including a lenient reward approximation and a prioritized replay strategy.

A different research direction can be seen in distributed population-based training schemes where agents are optimized through an online evolutionary process such that under-performing agents are substituted by mutated versions of better agents (Jaderberg et al. 2019; Liu et al. 2019).

3.2 Centralized training

The centralized training paradigm describes agent policies that are updated based on mutual information. While the sharing of mutual information between agents is enabled during the training, this additional information is then discarded at test time. The centralized training can be further differentiated into the centralized and decentralized execution scheme.

Definition 11 *Centralized training centralized execution (CTCE)* The CTCE scheme describes a centralized executor $\pi : \mathscr{O} \rightarrow P(\mathscr{U})$ modeling the joint policy that maps the collection of distributed observations to a set of distributions over individual actions.

Some applications assume an unconstrained and instantaneous information exchange between agents. In such a setting, a centralized executor can be leveraged to learn the joint policy for all agents. The CTCE paradigm allows the straightforward employment of single-agent training methods such as actor-critics (Mnih et al. 2016) or policy gradient algorithms (Schulman et al. 2017) to multi-agent problems. An obvious flaw is that state-action spaces grow exponentially by the number of agents. To address the so-called *curse of dimensionality*, the joint model can be factored into individual policies for each agent. Gupta et al. (2017) represented the centralized executor as a set of independent sub-policies such that agents' individual action distributions are captured rather than the joint action distribution of all agents, i.e. the joint action distribution $P(\mathscr{U}) = \prod_i P(\mathscr{U}^i)$ is factored into independent action distributions. Next to the policy, the value function can be factored so that the joint value is decomposed into a sum of local value functions, e.g. the joint action-value function can be expressed by $Q_\pi(o^1, \ldots, o^N, u^1, \ldots, u^n) = \sum_i Q^i_\pi(o^i, u^i)$ as shown in Russell and Zimdars (2003). A recent approach for the value function factorization is investigated in Sunehag et al. (2018). However, a phenomenon called *lazy agents* may occur in the CTCE setting when one agent learns a good policy but a second agent has less incentive to learn a good policy, as his actions may hinder the first agent, resulting in a lower reward (Sunehag et al. 2018).

Although CTCE regards the learning problem as a single-agent case, we include the paradigm in this paper because the training schemes presented in the subsequent sections occasionally use CTCE as performance baseline and conduct comparisons.

Definition 12 *Centralized training decentralized execution (CTDE)* Each agent i holds an individual policy $\pi^i : \mathcal{O}^i \rightarrow P(\mathcal{U}^i)$ which maps local observations to a distribution over individual actions. During training, agents are endowed with additional information, which is then discarded at test time.

The CTDE paradigm presents the state-of-the-art practice for learning with multiple agents (Kraemer and Banerjee 2016; Oliehoek et al. 2008). In classical MARL, such setting was utilized as *joint action learners* which has the advantage that perceiving joint actions a-posteriori discards the non-stationarity in the environment (Claus and Boutilier 1998). As of late, CTDE has been successful in MADRL approaches (Foerster et al. 2016; Jorge et al. 2016). Agents utilize shared computational facilities or other forms of communication to exchange information during training. By sharing mutual information, the training process can be eased and the learning speed can become superior when matched against independently trained agents (Foerster et al. 2018b). Moreover, agents can bypass non-stationarity when extra information about the selected actions is available to all agents during training such that the consequences of actions can be attributed to the respective agents. In what follows, we classify the CTDE literature according to the agent structure.

Homogeneous agents exhibit a common structure or the same set of skills, e.g. the same learning model or share common goals. Owning the same structure, agents can share parts of their learning model or experience with other agents. These approaches can scale well with the number of agents and may allow an efficient learning of behaviors. Gupta et al. (2017) showed that policies based on *parameter sharing* can be trained more efficiently and, thus, can outperform independently learned ones. Although agents own the same policy network, different agent behaviors can emerge because each agent perceives different observations at test time. It has been thoroughly demonstrated that parameter sharing can help to accelerate the learning progress (Ahilan and Dayan 2019; Chu and Ye 2017; Peng et al. 2017; Sukhbaatar et al. 2016; Sunehag et al. 2018). Next to parameter sharing, homogeneous agents can employ *value-based* methods where an approximation of the value function is learned based on mutual information. Agents profit from the joint actions and other agents' policies that are available during training and incorporate this extra information into centralized value functions (Foerster et al. 2016; Jorge et al. 2016). Such information is then discarded at test time. Many approaches consider the decomposition of a joint value function into combinations of individual value functions (Castellini et al. 2019; Rashid et al. 2018; Son et al. 2019; Sunehag et al. 2018). Through decomposition, each agent faces a simplified sub-problem of the original problem. Sunehag et al. (2018) showed that agents learning on local sub-problems scale better with the number of agents than CTCE or independent learners. We elaborate on value function-based factorization more detailed in Sect. 5.4 as an effective approach to tackle credit assignment problems.

Heterogeneous agents, on the contrary, differ in structure and skill. An instance for heterogeneous policies can be seen in the extension of an actor-critic approach with a *centralized critic*, which allows information sharing to amplify the performance of individual agent policies. These methods can be distinguished from each other based on the representation of the critic. Lowe et al. (2017) utilized one centralized critic for each agent that is augmented with additional information during training. The critics are provided with

information about every agent's policy, whereas the actors perceive only local observations. As a result, the agents do not depend on explicit communication and can overcome the non-stationarity in the environment. Likewise, Bono et al. (2019) trained multiple agents with individual policies that share information with a centralized critic and demonstrated that such setup might improve results on standard benchmarks. Besides the utilization of one critic for each agent, Foerster et al. (2018b) applied one centralized critic for all agents to estimate a counterfactual baseline function that marginalizes out a single agent's action. The critic is conditioned on the history of all agents' observations or, if available, on the true global state. Typically, actor-critic methods underlie a variance in the critic estimation that is further exacerbated by the number of agents. Therefore, Wu et al. (2018) proposed an action-dependent baseline which includes information from other agents to reduce the variance in the critic estimation function. Further works that incorporate one centralized critic for distributed policies can be found in Das et al. (2019), Iqbal and Sha (2019) and Wei et al. (2018).

Another way to perform decentralized execution is by employing a *master-slave* architecture, which can resolve coordination conflicts between multiple agents. Kong et al. (2017) applied a centralized master executor which shares information with decentralized slaves. In each time step, the master receives local information from the slaves and shares its internal state in return. The slaves compute actions conditioned on their local observation and the master's internal state. Similar approaches that make use of different levels of abstraction are hierarchical methods (Kumar et al. 2017) that operate at different time scales or levels of abstraction. We elaborate on hierarchical methods in more detail in Sect. 5.3.

4 Emergent patterns of agent behavior

Agents adjust their policy to maximize the task success and react to the behavioral changes of other agents. The dynamic interaction between multiple decision-makers, which simultaneously affects the state of the environment, can cause the emergence of specific behavioral patterns. An obvious way to influence the development of agent behavior is through the designed reward structure. By promoting incentives for cooperation, agents can learn team strategies where they try to collaborate and optimize upon a mutual goal. Agents support other agents since the cumulative reward for cooperation is greater than acting selfishly. On the contrary, if the appeals for maximizing the individual performance are larger than being cooperative, agents can learn greedy strategies and maximize their individual reward. Such competitive attitudes can yield high-level strategies like manipulating adversaries to gain an advantage. However, the boundaries between competition and cooperation can be blurred in the multi-agent setting. For instance, if one agent competes with other agents, it is sometimes useful to cooperate temporarily in order to receive a higher reward in the long run.

In this section, we review the literature that is interested in developed agent behaviors. We differentiate occurring behaviors according to the reward structure (Sect. 4.1), the language between agents (Sect. 4.2), and the social context (Sect. 4.3). Table 2 summarizes the reviewed literature based on this classification. Note that we focus in this section not on works that introduce new methodologies but on literature that analyzes the emergent behavioral patterns.

Table 2 Overview of MADRL papers that investigate emergent patterns of agent behavior

Emergence	Setting	Literature
Reward structure	Cooperative	Diallo et al. (2017), Leibo et al. (2017) and Tampuu et al. (2017)
	Competitive	Bansal et al. (2018), Leibo et al. (2017), Liu et al. (2019) and Tampuu et al. (2017)
	Intrinsic rewards	Baker et al. (2020), Hughes et al. (2018), Jaderberg et al. (2019), Jaques et al. (2019), Jaques et al. (2018), Peysakhovich and Lerer (2018), Sukhbaatar et al. (2017), Wang et al. (2019) and Wang et al. (2020b)
Language	Referential games	Choi et al. (2018), Evtimova et al. (2018), Havrylov and Titov (2017), Jorge et al. (2016), Lazaridou et al. (2017), Lazaridou et al. (2018), Lee et al. (2017) and Mordatch and Abbeel (2018)
	Dialogues	Cao et al. (2018), Das et al. (2017) and Lewis et al. (2017)
Social context	Commons dilemmas	Foerster et al. (2018a), Jaques et al. (2018), Jaques et al. (2019), Leibo et al. (2017) and Lerer and Peysakhovich (2017)
	Public good dilemmas	Pérolat et al. (2017), Hughes et al. (2018) and Zhu and Kirley (2019)

4.1 Reward structure

The primary factor that influences the emergence of agent behavior is the reward structure. If the reward for mutual *cooperation* is larger than individual reward maximization, agents tend to learn policies that seek to collaboratively solve the task. In particular, Leibo et al. (2017) compared the magnitude of the team reward in relation to the individual agent reward. They showed that the higher the numerical team reward is compared to the individual reward, the greater is the willingness to collaborate with other agents. The work by Tampuu et al. (2017) demonstrated that punishing the whole team of agents for the failure of a single agent can also cause cooperation. Agents learn policies to avoid the malfunction of an individual, support other agents to prevent failure, and improve the performance of the whole team. Similarly, Diallo et al. (2017) used the Pong video game to investigate the coordination between agents and examined how developed behaviors change regarding the reward function. For a comprehensive review of learning in cooperative settings, one can consider the article by Panait and Luke (2005) for classical MARL and Oroojlooyjadid and Hajinezhad (2019) for recent MADRL.

In contrast to the cooperative scenario, one can value individual performance greater than the collaboration among agents. A *competitive* setting motivates agents to outperform their adversary counterparts. Tampuu et al. (2017) used the video game Pong and manipulated the rewarding structure to examine the emergence of agent behavior. They showed that the higher the reward for competition, the more likely an agent tries to outplay its opponents by using techniques such as wall bouncing or faster ball speed. Employing such high-level strategies to overwhelm the adversary maximizes the individual reward. Similarly, Bansal et al. (2018) investigated competitive scenarios, where agents competed in a 3D world with simulated physics to learn locomotion skills such as running, blocking, or tackling other agents with arms and legs. They argued that adversarial training could help to learn more complex agent behaviors than the environment can exhibit. Likewise, the works of Leibo et al. (2017) and Liu et al. (2019) investigated the emergence of behaviors due to the reward structure in competitive scenarios.

If the rewards appear in sparse frequency, agents can be equipped with *intrinsic reward functions* that provide denser feedback signals and, thus, can overcome the sparsity or even the absence of external rewards. One way to realize this is with intrinsic motivation, which is based on the concept of maximizing an internal reinforcement signal by actively discovering novel or surprising patterns (Chentanez et al. 2005; Oudeyer and Kaplan 2007; Schmidhuber 2010). Intrinsic motivation encourages agents to explore states that have been scarcely or never visited and to perform novel actions in those states. Most approaches of intrinsic motivation can be broadly divided into two categories (Pathak et al. 2017). First, agents are encouraged to explore unknown states where the novelty of states is measured by a model that captures the distribution of visited environment states (Bellemare et al. 2016). Second, agents can be motivated to reduce the uncertainty about the consequences of their own actions. The agent builds a model that learns the dynamics of the environment by lowering the prediction error of the follow-up states with respect to the taken actions. The uncertainty indicates the novelty of new experience since the model can only be accurate in states which it has already encountered or can generalize from previous knowledge (Houthooft et al. 2016; Pathak et al. 2017). For a recent survey on intrinsic motivation in RL, one can regard the paper by Aubret et al. (2019). The concept of intrinsic motivation was transferred to the multi-agent domain by Sequeira et al. (2011), who studied the motivational impact on multiple agents. Investigations on the emergence of agent behavior based

on intrinsic rewards have been abundantly conducted in Baker et al. (2020), Hughes et al. (2018), Jaderberg et al. (2019), Jaques et al. (2018), Jaques et al. (2019), Peysakhovich and Lerer (2018), Sukhbaatar et al. (2017), Wang et al. (2019) and Wang et al. (2020b).

4.2 Language

The development of language corpora and communication skills of autonomous agents attracts great attention within the community. For one, the behavior that emerges during the deployment of abstract language as well as the learned composition of multiple words to form meaningful contexts is of interest (Kirby 2002). Deep learning methods have widened the scope of computational methodologies for investigating the development of language between dynamic agents (Lazaridou and Baroni 2020). For building rich behaviors and complex reasoning, communication based on high-dimensional data like visual perception is a widespread practice (Antol et al. 2015). In the following, we focus on works that investigate the emergence of language and analyze behavior. Papers that propose new methodologies for developing communication protocols are discussed in Sect. 5.2. We classify the learning of language according to the performed task and the type of interaction the agents pursue. In particular, we differentiate between referential games and dialogues.

The former, *referential games*, describe cooperative games where the speaking agent communicates an objective via messages to another listening agent. Lazaridou et al. (2017) showed that agents could learn communication protocols solely through interaction. For a meaningful information exchange, agents evolved semantic properties in their language. A key element of the study was to analyze if the agents' interactions are interpretable for humans, showing limited yet encouraging results. Likewise, Mordatch and Abbeel (2018) investigated the emergence of abstract language that arises through the interaction between agents in a physical environment. In their experiments, the agents should learn a discrete set of vocabulary by solving navigation tasks through communication. By involving more than three agents in the conversation and by penalizing an arbitrary size of vocabulary, agents agreed on a coherent set of vocabulary and discouraged ambiguous words. They also observed that agents learned a syntax structure in the communication protocol that is consistent in vocabulary usage. Another work by Li and Bowling (2019) found out that compositional languages are easier to communicate with other agents than languages with less structure. In addition, changing listening agents during the learning can promote the emergence of language grounded on a higher degree of structure. Many studies are concerned with the development of communication in referential games grounded on visual perception as it can be found in Choi et al. (2018), Evtimova et al. (2018), Havrylov and Titov (2017), Jorge et al. (2016), Lazaridou et al. (2018) and Lee et al. (2017). Further works consider the development of communication in social dilemmas (Jaques et al. 2018, 2019).

As the second category, we describe the emergence of behavioral patterns in communication while conducting *dialogues*. One type of dialogue are negotiations in which agents pursue to agree on decisions. In a study about negotiations with natural language, Lewis et al. (2017) showed that agents could master linguistic and reasoning problems. Two agents were both shown a collection of items and were instructed to negotiate about how to divide the objects among both agents. Each agent was expected to maximize the value of the bargained objects. Eventually, the agents learned to use high-level strategies such as deception to accomplish higher rewards over their opponents. Similar studies concerned with negotiations are covered in Cao et al. (2018) and He et al. (2018). Another

type of dialogue are scenarios where the emergence of communication is investigated in a question-answering style as shown by Das et al. (2017). One agent received an image as input and was instructed to ask questions about the shown image while the second agent responded, both in natural language.

Many of the above-mentioned papers report that utilizing a communication channel can increase task performance in terms of the cumulative reward. However, numerical performance measurements provide evidence but do not give insights about the communication abilities learned by the agents. Therefore, Lowe et al. (2019) surveyed metrics which are applied to assess the quality of learned communication protocols and provided recommendations about the usage of such metrics. Based on that, Eccles et al. (2019) proposed to incorporate inductive bias into the learning objective of agents, which could promote the emergence of a meaningful communication. They showed that inductive bias could lead to improved results in terms of interpretability.

4.3 Social context

Next to the reward structure and language, the research community actively investigates the emerging agent behaviors in social contexts. Akin to humans, artificial agents can develop strategies that exploit patterns in complex problems and adapt behaviors in response to others (Baker et al. 2020; Jaderberg et al. 2019). We differentiate the following literature along different dimensions, such as the type of social dilemma and the examined psychological variables.

Social dilemmas have long been studied as conflict scenario in which agents gauge between individualistic and collective profits (Crandall and Goodrich 2011; De Cote et al. 2006). The tension between cooperation and defection is evaluated as an atomic decision according to the numerical values of a pay-off matrix. This pay-off matrix satisfies inequalities in the reward function such that agents must decide between cooperation, to benefit as a whole team, or defection, to maximize selfish performance. To temporally extend matrix games, sequential social dilemmas have been introduced to investigate long-term strategic decisions of agent policies rather than short-term actions (Leibo et al. 2017). The arising behaviors in these dilemmas can be classified along psychological variables known from human interaction (Lange et al. 2013) such as the gain of individual benefits (Lerer and Peysakhovich 2017), the fear of future consequences (Pérolat et al. 2017), the assessment of the impact on another agent's behavior (Jaques et al. 2018, 2019), the trust between agents (Pinyol and Sabater-Mir 2013; Ramchurn et al. 2004; Yu et al. 2013), and the impact of emotions on the decision-making (Moerland et al. 2018; Yu et al. 2013).

Kollock (1998) divided social dilemmas into commons dilemmas and public goods dilemmas. The former, *commons dilemmas* describe the trade-off between individualistic short-term benefits and long-term common interests on a task that is shared by all agents. Recent works on the commons dilemma can be found in Foerster et al. (2018a), Leibo et al. (2017) and Lerer and Peysakhovich (2017). In *public goods dilemmas*, agents face a scenario where common-pool resources are constrained and oblige a sustainable use of resources. The phenomenon called the tragedy of commons predicts that self-interested agents fail to find socially positive equilibria, which eventually results in the over-exploitation of the common resources (Hardin 1968). Investigations on the trial-and-error learning in common-pool resource scenarios with multiple decision-makers are covered in Hughes et al. (2018), Pérolat et al. (2017) and Zhu and Kirley (2019).

5 Current challenges

In this section, we depict several challenges that arise in the multi-agent RL domain and, thus, are currently under active research. We approach the problem of non-stationarity (Sect. 5.1) due to the presence of multiple learners in a shared environment and review literature regarding the development of communication skills (Sect. 5.2). We further investigate the challenge of learning coordination (Sect. 5.3). Then, we survey the difficulty of attributing rewards to specific agents as the credit assignment problem (Sect. 5.4) and examine scalability issues (Sect. 5.5), which increase with the number of agents. Finally, we consider environments where states are only partially observable (Sect. 5.6). While some challenges are omnipresent in the MARL domain, such as non-stationarity or scalability, others like the credit assignment problem or the learning of coordination and communication are prevailing in the cooperative setting.

We aim to provide a holistic overview of the contemporary challenges that constitute the landscape in reinforcement learning with multiple agents and survey treatments that were suggested in recent works. In particular, we focus on those challenges which are currently under active research and where progress has been accomplished recently. There are still open problems that have not been or partially addressed so far. Such problems are discussed in Sect. 6. Deliberately, we do not regard challenges that also persist in the single-agent domain, such as sparse rewards or the exploration-exploitation dilemma. We refer the interested reader for an overview of those topics to the articles of Arulkumaran et al. (2017) and Li (2018). Much of the surveyed literature cannot be assigned to one particular but rather to several of the proposed challenges. Hence, we associate the subsequent literature to the one challenge which we believe best addresses it (Table 3).

5.1 Non-stationarity

One major problem resides in the presence of multiple agents that interact within a shared environment and learn simultaneously. Due to the co-adaption, the environment dynamics appear non-stationary from the perspective of a single agent. Thus, agents face a moving target problem if they are not provided with additional knowledge about other agents. As a result, the Markov assumption is violated, and the learning constitutes an inherently difficult problem (Hernandez-Leal et al. 2017; Laurent et al. 2011). The naïve approach is to neglect the adaptive behavior of agents. One can either ignore the existence of other agents (Matignon et al. 2012b) or discount the adaptive behavior by assuming the others' behavior to be static or optimal (Lauer and Riedmiller 2000). By making such assumptions, the agents are considered as independent learners, and traditional single-agent reinforcement algorithms can be applied. First attempts have been studied in Claus and Boutilier (1998) and Tan (1993), which showed that independent learners could perform well in simple deterministic environments. However, in complex or stochastic environments, independent learners often result in poor performance (Lowe et al. 2017; Matignon et al. 2012b). Moreover, Lanctot et al. (2017) argued that independent learners could over-fit to other agents' policies during the training and, thus, may fail to generalize at test time.

In the following, we review literature, which addresses the non-stationarity in a multi-agent environment, and categorize the approaches into those with experience replay, centralized units, and meta-learning. A similar categorization proposed Papoudakis et al. (2019). We identify further approaches which cope with non-stationarity by establishing

Table 3 Overview of MADRL challenges and approaches proposed in recent literature

Challenge	Approach	Literature
Non-stationarity	Experience replay	Foerster et al. (2017), Palmer et al. (2018), Tang et al. (2018), and Zheng et al. (2018a)
	Centralized training	Bono et al. (2019), Foerster et al. (2016), Foerster et al. (2018b), Iqbal and Sha (2019), Jorge et al. (2016), Lowe et al. (2017), Rashid et al. (2018), and Wei et al. (2018)
	Meta-learning	Al-Shedivat et al. (2018), and Rabinowitz et al. (2018)
Communication	Broadcasting	Foerster et al. (2016), Peng et al. (2017), and Sukhbaatar et al. (2016)
	Targeted	Das et al. (2019), Hoshen (2017), Jain et al. (2019), Jiang and Lu (2018), and Singh et al. (2019)
	Networked	Chu et al. (2020), Chu et al. (2020), Qu et al. (2020), Zhang et al. (2018), and Zhang et al. (2019)
	Extensions	Celikyilmaz et al. (2018), Jaques et al. (2018), Jaques et al. (2019), Kim et al. (2019), Li et al. (2019b), Singh et al. (2019), and Wang et al. (2020c)
Coordination	Independent learners	Foerster et al. (2018b), Lyu and Amato (2020), Omidshafiei et al. (2017), Palmer et al. (2018), Palmer et al. (2019), Sunehag et al. (2018), and Zheng et al. (2018a)
	Constructing models	Barde et al. (2019), Everett and Roberts (2018), Foerster et al. (2018a), Foerster et al. (2019), Grover et al. (2018), He et al. (2016), Hong et al. (2017), Hoshen (2017), Jaques et al. (2019), Le et al. (2017), Letcher et al. (2019), Raileanu et al. (2018), Tacchetti et al. (2019), Yang et al. (2018a), and Zheng et al. (2018b)
	Hierarchical methods	Ahilan and Dayan (2019), Cai et al. (2013), Han et al. (2019), Jaderberg et al. (2019), Kumar et al. (2017), Lee et al. (2020), Ma and Wu (2020), Tang et al. (2018), and Vezhnevets et al. (2019)
Credit assignment	Decomposition	Castellini et al. (2019), Chen et al. (2018), Nguyen et al. (2017b), Rashid et al. (2018), Son et al. (2019), Sunehag et al. (2018), Wang et al. (2020a), Wang et al. (2020c), Yang et al. (2018b)
	Marginalization	Foerster et al. (2018b), Nguyen et al. (2018), and Wu et al. (2018)
	Inverse RL	Barrett et al. (2017), Le et al. (2017), Lin et al. (2018), Song et al. (2018), and Yu et al. (2019)
Scalability	Knowledge Reuse	Baker et al. (2020), Da Silva et al. (2017), Da Silva and Costa (2017), Gupta et al. (2017), Hernandez-Leal et al. (2019), Jiang and Lu (2018), Long et al. (2020), Luketina et al. (2019), Narvekar et al. (2016), Omidshafiei et al. (2019), Peng et al. (2017), Sukhbaatar et al. (2016), Sukhbaatar et al. (2017), Sunehag et al. (2018), and Svetlik et al. (2017)
	Complexity reduction	Chen et al. (2018), Lin et al. (2018), Nguyen et al. (2017a), Nguyen et al. (2017b), and Yang et al. (2018b)
	Robustness	Baker et al. (2020), Bansal et al. (2018), Berner et al. (2019), Gleave et al. (2020), Heinrich and Silver (2016), Lanctot et al. (2017), Li et al. (2019a), Liu et al. (2020), Lowe et al. (2017), Pinto et al. (2017), Raghu et al. (2018), Silver et al. (2016), Silver et al. (2018), Spooner and Savani (2020), and Sukhbaatar et al. (2017)

Table 3 (continued)

Challenge	Approach	Literature
Partial observability	Memory mechanism	Dibangoye and Buffet (2018), Foerster et al. (2018b), Foerster et al. (2019), Gupta et al. (2017), and Omidshafiei et al. (2017)

communication between agents (Sect. 5.2) or building models (Sect. 5.3). However, we discuss these topics separately in the respective sections.

Experience replay mechanism Recent successes with reinforcement learning methods such as deep Q-networks (Mnih et al. 2015) rest upon an experience replay mechanism. However, it is not straightforward to employ experience replays to the multi-agent setting because past experience becomes obsolete with the adaption of agent policies over time. To encounter this, Foerster et al. (2017) proposed two approaches. First, they decay outdated transition samples from the replay memory to stabilize targets and then use importance sampling to incorporate off-policy samples. Since the agents' policies are known during the training, off-policy updates can be corrected with importance-weighted policy likelihoods. Second, the state space of each agent is enhanced with estimates of the other agents' policies, so-called fingerprints[5], to prevent non-stationarity. The value functions can then be conditioned on a fingerprint, which clears the age of data sampled from the replay memory. Another extension for experience replays was proposed by Palmer et al. (2018) who applied leniency to every stored transition sample. Leniency associates each sample of the experience memory with a temperature value, which gradually decays by the number of state-action pair visits. Further utilization of the experience replay mechanism to cope with non-stationarity can be found in Tang et al. (2018) and Zheng et al. (2018a). Nevertheless, if the contemporary dynamics of the learners are neglected, algorithms can utilize short-term buffers as applied in Baker et al. (2020) and Leibo et al. (2017).

Centralized Training Scheme As already discussed in Sect. 3.2, the CTDE paradigm can be leveraged to share mutual information between learners to ease training. The availability of information during the training can loosen the non-stationarity of the environment since agents are augmented with information about others. One approach is to enhance *actor-critic* methods with centralized critics over which mutual information is shared between agents during the training (Bono et al. 2019; Iqbal and Sha 2019; Wei et al. 2018). Lowe et al. (2017) embedded each agent with one centralized critic that is augmented with all agents' observations and actions. Based on this additional information, agents face a stationary environment during the training while acting decentralized on local observations at test time. Next to the equipment of one critic per agent, all agents can share one global centralized critic. Foerster et al. (2018b) applied one centralized critic conditioned on the joint action and observations of all agents. The critic computes an agent's individual advantage through estimating the value of the joint action based on a counterfactual baseline, which marginalizes out single agents' influence. Another approach to the CTDE scheme can be seen in *value-based* methods. Rashid et al. (2018) learned a joint action-value function conditioned on the joint observation-action history. The joint action-value function is then divided into agent individual value functions based on monotonic non-linear composition. Foerster et al. (2016) used action-value functions that share information through a communication channel during the training but then discarded it at test time. Similarly, Jorge et al. (2016) employed communication during training to promote information exchange for optimizing action-value functions.

Meta-Learning Sometimes, it can be useful to learn how to adapt to the behavioral changes of others. This learning-to-learn approach is known as meta-learning (Finn and Levine 2018; Schmidhuber et al. 1996). Recent works in the single-agent domain have shown promising results (Duan et al. 2016; Wang et al. 2016a). Al-Shedivat et al. (2018)

[5] Fingerprints draw their inspiration from Tesauro (2004) who eluded non-stationarity by conditioning each agent's policy on estimates of other agents' policies.

transferred this approach to the multi-agent domain and developed a meta-learning based method to tackle the consecutive adaptation of agents in non-stationary environments. Regarding non-stationarity as a sequence of stationary tasks, agents learn to exploit dependencies between successive tasks and generalize over co-adapting agents at test time. They evaluated the resulting behaviors in a competitive multi-agent setting where agents fight in a simulated physics environment. Meta-learning can also be utilized to construct agent models (Rabinowitz et al. 2018). By learning how to model other agents and make inferences on them, agents learn to predict the other agent's future action sequences. They embedded this principle into how one agent learns to capture the behavioral patterns of other agents efficiently.

5.2 Learning communication

Agents capable of developing communication and language corpora pose one of the vital challenges in machine intelligence (Kirby 2002). Intelligent agents must not only decide on what to communicate but also when and with whom. It is indispensable that the developed language is grounded on a common consensus such that all agents understand the spoken language, including its semantics. The research efforts in learning to communicate have intensified because many pathologies can be overcome by incorporating communication skills into agents, including non-stationarity, coherent coordination among agents, and partial observability. For instance, when an agent knows the actions taken by others, the learning problem becomes stationary again from a single agent's perspective in a fully observable environment. Even partial observability can be loosened by messaging local observations to other participants through communication, which helps compensate for limited knowledge (Goldman and Zilberstein 2004).

The common framework to investigate communication is the dec-POMDP (Oliehoek and Amato 2016) which is a fully cooperative setting where agents perceive partial observations of the environment and try to improve upon an equally-shared reward. In such distributed systems, agents must not only learn how to cooperate but also how to communicate in order to optimize the mutual objective. Early MARL works investigated communication rooted in tabular worlds with limited observability (Kasai et al. 2008). Since the spring of deep learning methods, the research of learning communication has witnessed great attention because advanced computational methods provide new opportunities to study highly complex data.

In the following, we categorize the surveyed literature according to the message addressing. First, we describe the broadcasting scenario where sent messages are received by all agents. Second, we look into works that use targeted messages to decide on the recipients by using an attention mechanism. Third and last, we review communication in networked settings where agents communicate only with their local neighborhood instead of the whole population. Figure 3 shows a schematic illustration of this categorization. Another taxonomy may be based on the discrete or continuous nature of messages and the frequency of passed messages.

Broadcasting Messages are addressed to all participants of the communication channel. Foerster et al. (2016) studied how agents learn *discrete* communication protocols in dec-POMDPs in order to accomplish a fully-cooperative task. Being in a CTDE setting, the communication is not restricted during the training but bandwidth-limited at test time. To discover meaningful communication protocols, they proposed two methods. The first, reinforced inter-agent learning (RIAL), is based on deep recurrent Q-networks combined with

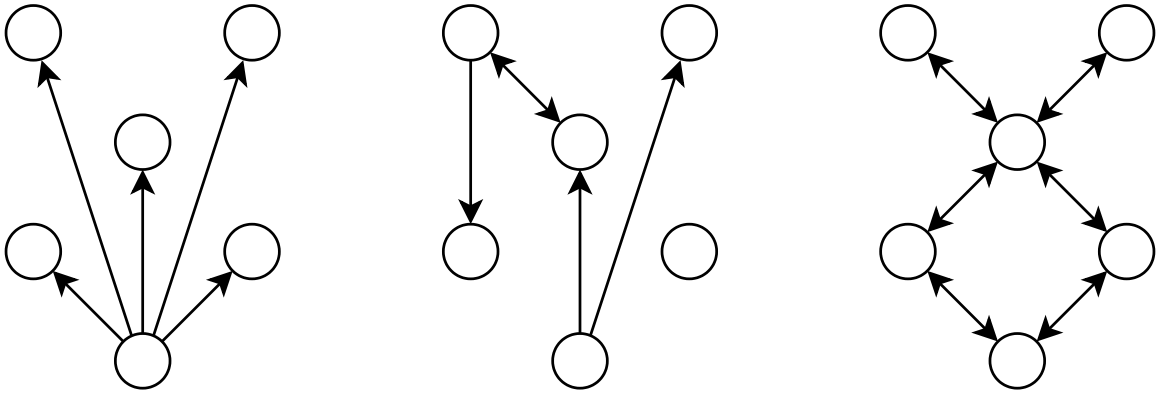


Fig. 3 Schematic illustration of communication types. Unilateral arrows represent unidirectional messages, while bilateral arrows symbolize bidirectional message passing. (Left) In broadcasting, messages are sent to all participants of the communication channel. For better visualization, the broadcasting of only one agent is illustrated but each agent can broadcast messages to all other agents. (Middle) Agents can target the communication through an attention mechanism that determines when, what and with whom to communicate. (Right) Networked communication describes the local connection to neighborhood agents

independent Q-learning where each agent learns an action-value function conditioned on the observation history as well as messages from other agents. Additionally, they applied parameter sharing so that all agents share and update common features from only one Q-network. The second method, differentiable inter-agent learning (DIAL), combines the centralized learning paradigm with deep Q-networks. Messages are delivered over discrete connections, which are based on a relaxation to become differentiable. In contrast, Sukhbaatar et al. (2016) proposed CommNet as an architecture that allows the learning of communication between agents purely based on *continuous* protocols. They showed that each agent learns the joint-action and a sparse communication protocol that encodes meaningful information. The authors emphasized that the decreased observability of vicious states encourages the importance of communication between agents. To foster scalable communication protocols that also facilitate heterogeneous agents, Peng et al. (2017) introduced the bidirectionally-coordinated network (BiCNet) where agents learn in a vectorized actor-critic framework to communicate. Through communication, they were able to coordinate heterogeneous agents in a combat game of StarCraft.

Targeted communication When agents are endowed with targeted communication protocols, they utilize an attention mechanism to determine when, what and with whom to communicate. Jiang and Lu (2018) introduced ATOC as an attentional communication model that enables agents to send messages dynamically and selectively so that communication takes place among a group of agents only when required. They argued that attention is essential for large-scale settings because agents learn to decide which information is most useful for decision-making. Selective communication is the reason why ATOC outperforms CommNet and BiCNet on the conducted navigation tasks. A similar conclusion was drawn by Hoshen (2017) who introduced the vertex attention interaction network (VAIN) as an extension to the CommNet. The baseline approach is extended with an attention mechanism that increases performance due to the focus on only relevant agents. The work by Das et al. (2019) introduced targeted multi-agent communication (TarMAC) that uses attention to decide with whom and what to communicate by actively addressing other agents for message passing. Jain et al. (2019) proposed TBONE for visual navigation in cooperative tasks. In contrast to former works, which are limited to the fully-cooperative setting, Singh et al. (2019) considered mixed settings where each agent owns an individual

reward function. They proposed the individualized controlled continuous communication model (IC3Net), where agents learn when to exchange information using a gating mechanism that blocks incoming communication requests if necessary.

Networked communication Another form of communication is a networked communication protocol where agents can exchange information with their neighborhood (Nedic and Ozdaglar 2009; Zhang et al. 2018). Agents act decentralized based on local observations and received messages from network neighbors. Zhang et al. (2018) used an actor-critic framework where agents share their critic information with their network neighbors to promote global optimality. Chu et al. (2020) introduced the neural communication protocol (NeurComm) to enhance communication efficiency by reducing queue length and intersection delay. Further, they showed that a spatial discount factor could stabilize training when only the local vicinity is regarded to perform policy updates. For theoretical contributions, one may consider the works of Qu et al. (2020), Zhang et al. (2018) and Zhang et al. (2019) whereas the paper of Chu et al. (2020) provides an application perspective in the domain of traffic light control.

Extensions Further methods approach the improvement of coordination skills by applying intrinsic motivation (Jaques et al. 2018, 2019), by making the communication protocol more robust or scalable (Kim et al. 2019; Singh et al. 2019), and maximizing the utility of the communication through efficient encoding (Celikyilmaz et al. 2018; Li et al. 2019b; Wang et al. 2020c).

The above-reviewed papers focus on new methodologies about communication protocols. Besides that, a bulk of literature considers the analysis of emergent language and the occurrence of agent behavior, which we discuss in Sect. 4.2.

5.3 Coordination

Successful coordination in multi-agent systems requires agents to agree on a consensus (Wei Ren et al. 2005). In particular, accomplishing a joint goal in cooperative settings demands a coherent action selection such that the joint action optimizes the mutual task performance. Cooperation among agents is complicated when stochasticity is present in system transitions and rewards or when agents observe only partial information of the environment's state. Mis-coordination may arise in the form of action shadowing when exploratory behavior influences the other agents' search space during learning and, as a result, sub-optimal solutions are found.

Therefore, the agreement upon a mutual consensus necessitates the sharing and collection of information about other agents to derive optimal decisions. Finding such a consensus in the decision-making may happen explicitly through communication or implicitly by constructing models of other agents. The former requires skills to communicate with others so that agents can express their purpose and align their coordination. For the latter, agents need the ability to observe other agents' behavior and reason about their strategies to build a model. If the prediction model is accurate, an agent can learn the other agents' behavioral patterns and direct actions towards a consensus, leading to coordinated behavior. Besides explicit communication and constructing agent models, the CTDE scheme can be leveraged to build different levels of abstraction, which are applied to learn high-level coordination while independent skills are trained at low-level.

In the remainder of this section, we focus on methods that solve coordination issues without establishing communication protocols between agents. Although communication may ease coordination, we discuss this topic separately in Sect. 5.2.

Independent learners The naïve approach to handle multi-agent problems is to regard each agent individually such that other agents are perceived as part of the environment and, thus, are neglected during learning. Opposed to joint action learners, where agents experience the selected actions of others a-posteriori, independently learning agents face the main difficulty of coherently choosing actions such that the joint action becomes optimal concerning the mutual goal (Matignon et al. 2012b). During the learning of good policies, agents influence each other's search space, which can lead to *action shadowing*. The notion of coordination among several autonomously and independently acting agents enjoys a long record, and a bulk of research was conducted in settings with non-communicative agents (Fulda and Ventura 2007; Matignon et al. 2012b). Early works investigated the convergence of independent learners and showed that the convergence to solutions is feasible under certain conditions in deterministic games but fails in stochastic environments (Claus and Boutilier 1998; Lauer and Riedmiller 2000). Stochasticity, relative over-generalization, and other pathologies such as non-stationarity and the alter-exploration problem led to new branches of research including hysteretic learning (Matignon et al. 2007) and leniency (Potter and De Jong 1994). *Hysteretic* Q-learning was introduced to encounter the over-estimation of the value function evoked by stochasticity. Two learning rates are used to increase and decrease the value function updates while relying on an optimistic form of learning. A modern approach to hysteretic learning can be seen in Palmer et al. (2018) and Omidshafiei et al. (2017). An alternative method to adjust the degree of applied optimism during learning is *leniency* (Panait et al. 2006; Wei and Luke 2016). Leniency associates selected actions with decaying temperature values that govern the amount of applied leniency. Agents are optimistic during the early phase when exploration is still high but become less lenient for frequently visited state-action pairs over the training so that value estimations become more accurate towards the end of learning.

Further works expanded independent learners with enhanced techniques to cope with the MARL pathologies mentioned above. Extensions to the deep Q-network can be seen in additional mechanisms used for the experience replay (Palmer et al. 2019), the utilization of specialized estimators (Zheng et al. 2018a) and the use of implicit quantile networks (Lyu and Amato 2020). Further literature investigated independent learners as benchmark reference but reported limited success in cooperative tasks of various domains when no other techniques are applied to alleviate the issue of independent learners (Foerster et al. 2018b; Sunehag et al. 2018).

Constructing models An implicit way to achieve coordination among agents is to capture the behavior of others by constructing models. Models are functions that take past interaction data as input and output predictions about the agents of interest. This can be very important to render the learning process robust against the decision-making of other agents in the environment (Hu and Wellman 1998). The constructed models and the predicted behavior vary widely depending on the approaches and the assumptions being made (Albrecht and Stone 2018).

One of the first works based on deep learning methods was conducted by He et al. (2016) in an adversarial setting. They proposed an architecture that utilizes two neural networks. One neural network captures the opponents' strategies, and the second network estimates the opponents' Q-values. These networks jointly learn models of opponents by encoding observations into a deep Q-network. Another work by Foerster et al. (2018a) introduced a learning method where the policy updates rely on the impact on other agents. The opponent's policy parameters can be inferred from the observed trajectory by using a maximum likelihood technique. The arising non-stationarity is tackled by accounting only recent data. An additional possibility is to address the information gain about other agents

through Bayesian methods. Raileanu et al. (2018) employed a model where agents estimate the other agents' hidden states and embed these estimations into their own policy. Inferring other agents' hidden states from their behavior allows them to choose appropriate actions and promotes eventual coordination. Foerster et al. (2019) used all publicly available observations in the environment to calculate a public belief over agents' local information. Another work by Yang et al. (2018a) used Bayesian techniques to detect opponent strategies in competitive games. A particular challenge is to learn agent models in the presence of fast adapting agents, which amplifies the problem of non-stationarity. As a countermeasure, Everett and Roberts (2018) proposed the switching agent model (SAM), which learns a set of opponent models and a switching mechanism between models. By tracking and detecting the behavioral adaption of other agents, the switching mechanism learns to select the best response from the learned set of opponent models and, thus, showed superior performance over single model learners.

Further works on constructing models can be found in cooperative tasks (Barde et al. 2019; Tacchetti et al. 2019; Zheng et al. 2018b) with imitation learning (Grover et al. 2018; Le et al. 2017), in social dilemmas (Jaques et al. 2019; Letcher et al. 2019), and by predicting behaviors from observations (Hong et al. 2017; Hoshen 2017). For a comprehensive survey on constructing models in multi-agent systems, one may consider the work of Albrecht and Stone (2018).

Besides resolving the coordination problem, building models of other agents can cope with the non-stationarity in the environment. As soon as one agent has knowledge about others' behavior, previously unexplainable transition dynamics can be attributed to the responsible agents, and the environment becomes stationary again from the viewpoint of an individual agent.

Hierarchical methods Learning to coordinate can be challenging if multiple decision-makers are involved due to the increasing complexity (Bernstein et al. 2002). An approach to deal with the coordination problem is by abstracting low-level coordination to higher levels. The idea originated in the single-agent domain where hierarchies for temporal abstraction are employed to ease long-term reward assignments (Dayan and Hinton 1993; Sutton et al. 1999). Lower levels entail only partial information of the higher levels so that the learning task becomes simpler the lower the level of abstraction. First attempts for hierarchical multi-agent RL can be found in the tabular case (Ghavamzadeh et al. 2006; Makar et al. 2001). A deep approach was proposed by Kumar et al. (2017), where a higher-level controller guides the information exchange between decentralized agents. Grounded on the high-level controller, the agents communicate with only one other agent at each time step, which allows the exploration of distributed policies. Another work by Han et al. (2019) is built upon the options framework (Sutton et al. 1999) where they embedded a dynamic termination criterion for Q-learning. By adding a termination criterion, agents could flexibly quit the option execution and react to the behavioral changes of other agents. Related to the idea of feudal networks (Dayan and Hinton 1993), Ahilan and Dayan (2019) applied a two-level abstraction of agents to a cooperative multi-agent setting where, in contrast to other methods, the hierarchy relied on rewards instead of state goals. They showed that this approach could be well suited for decentralized control problems. Jaderberg et al. (2019) used hierarchical representations that allowed agents to reason at different time scales. The authors demonstrated that agents are capable of solving mixed cooperative and competitive tasks in simulated physics environments. Another work by Lee et al. (2020) proposed a hierarchical method to coordinate two agents on robotic manipulation and locomotion tasks to accomplish collaboration such as object pick and placement. They learned primitive skills on the low-level, which are guided by a higher-level policy. Further works

cover hierarchical methods in cooperation tasks (Cai et al. 2013; Ma and Wu 2020; Tang et al. 2018) or social dilemmas (Vezhnevets et al. 2019). An open challenge for hierarchical methods is the autonomous creation and discovery of abstract goals from data (Schaul et al. 2015; Vezhnevets et al. 2017).

5.4 Credit assignment problem

In the fully-cooperative setting, agents are encouraged to maximize an equally-shared reward signal. Even in a fully-observable state space, it is difficult to determine which agents and actions contributed to the eventual reward outcome when agents do not have access to the joint action. Claus and Boutilier (1998) showed that independent learners could not differentiate between the teammate's exploration and the stochasticity in the environment even in a simple bi-matrix game. This can render the learning problem difficult because agents should be ideally provided with feedback corresponding to the task performance to enable sufficient learning. Associating rewards to agents is known as the *credit assignment problem* (Weiß 1995; Wolpert and Tumer 1999). This problem is intensified by the sequential nature of reinforcement learning where agents must understand not only the impact of single actions but also the entire action sequences that eventually lead to the reward outcome (Sen and Weiss 1999). An additional challenge arises when agents have only access to local observations of the environment, which we discuss in Sect. 5.6. In the remainder of this section, we consider three actively investigated approaches that deal with how to determine the contribution of agents jointly-shared reward settings.

Decomposition Early works approached the credit assignment problem by applying filters (Chang et al. 2004) or modifying the reward function such as reward shaping (Ng et al. 1999). Recent approaches focus on exploiting dependencies between agents to decompose the reward among the agents with respect to their actual contribution towards the global reward (Kok and Vlassis 2006). The learning problem is simplified by dividing the task into smaller and, hence, easier sub-problems through decomposition. Sunehag et al. (2018) introduced the value decomposition network (VDN) which factorizes the joint action-value function into a linear combination of individual action-value functions. The VDN learns how to optimally assign an individual reward according to the agent's performance. The neural network helps to disambiguate the joint reward signal concerning the impact of the agent. Rashid et al. (2018) proposed QMIX as an improvement over VDN. QMIX learns a centralized action-value function that is decomposed into agent individual action-value functions through non-linear combinations. Under the assumption of monotonic relationships between the centralized Q-function and the individual Q-functions, decentralized policies can be extracted by individual argmax operations. As an advancement over both VDN and QMIX, Son et al. (2019) proposed QTRAN, which discards the assumption of linearity and monotonicity in the factorization and allows any non-linear combination of value functions. Further approaches about the factorization of value functions can be found in Castellini et al. (2019), Chen et al. (2018), Nguyen et al. (2017b), Wang et al. (2020a), Wang et al. (2020c) and Yang et al. (2018b).

Marginalization Next to the decomposition into simpler sub-problems, one can apply an extra function that marginalizes out the effect of agent individual actions. Nguyen et al. (2018) introduced a mean collective actor-critic framework which marginalizes out the actions of agents by using an approximation of the critic and reduces the variance of the gradient estimation. Similarly, Foerster et al. (2018b) marginalized out the individual actions of agents by applying a counterfactual baseline function. The counterfactual

baseline function uses a centralized critic, which calculates the advantage of a single agent by comparing the estimated return of the current joint-action to the counterfactual baseline. The impact of a single agent's action is determined and can be attributed to the agent itself. Another work by Wu et al. (2018) used a marginalized action-value function as a baseline to reduce the variance of critic estimates. The marginalization approaches are closely related to the *difference rewards* proposed by Tumer and Wolpert (2004) who determine the impact of an agent's individual action compared to the average reward of all agents.

Inverse reinforcement learning Credit assignment problems can be evoked by a bad design of the reinforcement learning problem. Misinterpretations of the agents can lead to failure because unintentional strategies are explored, e.g. if the reward function does not capture all important aspects of the underlying task (Amodei et al. 2016). Therefore, an important step in the problem design is the reward function. However, designing a reward function can be challenging for complex problems (Hadfield-Menell et al. 2017) and becomes even more complicated for multi-agent systems since different agents may accomplish different goals. Another approach to address the credit assignment problem is by *inverse reinforcement learning* (Ng and Russell 2000) that describes how an agent learns a reward function that explains the demonstrated behavior of an expert without having access to the reward signal. The learned reward function can then be used to build strategies. The work of Lin et al. (2018) applied the principle of inverse reinforcement learning to the multi-agent setting. They showed that multiple agents could recover reward functions that are correlated with the ground truths. Related to inverse RL, imitation learning can be used to learn from expert knowledge. Yu et al. (2019) imitated expert behaviors to learn high-dimensional policies in both cooperative and competitive environments. They were able to recover the expert policies for each individual agent from the provided expert demonstrations. Further works on imitation learning consider the fully cooperative setting (Barrett et al. 2017; Le et al. 2017) and Markov Games with mixed settings (Song et al. 2018).

5.5 Scalability

Training a large number of agents is inherently difficult. Every agent involved in the environment adds extra complexity to the learning problem such that the computational effort grows exponentially by the number of agents. Besides complexity concerns, sufficient scaling also demands agents to be robust towards the behavioral adaption of other agents. However, agents can leverage the benefit of distributed knowledge shared and reused between agents to accelerate the learning process. In the following, we review approaches that address the handling of many agents and discuss possible solutions. We broadly classify the surveyed works into those that apply some form of knowledge reuse, reduce the complexity of the learning problem, and develop robustness against the policy adaptions of other agents.

Knowledge reuse The training of individual learning models does scale poorly with the increasing number of agents because the computational effort increases due to the combinatorial possibilities. Knowledge reuse strategies are employed to ease the learning process and scale RL to complex problems by reutilizing previous knowledge into new tasks. Knowledge reuse can be applied in many facets (Silva et al. 2018).

First, agents can make use of a *parameter sharing* technique if they exhibit homogeneous structures, e.g. the weights in a neural network for sharing parts or the whole learning model with others. Sharing the parameters of a policy enables an efficient training process that can scale up to an arbitrary number of agents and, thus, can boost the learning

process (Gupta et al. 2017). Parameter sharing has proven to be useful in various applications such as learning to communicate (Foerster et al. 2016; Jiang and Lu 2018; Peng et al. 2017; Sukhbaatar et al. 2016), modeling agents (Hernandez-Leal et al. 2019), and in partially observable cooperative games (Sunehag et al. 2018). For a discussion on different parameter sharing strategies, one may consider the paper by Chu and Ye (2017).

As the second approach, knowledge reuse can be applied in form of *transfer learning* (Da Silva et al. 2019; Da Silva and Costa 2019). Experience obtained in learning to perform one task may also improve the performance in a related but different task (Taylor and Stone 2009). Da Silva and Costa (2017) used a knowledge database from which an agent can extract previous solutions of related tasks and embed such information into the current task's training. Likewise, Da Silva et al. (2017) applied expert demonstrations where the agents take the role of students that ask a teacher for advice. They demonstrated that simultaneously learning agents could advise each other through knowledge transfer. Further works on transfer learning can be found in the cooperative multi-agent setting (Omidshafiei et al. 2019) and in natural language applications (Luketina et al. 2019). In general multi-agent systems, the works of (Boutsioukis et al. 2012; Taylor et al. 2013) substantiate that transfer learning can speed up the learning process.

Besides parameter sharing and transfer learning, *curriculum learning* may be applied for the scaling to many agents. Since tasks become more challenging to master and more time consuming to train as the number of agents increases, it is often challenging to learn from scratch. Curriculum learning starts with a small number of agents and then gradually enlarges the number of agents over the training course. Through the steady increase within the curriculum, trained policies can perform better than without a curriculum (Gupta et al. 2017; Long et al. 2020; Narvekar et al. 2016). Curriculum learning schemes can also cause improved generalization and faster convergence of agent policies (Bengio et al. 2009). Further works show that agents can generate learning curricula automatically (Sukhbaatar et al. 2017; Svetlik et al. 2017) or can create arms races in competitive settings (Baker et al. 2020).

Complexity reduction Many real-world applications naturally encompass large numbers of simultaneously interacting agents (Nguyen et al. 2017a, b). As the quantity of agents increases, the requirement to contain the curse of dimensionality becomes inevitable. Yang et al. (2018b) addressed the issue of scalability with a mean-field method. The interactions between large numbers of agents are estimated by the impact of a single agent compared to the mean impact of the whole or local agent population. The complexity reduces as the problem is broken down into pairwise interactions between an agent and its neighborhood. Regarding the average effect to its neighbors, each agent learns the best response towards its proximity. Another approach to constrain the explosion in complexity is by factorizing the problem into smaller sub-problems (Guestrin et al. 2002). Chen et al. (2018) decomposed the joint action-value function into independent components and used pairwise interactions between agents to render large-scale problems computationally tractable. Further works studied large-scale MADRL problems with graphical models (Nguyen et al. 2017a) and the CTDE paradigm (Lin et al. 2018).

Robustness Another desired property is the *robustness* of learned policies to perturbations in the environment caused by other agents. Perturbations are fortified by the number of agents and the resulting growth of the state-action space. In supervised learning, a common problem is that models can over-fit to the data set. Similarly, over-fitting can occur in RL frameworks if environments provide little or no deviation (Bansal et al. 2018). To maintain robustness over the training process and to the other agents' adaption, several methods have been proposed.

First, *regularization* techniques can be used to prevent over-fitting to other agents' behavior. Examples can be seen in policies ensembles (Lowe et al. 2017), where a collection of different sub-policies is trained for each agent, or can be found in best responses to policy mixtures (Lanctot et al. 2017).

Second, *adversarial training* can be applied to mitigate the vulnerability of polices towards perturbations. Pinto et al. (2017) added an adversarial agent to the environment that applied targeted disturbances to the learning process. By hampering the training, the agents were compelled to encounter these disturbances and develop robust policies. Similarly, Li et al. (2019a) used an adversarial setting to reduce the sensitivity of agents towards the environment. Bansal et al. (2018) demonstrated that policies, which are trained in a competitive setting, could yield behaviors that are far more complex than the environment itself. From an application perspective, Spooner and Savani (2020) studied robust decision-making in market making.

The observations from above are in accordance with the findings of related studies about the impact of *self-play* (Raghu et al. 2018; Sukhbaatar et al. 2017). Heinrich and Silver (2016) used self-play to learn approximate Nash equilibria of imperfect-information games and showed that self-play could be used to obtain better robustness in the learned policies. Similarly, self-play was used to compete with older versions of policies to render the learned behaviors more robust (Baker et al. 2020; Berner et al. 2019; Silver et al. 2018). Silver et al. (2016) adapted self-play as a regularization technique to prevent the policy network from over-fitting by playing against older versions of itself. However, Gleave et al. (2020) studied the existence of adversarial policies in competitive games and showed that complex policies could be fooled by comparably easy strategies. Although agents trained through self-play proved to be more robust, allegedly random and uncoordinated strategies caused agents to fail at the task. They argued that the vulnerability towards adversarial attacks increases with the dimensionality of the observation space. A further research direction for addressing robustness is to render the learning representation invariant towards permutations, as shown in Liu et al. (2020).

5.6 Partial observability

Outside an idealized setting, agents neither can observe the global state of the environment, nor do they have access to the internal knowledge of other agents. By perceiving only partial observations, a single observation does not capture all relevant information about the environment and its history. Hence, the Markov property is not fulfilled, and the environment appears non-Markovian. An additional difficulty elicited by partial observability is the *lazy agent problem* which can occur in cooperative settings (Sunehag et al. 2018). As introduced in Sect. 2.2, the common frameworks that deal with partial observability are POMPDs for general settings and dec-POMDPs for cooperative settings with a shared reward function. Dec-POMDPs are computationally challenging (Bernstein et al. 2002) and still intractable when solving problems with real-world complexity (Amato et al. 2015). However, recent work accomplished promising results in video games with imperfect information (Baker et al. 2020; Berner et al. 2019; Jaderberg et al. 2019; Vinyals et al. 2019).

A natural way to deal with non-Markovian environments is through information exchange between the decision-makers (Goldman and Zilberstein 2004). Agents that are able to communicate can compensate for their limited knowledge by propagating information and fill the lack of knowledge about other agents or the environment (Foerster et al.

2016). As we already discussed in Sect. 5.2, there are several ways to incorporate communication capabilities into agents. A primary example is Jiang and Lu (2018) who used an attention mechanism to establish communication under partial observations. Rather than having a fixed frequency for the information exchange, they learned to communicate on-demand. Further approaches under partial observability have been investigated in cooperative tasks (Das et al. 2019; Sukhbaatar et al. 2016) or mixed settings (Singh et al. 2019).

In the following, we review papers that cope with partial observability by incorporating a memory mechanism. Agents, which have the capability of memorizing past experiences, can compensate for the lack of information.

Memory mechanism A common way to tackle partial observability is the usage of deep recurrent neural networks, which equip agents with a memory mechanism to store information that can be relevant in the future (Hausknecht and Stone 2015). However, long-term dependencies render the decision-making difficult since experiences that were observed in the further past may have been forgotten (Hochreiter and Schmidhuber 1997). Approaches involving recurrent neural networks to deal with partial observability can be realized with value-based approaches (Omidshafiei et al. 2017) or actor-critic methods (Dibangoye and Buffet 2018; Foerster et al. 2018b; Gupta et al. 2017). Foerster et al. (2019) used a Bayesian method to tackle partial observability in cooperative settings. They used all publicly available features of the environment and agents to determine a public belief over the agents' internal states. A severe concern in MADRL is that the memorization of past information is exacerbated by the number of agents involved during the learning process.

6 Discussion

In this section, we discuss findings from previous sections. We enumerate trends that we have identified in recent literature. Since these trends are useful for addressing current challenges, they may also be an avenue for upcoming research. To the end of our discussion, we point out possible future work. We elaborate on problems where only a minority of research has been conducted and pose two problems which we find the toughest ones to overcome.

Despite the recent advances in many directions, many pathologies such as relative overgeneralization combined with reward stochasticity are not yet solved, even in allegedly simple tabular worlds. MADRL has taken profit from the history of MARL by scaling up the insights to more complex problems. Approaches where strong solutions exist in simplified MARL settings may be transferable to the MADRL domain. Thus by enhancing older methods with new deep learning approaches, unsolved problems and concepts from MARL continue to matter in MADRL. An essential point for MADRL is that reproducibility is taken conscientiously. Well-known papers from the single-agent domain underline the significance of hyper-parameters, the number of independent random seeds, and chosen code-base towards the eventual task performance (Henderson et al. 2018; Islam et al. 2017). To maintain steady progress, the reporting of all used hyper-parameters and a transparent conduction of experiments is crucial. We want to make the community aware that these findings may also be valid for the multi-agent domain. Therefore, it is inevitable that standardized frameworks are created in which different algorithms can be compared along with their merits and demerits. Many individual environments have been proposed which exhibit intricate structure and real-world complexity (Baker et al. 2020; Beattie et al. 2016; Johnson et al. 2016; Juliani et al. 2018; Song et al. 2019; Vinyals et al. 2017). However,

Table 4 Our identified trends in MADRL and the addressed challenges

Trend	Addressed challenge(s)
Curriculum learning	Scalability
Memory	Non-stationarity, partial observability
Communication	Non-stationarity, coordination, partial observability
CTDE	Non-stationarity, coordination, partial observability, credit assignment, scalability

no consistent benchmark yet exists that provides a unified interface and allows a fair comparison between different kinds of algorithms grounded on a great variety of tasks like the *OpenAI Gym* (Brockman et al. 2016) for single-agent problems.

6.1 Trends

Over the last years, approaches in the multi-agent domain achieved successes based on recurring patterns of good practice. We have identified four trends in state-of-the-art literature that have been frequently applied to address current challenges (Table 4).

As the first trend, we observe curriculum learning as an approach to divide the learning process into stages to deal with scalability issues. By starting with a small quantity, the number of agents is gradually enlarged over the learning course so that large-scale training becomes feasible (Gupta et al. 2017; Long et al. 2020; Narvekar et al. 2016). Alternatively, curricula can also be employed to create different stages of difficulty, where agents face relatively easy tasks at the beginning and gradually more complex tasks as their skills increase (Vinyals et al. 2019). Besides that, curriculum training is used to investigate the emergence of agent behavior. Curricula describe engineered changes in the dynamics of the environment. Agents adapt their behaviors over time in response to the strategic changes of others, which can yield arms races between agents. This process of continual adaption is referred to *autocurricula* (Leibo et al. 2019), which have been reported in several works (Baker et al. 2020; Sukhbaatar et al. 2017; Svetlik et al. 2017).

Second, we recognize a trend towards deep neural networks embedded with recurrent units to memorize experience. By having the ability to track the history of state transitions and the decisions of other agents, the non-stationarity of the environment due to multiple decision-makers and partially observable states can be addressed in small problems (Omidshafiei et al. 2017), and can be managed sufficiently well in complex problems (Baker et al. 2020; Berner et al. 2019; Jaderberg et al. 2019).

Third, an active line of research is exploring the development of communication skills. Due to the rise of deep learning methods, new computational approaches are available to investigate the emergence of language between interactive agents (Lazaridou and Baroni 2020). Despite the plethora of works that analyze emergent behaviors and semantics, many works propose methods that endow agents with communication skills. By expressing their intension, agents can align their coordination and find a consensus (Foerster et al. 2016). The non-stationarity from the perspective of a single learner can be eluded when agents disclose their history. Moreover, agents can share their local information with others to alleviate partial observability (Foerster et al. 2018b; Omidshafiei et al. 2017).

Fourth and last, we note a clear trend towards the CTDE paradigm that enables the sharing of information during the training. Local information such as the observation-action history, function values, or policies can be made available to all agents during the training, which renders the environment stationary from the viewpoint of an individual agent and may diminish partial observability (Lowe et al. 2017). Further, the credit assignment problem can be addressed when information is available about all agents, and a centralized mechanism can attribute the individual contribution to the respective agent (Foerster et al. 2018b). Further challenges that can be loosened are coordination and scalability when the lack of information of an individual agent is compensated, and the learning process is accelerated (Gupta et al. 2017).

6.2 Future work

Next to our identified trends, which are already under active research, we recognize areas that have not been sufficiently explored yet. One such area is *multi-goal learning* where each agent has an individually associated goal that needs to be optimized. However, global optimality can only be accomplished if agents also allow others to be successful in their task (Yang et al. 2020). Typical scenarios are cooperative tasks such as public good dilemmas, where agents are obliged to the sustainable use of limited resources, or autonomous driving, where agents have individual destinations and are supposed to coordinate the path-finding to avoid crashes. A similar direction is *multi-task* learning where agents are expected to perform well not only on one single but also on related other tasks (Omidshafiei et al. 2017; Taylor and Stone 2009). Besides multi-goal and multi-task learning, another avenue for future work is present in *safe* MADRL. Safety is a highly desired property because autonomously acting agents are expected to ensure system performance while holding to safety guarantees during learning and employment (García et al. 2015). Several works in single-agent RL are concerned with safety concepts, but its applicability to multiple agents is limited and still in its infancy (Zhang and Bastani 2019; Zhu et al. 2020). Akin to the growing interest in learning to communicate, a similar effect may happen in the multi-agent domain, where deep learning methods open new paths. For an application perspective on safe autonomous driving, one can consider the article by Shalev-Shwartz et al. (2016). Another possible direction for future research offers the intersection between MADRL and *evolutionary* methodologies. Evolutionary algorithms have been used in versatile contexts of multi-agent RL, e.g. for building intrinsic motivation (Wang et al. 2019), shaping rewards (Jaderberg et al. 2019), generating curricula (Long et al. 2020) and analyzing dynamics (Bloembergen et al. 2015). Since evolution requires many entities to adapt, multi-agent RL is a natural playground for such algorithms.

Beyond the current challenges and reviewed literature of Sect. 5, we identify two problems that we regard as the most challenging problems to overcome by future work. We primarily choose these two problems since they are the ones that matter the most when it comes to the applicability of algorithms to real-world scenarios. Most research focuses on learning within homogeneous settings where agents share common interests and optimize a mutual goal. For instance, the learning of communication is mainly studied in dec-POMDPs, where agents are expected to optimize upon a joint reward signal. When agents share common interests, the CTDE paradigm is usually a beneficial choice to exchange information between agents, and problems like non-stationarity, partial observability, and coordination can be diminished. However, *heterogeneity* implies that agents may have their own interests and goals, individual experience and knowledge, or different skills and

capabilities. Limited research has been conducted in heterogeneous scenarios, although many real-world problems naturally comprise a mixture of different entities. Under real-world conditions, agents have only access to local and heterogeneous information on which decisions must be taken. The fundamental problem in the multi-agent domain is and ever has been the *curse of dimensionality* (Busoniu et al. 2008; Hernandez-Leal et al. 2019). The state-action space and the combinatorial possibilities of agent interactions increase exponentially by the number of agents, which renders sufficient exploration itself a difficult problem. This is intensified when agents have only access to partial observations of the environment or when the environment is of continuous nature. Although powerful function approximators like neural networks can cope with continuous spaces and generalize well over large spaces, open questions remain like how to explore large and complex spaces sufficiently well and how to solve large combinatorial optimization problems.

7 Conclusion

Even though multi-agent reinforcement learning enjoys a long record, historical approaches hardly exceeded the complexity of discretized environments with a limited amount of states and actions (Busoniu et al. 2008; Tuyls and Weiss 2012). Since the breakthrough of deep learning methods, the field is undergoing a rapid transformation, and many previously unsolved problems have become step by step tractable. Latest advances showed that tasks with real-world complexity could be mastered (Baker et al. 2020; Berner et al. 2019; Jaderberg et al. 2019; Vinyals et al. 2019). Still, MADRL is a young field which attracts growing interest, and the amount of published literature rises swiftly. In this article, we surveyed recent works that combine deep learning methods with multi-agent reinforcement learning. We analyzed training schemes that are used to learn policies, and we reviewed patterns of agent behavior that emerge when multiple entities interact simultaneously. In addition, we systematically investigated challenges that are present in the multi-agent context and studied recent approaches that are under active research. Finally, we outlined trends which we have identified in state-of-the-art literature and proposed possible avenues for future work. With this contribution, we want to equip interested readers with the necessary tools to understand the contemporary challenges in MADRL by providing a more holistic overview of the recent approaches. We want to emphasize its potential and reveal opportunities as well as its limitations. In the foreseeable future, we expect an abundance of new literature to emanate and, hence, we want to encourage the community for further developments in this interesting and young field of research.

Acknowledgements We would like to thank the editor and the three anonymous reviewers for providing their comprehensive feedback. Without their suggestions, this manuscript would not look as it does in this final version. We want to thank our colleagues and friends who read through earlier versions of this manuscript. In particular, we appreciate the help of Matthias Kissel, Patrick Krämer, Anke Müller and Martin Gottwald.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

Ahilan S, Dayan P (2019) Feudal multi-agent hierarchies for cooperative reinforcement learning. CoRR arxiv: abs/1901.08492

Al-Shedivat M, Bansal T, Burda Y, Sutskever I, Mordatch I, Abbeel P (2018) Continuous adaptation via meta-learning in nonstationary and competitive environments. In: International conference on learning representations. https://openreview.net/forum?id=Sk2u1g-0-

Albrecht SV, Stone P (2018) Autonomous agents modelling other agents: a comprehensive survey and open problems. Artif Intell 258:66–95. https://doi.org/10.1016/j.artint.2018.01.002. http://www.sciencedirect.com/science/article/pii/S0004370218300249

Amato C, Konidaris G, Cruz G, Maynor CA, How JP, Kaelbling LP (2015) Planning for decentralized control of multiple robots under uncertainty. In: 2015 IEEE international conference on robotics and automation (ICRA), pp 1241–1248. https://doi.org/10.1109/ICRA.2015.7139350

Amodei D, Olah C, Steinhardt J, Christiano PF, Schulman J, Mané D (2016) Concrete problems in AI safety. CoRR. arxiv: abs/1606.06565,

Antol S, Agrawal A, Lu J, Mitchell M, Batra D, Lawrence Zitnick C, Parikh D (2015) Vqa: Visual question answering. In: The IEEE international conference on computer vision (ICCV)

Arulkumaran K, Deisenroth MP, Brundage M, Bharath AA (2017) Deep reinforcement learning: a brief survey. IEEE Signal Process Mag 34(6):26–38. https://doi.org/10.1109/MSP.2017.2743240

Aubret A, Matignon L, Hassas S (2019) A survey on intrinsic motivation in reinforcement learning. arXiv e-prints arXiv:1908.06976,

Baker B, Kanitscheider I, Markov T, Wu Y, Powell G, McGrew B, Mordatch I (2020) Emergent tool use from multi-agent autocurricula. In: International conference on learning representations. https://openreview.net/forum?id=SkxpxJBKwS

Bansal T, Pachocki J, Sidor S, Sutskever I, Mordatch I (2018) Emergent complexity via multi-agent competition. In: International conference on learning representations. https://openreview.net/forum?id=Sy0GnUxCb

Barde P, Roy J, Harvey FG, Nowrouzezahrai D, Pal C (2019) Promoting coordination through policy regularization in multi-agent reinforcement learning. arXiv e-prints arXiv:1908.02269,

Barrett S, Rosenfeld A, Kraus S, Stone P (2017) Making friends on the fly: cooperating with new teammates. Artif Intell 242:132–171

Beattie C, Leibo JZ, Teplyashin D, Ward T, Wainwright M, Küttler H, Lefrancq A, Green S, Valdés V, Sadik A, Schrittwieser J, Anderson K, York S, Cant M, Cain A, Bolton A, Gaffney S, King H, Hassabis D, Legg S, Petersen S (2016) Deepmind lab. CoRR. arxiv: abs/1612.03801

Becker R, Zilberstein S, Lesser V, Goldman CV (2004) Solving transition independent decentralized Markov decision processes. J Artif Intell Res 22:423–455

Bellemare M, Srinivasan S, Ostrovski G, Schaul T, Saxton D, Munos R (2016) Unifying count-based exploration and intrinsic motivation. In: Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R (eds) Advances in neural information processing systems 29, Curran Associates, Inc., pp 1471–1479. http://papers.nips.cc/paper/6383-unifying-count-based-exploration-and-intrinsic-motivation.pdf

Bellman R (1957) A Markovian decision process. J Math Mechanics 6(5):679–684. http://www.jstor.org/stable/24900506

Bengio Y, Louradour J, Collobert R, Weston J (2009) Curriculum learning. In: Proceedings of the 26th annual international conference on machine learning, ACM, New York, NY, USA, ICML '09, pp 41–48. https://doi.org/10.1145/1553374.1553380,

Berner C, Brockman G, Chan B, Cheung V, Debiak P, Dennison C, Farhi D, Fischer Q, Hashme S, Hesse C, Józefowicz R, Gray S, Olsson C, Pachocki JW, Petrov M, de Oliveira Pinto HP, Raiman J, Salimans T, Schlatter J, Schneider J, Sidor S, Sutskever I, Tang J, Wolski F, Zhang S (2019) Dota 2 with large scale deep reinforcement learning. ArXiv arxiv: abs/1912.06680

Bernstein DS, Givan R, Immerman N, Zilberstein S (2002) The complexity of decentralized control of Markov decision processes. Math Oper Res 27(4):819–840. https://doi.org/10.1287/moor.27.4.819.297

Bertsekas DP (2012) Dynamic programming and optimal control, vol 2, 4th edn. Athena Scientific, Belmont

Bertsekas DP (2017) Dynamic programming and optimal control, vol 1, 4th edn. Athena Scientific, Belmont

Bloembergen D, Tuyls K, Hennes D, Kaisers M (2015) Evolutionary dynamics of multi-agent learning: a survey. J Artif Intell Res 53:659–697

Bono G, Dibangoye JS, Matignon L, Pereyron F, Simonin O (2019) Cooperative multi-agent policy gradient. In: Berlingerio M, Bonchi F, Gärtner T, Hurley N, Ifrim G (eds) Machine learning and knowledge discovery in databases. Springer International Publishing, Cham, pp 459–476

Boutsioukis G, Partalas I, Vlahavas I (2012) Transfer learning in multi-agent reinforcement learning domains. In: Sanner S, Hutter M (eds) Recent advances in reinforcement learning. Springer, Berlin, pp 249–260

Bowling M, Veloso M (2002) Multiagent learning using a variable learning rate. Artif Intell 136(2):215–250

Brockman G, Cheung V, Pettersson L, Schneider J, Schulman J, Tang J, Zaremba W (2016) Openai gym. arXiv:1606.01540

Busoniu L, Babuska R, De Schutter B (2008) A comprehensive survey of multiagent reinforcement learning. IEEE Trans Syst Man Cybern Part C (Appl Rev) 38(2):156–172. https://doi.org/10.1109/TSMCC.2007.913919

Cai Y, Yang SX, Xu X (2013) A combined hierarchical reinforcement learning based approach for multi-robot cooperative target searching in complex unknown environments. In: 2013 IEEE symposium on adaptive dynamic programming and reinforcement learning (ADPRL), pp 52–59. https://doi.org/10.1109/ADPRL.2013.6614989

Cao K, Lazaridou A, Lanctot M, Leibo JZ, Tuyls K, Clark S (2018) Emergent communication through negotiation. In: International conference on learning representations. https://openreview.net/forum?id=Hk6WhagRW

Cao Y, Yu W, Ren W, Chen G (2013) An overview of recent progress in the study of distributed multi-agent coordination. IEEE Trans Industr Inf 9(1):427–438. https://doi.org/10.1109/TII.2012.2219061

Castellini J, Oliehoek FA, Savani R, Whiteson S (2019) The representational capacity of action-value networks for multi-agent reinforcement learning. In: Proceedings of the 18th international conference on autonomous agents and multiagent systems, international foundation for autonomous agents and multiagent systems, Richland, SC, AAMAS '19, pp 1862–1864. http://dl.acm.org/citation.cfm?id=3306127.3331944

Celikyilmaz A, Bosselut A, He X, Choi Y (2018) Deep communicating agents for abstractive summarization. CoRR arxiv: abs/1803.10357,

Chang Y, Ho T, Kaelbling LP (2004) All learning is local: Multi-agent learning in global reward games. In: Thrun S, Saul LK, Schölkopf B (eds) Advances in neural information processing systems 16, MIT Press, pp 807–814. http://papers.nips.cc/paper/2476-all-learning-is-local-multi-agent-learning-in-global-reward-games.pdf

Chen Y, Zhou M, Wen Y, Yang Y, Su Y, Zhang W, Zhang D, Wang J, Liu H (2018) Factorized q-learning for large-scale multi-agent systems. CoRR arxiv: abs/1809.03738

Chen YF, Liu M, Everett M, How JP (2016) Decentralized non-communicating multiagent collision avoidance with deep reinforcement learning. CoRR. arxiv: abs/1609.07845,

Chentanez N, Barto AG, Singh SP (2005) Intrinsically motivated reinforcement learning. In: Saul LK, Weiss Y, Bottou L (eds) Advances in neural information processing systems 17, MIT Press, pp 1281–1288. http://papers.nips.cc/paper/2552-intrinsically-motivated-reinforcement-learning.pdf

Choi E, Lazaridou A, de Freitas N (2018) Multi-agent compositional communication learning from raw visual input. In: International conference on learning representations. https://openreview.net/forum?id=rknt2Be0-

Chu T, Chinchali S, Katti S (2020) Multi-agent reinforcement learning for networked system control. In: International conference on learning representations. https://openreview.net/forum?id=Syx7A3NFvH

Chu T, Wang J, Codecà L, Li Z (2020) Multi-agent deep reinforcement learning for large-scale traffic signal control. IEEE Trans Intell Transp Syst 21(3):1086–1095

Chu X, Ye H (2017) Parameter sharing deep deterministic policy gradient for cooperative multi-agent reinforcement learning. CoRR arxiv: abs/1710.00336

Claus C, Boutilier C (1998) The dynamics of reinforcement learning in cooperative multiagent systems. In: Proceedings of the fifteenth national conference on artificial intelligence and tenth innovative applications of artificial intelligence conference, AAAI 98, IAAI 98, July 26–30, 1998, Madison, Wisconsin, USA, pp 746–752. http://www.aaai.org/Library/AAAI/1998/aaai98-106.php

Crandall JW, Goodrich MA (2011) Learning to compete, coordinate, and cooperate in repeated games using reinforcement learning. Mach Learn 82(3):281–314. https://doi.org/10.1007/s10994-010-5192-9

Da Silva FL, Costa AHR (2017) Accelerating multiagent reinforcement learning through transfer learning. In: Proceedings of the thirty-first AAAI conference on artificial intelligence, AAAI Press, AAAI'17, pp 5034–5035. http://dl.acm.org/citation.cfm?id=3297863.3297988

Da Silva FL, Costa AHR (2019) A survey on transfer learning for multiagent reinforcement learning systems. J Artif Int Res 64(1):645–703. https://doi.org/10.1613/jair.1.11396

Da Silva FL, Glatt R, Costa AHR (2017) Simultaneously learning and advising in multiagent reinforcement learning. In: Proceedings of the 16th conference on autonomous agents and multiagent systems, international foundation for autonomous agents and multiagent systems, Richland, SC, AAMAS '17, pp 1100–1108. http://dl.acm.org/citation.cfm?id=3091210.3091280

Da Silva FL, Warnell G, Costa AHR, Stone P (2019) Agents teaching agents: a survey on inter-agent transfer learning. Auton Agent Multi-Agent Syst 34(1):9. https://doi.org/10.1007/s10458-019-09430-0

Das A, Kottur S, Moura JMF, Lee S, Batra D (2017) Learning cooperative visual dialog agents with deep reinforcement learning. In: The IEEE international conference on computer vision (ICCV)

Das A, Gervet T, Romoff J, Batra D, Parikh D, Rabbat M, Pineau J (2019) TarMAC: Targeted multi-agent communication. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of the 36th international conference on machine learning, PMLR, Long Beach, California, USA, Proceedings of machine learning research, vol 97, pp 1538–1546. http://proceedings.mlr.press/v97/das19a.html

Dayan P, Hinton GE (1993) Feudal reinforcement learning. In: Hanson SJ, Cowan JD, Giles CL (eds) Advances in neural information processing systems 5, Morgan-Kaufmann, pp 271–278. http://papers.nips.cc/paper/714-feudal-reinforcement-learning.pdf

De Cote EM, Lazaric A, Restelli M (2006) Learning to cooperate in multi-agent social dilemmas. In: Proceedings of the fifth international joint conference on autonomous agents and multiagent systems, ACM, New York, NY, USA, AAMAS '06, pp 783–785. https://doi.org/10.1145/1160633.1160770

Diallo EAO, Sugiyama A, Sugawara T (2017) Learning to coordinate with deep reinforcement learning in doubles pong game. In: 2017 16th IEEE international conference on machine learning and applications (ICMLA), pp 14–19. https://doi.org/10.1109/ICMLA.2017.0-184

Dibangoye J, Buffet O (2018) Learning to act in decentralized partially observable MDPs. In: Dy J, Krause A (eds) Proceedings of the 35th international conference on machine learning, PMLR, Stockholmsmässan, Stockholm Sweden, Proceedings of Machine Learning Research, vol 80, pp 1233–1242. http://proceedings.mlr.press/v80/dibangoye18a.html

Dobbe R, Fridovich-Keil D, Tomlin C (2017) Fully decentralized policies for multi-agent systems: an information theoretic approach. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems 30, Curran Associates, Inc., pp 2941–2950. http://papers.nips.cc/paper/6887-fully-decentralized-policies-for-multi-agent-systems-an-information-theoretic-approach.pdf

Duan Y, Schulman J, Chen X, Bartlett PL, Sutskever I, Abbeel P (2016) Rl: fast reinforcement learning via slow reinforcement learning. CoRR arxiv: abs/1611.02779,

Eccles T, Bachrach Y, Lever G, Lazaridou A, Graepel T (2019) Biases for emergent communication in multi-agent reinforcement learning. In: Wallach H, Larochelle H, Beygelzimer A, Alche-Buc F, Fox E, Garnett R (eds) Advances in neural information processing systems 32, Curran Associates, Inc., pp 13111–13121. http://papers.nips.cc/paper/9470-biases-for-emergent-communication-in-multi-agent-reinforcement-learning.pdf

Everett R, Roberts S (2018) Learning against non-stationary agents with opponent modelling and deep reinforcement learning. In: 2018 AAAI Spring symposium series

Evtimova K, Drozdov A, Kiela D, Cho K (2018) Emergent communication in a multi-modal, multi-step referential game. In: International conference on learning representations. https://openreview.net/forum?id=rJGZq6g0-

Finn C, Levine S (2018) Meta-learning and universality: deep representations and gradient descent can approximate any learning algorithm. In: International conference on learning representations. https://openreview.net/forum?id=HyjC5yWCW

Foerster J, Assael IA, de Freitas N, Whiteson S (2016) Learning to communicate with deep multi-agent reinforcement learning. In: Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R (eds) Advances in neural information processing systems 29, Curran Associates, Inc., pp 2137–2145. http://papers.nips.cc/paper/6042-learning-to-communicate-with-deep-multi-agent-reinforcement-learning.pdf

Foerster J, Nardelli N, Farquhar G, Afouras T, Torr PHS, Kohli P, Whiteson S (2017) Stabilising experience replay for deep multi-agent reinforcement learning. In: Precup D, Teh YW (eds) Proceedings of the 34th international conference on machine learning, PMLR, International Convention Centre, Sydney, Australia, Proceedings of Machine Learning Research, vol 70, pp 1146–1155. http://proceedings.mlr.press/v70/foerster17b.html

Foerster J, Chen RY, Al-Shedivat M, Whiteson S, Abbeel P, Mordatch I (2018a) Learning with opponent-learning awareness. In: Proceedings of the 17th international conference on autonomous agents and multiagent systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, AAMAS '18, pp 122–130. http://dl.acm.org/citation.cfm?id=3237383.3237408

Foerster J, Farquhar G, Afouras T, Nardelli N, Whiteson S (2018b) Counterfactual multi-agent policy gradients. https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17193

Foerster J, Song F, Hughes E, Burch N, Dunning I, Whiteson S, Botvinick M, Bowling M (2019) Bayesian action decoder for deep multi-agent reinforcement learning. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of the 36th international conference on machine learning, PMLR, Long Beach, California, USA, Proceedings of Machine Learning Research, vol 97, pp 1942–1951. http://proceedings.mlr.press/v97/foerster19a.html

Fulda N, Ventura D (2007) Predicting and preventing coordination problems in cooperative q-learning systems. In: Proceedings of the 20th international joint conference on artifical intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'07, pp 780–785

García J, Fern, o Fernández (2015) A comprehensive survey on safe reinforcement learning. J Mach Learn Res 16(42):1437–1480. http://jmlr.org/papers/v16/garcia15a.html

Ghavamzadeh M, Mahadevan S, Makar R (2006) Hierarchical multi-agent reinforcement learning. Auton Agent Multi-Agent Syst. https://doi.org/10.1007/s10458-006-7035-4

Gleave A, Dennis M, Wild C, Kant N, Levine S, Russell S (2020) Adversarial policies: Attacking deep reinforcement learning. In: International conference on learning representations. https://openreview.net/forum?id=HJgEMpVFwB

Goldman CV, Zilberstein S (2004) Decentralized control of cooperative systems: categorization and complexity analysis. J Artif Int Res 22(1):143–174. http://dl.acm.org/citation.cfm?id=1622487.1622493

Grover A, Al-Shedivat M, Gupta J, Burda Y, Edwards H (2018) Learning policy representations in multiagent systems. In: Dy J, Krause A (eds) Proceedings of the 35th international conference on machine learning, PMLR, Stockholmsmässan, Stockholm Sweden, Proceedings of Machine Learning Research, vol 80, pp 1802–1811. http://proceedings.mlr.press/v80/grover18a.html

Guestrin C, Koller D, Parr R (2002) Multiagent planning with factored mdps. In: Dietterich TG, Becker S, Ghahramani Z (eds) Advances in neural information processing systems 14, MIT Press, pp 1523–1530. http://papers.nips.cc/paper/1941-multiagent-planning-with-factored-mdps.pdf

Gupta JK, Egorov M, Kochenderfer M (2017) Cooperative multi-agent control using deep reinforcement learning. In: Sukthankar G, Rodriguez-Aguilar JA (eds) autonomous agents and multiagent systems. Springer, Cham, pp 66–83

Hadfield-Menell D, Milli S, Abbeel P, Russell SJ, Dragan A (2017) Inverse reward design. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems 30, Curran Associates, Inc., pp 6765–6774. http://papers.nips.cc/paper/7253-inverse-reward-design.pdf

Han D, Boehmer W, Wooldridge M, Rogers A (2019) Multi-agent hierarchical reinforcement learning with dynamic termination. In: Proceedings of the 18th international conference on autonomous agents and multiagent systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, AAMAS '19, pp 2006–2008. http://dl.acm.org/citation.cfm?id=3306127.3331992

Hansen EA, Bernstein D, Zilberstein S (2004) Dynamic programming for partially observable stochastic games. In: AAAI

Hardin G (1968) The tragedy of the commons. Science 162(3859):1243–1248

Hausknecht M, Stone P (2015) Deep recurrent q-learning for partially observable mdps. https://www.aaai.org/ocs/index.php/FSS/FSS15/paper/view/11673

Havrylov S, Titov I (2017) Emergence of language with multi-agent games: Learning to communicate with sequences of symbols. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems 30, Curran Associates, Inc., pp 2149–2159. http://papers.nips.cc/paper/6810-emergence-of-language-with-multi-agent-games-learning-to-communicate-with-sequences-of-symbols.pdf

He H, Boyd-Graber J, Kwok K, III HD (2016) Opponent modeling in deep reinforcement learning. In: Balcan MF, Weinberger KQ (eds) Proceedings of The 33rd international conference on machine learning, PMLR, New York, New York, USA, Proceedings of Machine Learning Research, vol 48, pp 1804–1813. http://proceedings.mlr.press/v48/he16.html

He H, Chen D, Balakrishnan A, Liang P (2018) Decoupling strategy and generation in negotiation dialogues. CoRR arxiv: abs/1808.09637,

Heinrich J, Silver D (2016) Deep reinforcement learning from self-play in imperfect-information games. CoRR arxiv: abs/1603.01121,

Henderson P, Islam R, Bachman P, Pineau J, Precup D, Meger D (2018) Deep reinforcement learning that matters. https://www.aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/16669

Hernandez-Leal P, Kaisers M, Baarslag T, de Cote EM (2017) A survey of learning in multiagent environments: dealing with non-stationarity. CoRR arxiv: abs/1707.09183,

Hernandez-Leal P, Kartal B, Taylor ME (2019) Agent modeling as auxiliary task for deep reinforcement learning. CoRR arxiv: abs/1907.09597,

Hernandez-Leal P, Kartal B, Taylor ME (2019) A survey and critique of multiagent deep reinforcement learning. Auton Agent Multi-Agent Syst 33(6):750–797. https://doi.org/10.1007/s10458-019-09421-1

Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780. https://doi.org/10.1162/neco.1997.9.8.1735

Hong Z, Su S, Shann T, Chang Y, Lee C (2017) A deep policy inference q-network for multi-agent systems. CoRR arxiv: abs/1712.07893,

Hoshen Y (2017) Vain: Attentional multi-agent predictive modeling. In: Proceedings of the 31st international conference on neural information processing systems, Curran Associates Inc., USA, NIPS'17, pp 2698–2708. http://dl.acm.org/citation.cfm?id=3294996.3295030

Houthooft R, Chen X, Chen X, Duan Y, Schulman J, De Turck F, Abbeel P (2016) Vime: variational information maximizing exploration. In: Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R (eds) Advances in Neural Information Processing Systems 29, Curran Associates, Inc., pp 1109–1117. http://papers.nips.cc/paper/6591-vime-variational-information-maximizing-exploration.pdf

Hu J, Wellman MP (1998) Multiagent reinforcement learning: theoretical framework and an algorithm. In: Proceedings of the Fifteenth International Conference on Machine Learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, ICML '98, pp 242–250. http://dl.acm.org/citation.cfm?id=645527.657296

Hu J, Wellman MP (2003) Nash q-learning for general-sum stochastic games. J Mach Learn Res 4:1039–1069

Hughes E, Leibo JZ, Phillips M, Tuyls K, Dueñez Guzman E, García Castañeda A, Dunning I, Zhu T, McKee K, Koster R, Roff H, Graepel T (2018) Inequity aversion improves cooperation in intertemporal social dilemmas. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds) Advances in neural information processing systems 31, Curran Associates, Inc., pp 3326–3336. http://papers.nips.cc/paper/7593-inequity-aversion-improves-cooperation-in-intertemporal-social-dilemmas.pdf

Iqbal S, Sha F (2019) Actor-attention-critic for multi-agent reinforcement learning. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of the 36th international conference on machine learning, PMLR, Long Beach, California, USA, Proceedings of machine learning research, vol 97, pp 2961–2970. http://proceedings.mlr.press/v97/iqbal19a.html

Islam R, Henderson P, Gomrokchi M, Precup D (2017) Reproducibility of benchmarked deep reinforcement learning tasks for continuous control. CoRR arxiv: abs/1708.04133,

Jaderberg M, Czarnecki WM, Dunning I, Marris L, Lever G, Castañeda AG, Beattie C, Rabinowitz NC, Morcos AS, Ruderman A, Sonnerat N, Green T, Deason L, Leibo JZ, Silver D, Hassabis D, Kavukcuoglu K, Graepel T (2019) Human-level performance in 3d multiplayer games with population-based reinforcement learning. Science 364(6443):859–865

Jain U, Weihs L, Kolve E, Rastegari M, Lazebnik S, Farhadi A, Schwing AG, Kembhavi A (2019) Two body problem: Collaborative visual task completion. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition (CVPR)

Jaques N, Lazaridou A, Hughes E, Gülçehre Ç, Ortega PA, Strouse D, Leibo JZ, de Freitas N (2018) Intrinsic social motivation via causal influence in multi-agent RL. CoRR arxiv: abs/1810.08647,

Jaques N, Lazaridou A, Hughes E, Gulcehre C, Ortega P, Strouse D, Leibo JZ, De Freitas N (2019) Social influence as intrinsic motivation for multi-agent deep reinforcement learning. In: International conference on machine learning, pp 3040–3049

Jiang J, Lu Z (2018) Learning attentional communication for multi-agent cooperation. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds) Advances in neural information processing systems 31, Curran Associates, Inc., pp 7254–7264. http://papers.nips.cc/paper/7956-learning-attentional-communication-for-multi-agent-cooperation.pdf

Johnson M, Hofmann K, Hutton T, Bignell D (2016) The malmo platform for artificial intelligence experimentation. In: Proceedings of the twenty-fifth international joint conference on artificial intelligence, AAAI Press, IJCAI'16, pp 4246–4247. http://dl.acm.org/citation.cfm?id=3061053.3061259

Jorge E, Kågebäck M, Gustavsson E (2016) Learning to play guess who? and inventing a grounded language as a consequence. CoRR arxiv: abs/1611.03218,

Juliani A, Berges V, Vckay E, Gao Y, Henry H, Mattar M, Lange D (2018) Unity: a general platform for intelligent agents. CoRR arxiv: abs/1809.02627,

Kaelbling LP, Littman ML, Moore AW (1996) Reinforcement learning: a survey. J Artif Intell Res 4(1):237–285. http://dl.acm.org/citation.cfm?id=1622737.1622748
Kasai T, Tenmoto H, Kamiya A (2008) Learning of communication codes in multi-agent reinforcement learning problem. In: 2008 IEEE conference on soft computing in industrial applications, pp 1–6
Kim W, Cho M, Sung Y (2019) Message-dropout: An efficient training method for multi-agent deep reinforcement learning. In: Proceedings of the AAAI conference on artificial intelligence 33(01):6079–6086
Kirby S (2002) Natural language from artificial life. Artif Life 8(2):185–215. https://doi.org/10.1162/106454602320184248
Kok JR, Vlassis N (2006) Collaborative multiagent reinforcement learning by payoff propagation. J Mach Learn Res 7:1789–1828. http://dl.acm.org/citation.cfm?id=1248547.1248612
Kollock P (1998) Social dilemmas: the anatomy of cooperation. Annu Rev Sociol 24(1):183–214. https://doi.org/10.1146/annurev.soc.24.1.183
Kong X, Xin B, Liu F, Wang Y (2017) Revisiting the master-slave architecture in multi-agent deep reinforcement learning. CoRR arxiv: abs/1712.07305,
Kraemer L, Banerjee B (2016) Multi-agent reinforcement learning as a rehearsal for decentralized planning. Neurocomputing 190:82–94
Kumar S, Shah P, Hakkani-Tür D, Heck LP (2017) Federated control with hierarchical multi-agent deep reinforcement learning. CoRR arxiv: abs/1712.08266,
Lanctot M, Zambaldi V, Gruslys A, Lazaridou A, Tuyls K, Perolat J, Silver D, Graepel T (2017) A unified game-theoretic approach to multiagent reinforcement learning. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems 30, Curran Associates, Inc., pp 4190–4203. http://papers.nips.cc/paper/7007-a-unified-game-theoretic-approach-to-multiagent-reinforcement-learning.pdf
Lange PAV, Joireman J, Parks CD, Dijk EV (2013) The psychology of social dilemmas: a review. Organ Behav Hum Decis Process 120(2):125–141
Lauer M, Riedmiller M (2000) An algorithm for distributed reinforcement learning in cooperative multi-agent systems. In: In Proceedings of the Seventeenth International Conference on Machine Learning, Morgan Kaufmann, pp 535–542
Laurent GJ, Matignon L, Fort-Piat NL (2011) The world of independent learners is not markovian. Int J Knowl-Based Intell Eng Syst 15(1):55–64. http://dl.acm.org/citation.cfm?id=1971886.1971887
Lazaridou A, Baroni M (2020) Emergent multi-agent communication in the deep learning era. ArXiv arxiv: abs/2006.02419
Lazaridou A, Peysakhovich A, Baroni M (2017) Multi-agent cooperation and the emergence of (natural) language. In: 5th international conference on learning representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings. https://openreview.net/forum?id=Hk8N3Sclg
Lazaridou A, Hermann KM, Tuyls K, Clark S (2018) Emergence of linguistic communication from referential games with symbolic and pixel input. In: International conference on learning representations. https://openreview.net/forum?id=HJGv1Z-AW
Le HM, Yue Y, Carr P, Lucey P (2017) Coordinated multi-agent imitation learning. In: Precup D, Teh YW (eds) Proceedings of the 34th international conference on machine learning, PMLR, International Convention Centre, Sydney, Australia, Proceedings of Machine Learning Research, vol 70, pp 1995–2003. http://proceedings.mlr.press/v70/le17a.html
Lee J, Cho K, Weston J, Kiela D (2017) Emergent translation in multi-agent communication. CoRR arxiv: abs/1710.06922,
Lee Y, Yang J, Lim JJ (2020) Learning to coordinate manipulation skills via skill behavior diversification. In: International conference on learning representations. https://openreview.net/forum?id=ryxB2lBtvH
Leibo JZ, Zambaldi V, Lanctot M, Marecki J, Graepel T (2017) Multi-agent reinforcement learning in sequential social dilemmas. In: Proceedings of the 16th conference on autonomous agents and multiagent systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, AAMAS '17, pp 464–473. http://dl.acm.org/citation.cfm?id=3091125.3091194
Leibo JZ, Hughes E, Lanctot M, Graepel T (2019) Autocurricula and the emergence of innovation from social interaction: a manifesto for multi-agent intelligence research. CoRR arxiv: abs/1903.00742,
Lerer A, Peysakhovich A (2017) Maintaining cooperation in complex social dilemmas using deep reinforcement learning. CoRR arxiv: abs/1707.01068,
Letcher A, Foerster J, Balduzzi D, Rocktäschel T, Whiteson S (2019) Stable opponent shaping in differentiable games. In: International conference on learning representations. https://openreview.net/forum?id=SyGjjsC5tQ

Levine S, Finn C, Darrell T, Abbeel P (2016) End-to-end training of deep visuomotor policies. Journal of Machine Learning Research 17(1):1334–1373. http://dl.acm.org/citation.cfm?id=2946645.2946684

Lewis M, Yarats D, Dauphin YN, Parikh D, Batra D (2017) Deal or no deal? end-to-end learning for negotiation dialogues. CoRR arxiv: abs/1706.05125,

Li F, Bowling M (2019) Ease-of-teaching and language structure from emergent communication. In: Wallach H, Larochelle H, Beygelzimer A, Alche-Buc F, Fox E, Garnett R (eds) Advances in neural information processing systems 32, Curran Associates, Inc., pp 15851–15861. http://papers.nips.cc/paper/9714-ease-of-teaching-and-language-structure-from-emergent-communication.pdf

Li S, Wu Y, Cui X, Dong H, Fang F, Russell S (2019a) Robust multi-agent reinforcement learning via minimax deep deterministic policy gradient. Proc AAAI Conf Artif Intell 33(01):4213–4220

Li X, Sun M, Li P (2019b) Multi-agent discussion mechanism for natural language generation. Proc AAAI Conf Artif Intell 33(01):6096–6103

Li Y (2018) Deep reinforcement learning. CoRR arxiv: abs/1810.06339,

Lillicrap TP, Hunt JJ, Pritzel A, Heess N, Erez T, Tassa Y, Silver D, Wierstra D (2016) Continuous control with deep reinforcement learning. In: ICLR (Poster). http://arxiv.org/arxiv: abs/1509.02971

Lin K, Zhao R, Xu Z, Zhou J (2018) Efficient large-scale fleet management via multi-agent deep reinforcement learning. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining, ACM, New York, NY, USA, KDD '18, pp 1774–1783. https://doi.org/10.1145/3219819.3219993,

Lin X, Beling PA, Cogill R (2018) Multiagent inverse reinforcement learning for two-person zero-sum games. IEEE Trans Games 10(1):56–68. https://doi.org/10.1109/TCIAIG.2017.2679115

Littman M (2001) Value-function reinforcement learning in markov games. Cogn Syst Res 2:55–66

Littman ML (1994) Markov games as a framework for multi-agent reinforcement learning. In: Proceedings of the eleventh international conference on international conference on machine learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML'94, pp 157–163. http://dl.acm.org/citation.cfm?id=3091574.3091594

Liu IJ, Yeh RA, Schwing AG (2020) Pic: Permutation invariant critic for multi-agent deep reinforcement learning. In: PMLR, proceedings of machine learning research, vol 100, pp 590–602. http://proceedings.mlr.press/v100/liu20a.html

Liu S, Lever G, Heess N, Merel J, Tunyasuvunakool S, Graepel T (2019) Emergent coordination through competition. In: International conference on learning representations. https://openreview.net/forum?id=BkG8sjR5Km

Long Q, Zhou Z, Gupta A, Fang F, Wu Y, Wang X (2020) Evolutionary population curriculum for scaling multi-agent reinforcement learning. In: International conference on learning representations. https://openreview.net/forum?id=SJxbHkrKDH

Lowe R, WU Y, Tamar A, Harb J, Pieter Abbeel O, Mordatch I (2017) Multi-agent actor-critic for mixed cooperative-competitive environments. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems 30, Curran Associates, Inc., pp 6379–6390. http://papers.nips.cc/paper/7217-multi-agent-actor-critic-for-mixed-cooperative-competitive-environments.pdf

Lowe R, Foerster JN, Boureau Y, Pineau J, Dauphin YN (2019) On the pitfalls of measuring emergent communication. CoRR arxiv: abs/1903.05168,

Luketina J, Nardelli N, Farquhar G, Foerster JN, Andreas J, Grefenstette E, Whiteson S, Rocktäschel T (2019) A survey of reinforcement learning informed by natural language. CoRR arxiv: abs/1906.03926,

Luong NC, Hoang DT, Gong S, Niyato D, Wang P, Liang Y, Kim DI (2019) Applications of deep reinforcement learning in communications and networking: a survey. IEEE Communications Surveys Tutorials pp 1–1. https://doi.org/10.1109/COMST.2019.2916583

Lux T, Marchesi M (1999) Scaling and criticality in a stochastic multi-agent model of a financial market. Nature 397(6719):498–500. https://doi.org/10.1038/17290

Lyu X, Amato C (2020) Likelihood quantile networks for coordinating multi-agent reinforcement learning. In: Proceedings of the 19th international conference on autonomous agents and multiagent systems, pp 798–806

Ma J, Wu F (2020) Feudal multi-agent deep reinforcement learning for traffic signal control. In: Seghrouchni AEF, Sukthankar G, An B, Yorke-Smith N (eds) Proceedings of the 19th international conference on autonomous agents and multiagent systems, AAMAS '20, Auckland, New Zealand, May 9-13, 2020, International Foundation for Autonomous Agents and Multiagent Systems, pp 816–824. https://dl.acm.org/doi/arxiv: abs/10.5555/3398761.3398858

Makar R, Mahadevan S, Ghavamzadeh M (2001) Hierarchical multi-agent reinforcement learning. In: Proceedings of the fifth international conference on autonomous agents, ACM, New York, NY, USA, AGENTS '01, pp 246–253. https://doi.org/10.1145/375735.376302,

Matignon L, Laurent GJ, Le Fort-Piat N (2007) Hysteretic q-learning : an algorithm for decentralized reinforcement learning in cooperative multi-agent teams. In: 2007 IEEE/RSJ international conference on intelligent robots and systems, pp 64–69

Matignon L, Jeanpierre L, Mouaddib AI (2012a) Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes. https://www.aaai.org/ocs/index.php/AAAI/AAAI12/paper/view/5038

Matignon L, Gj Laurent, Le fort piat N, (2012b) Review: independent reinforcement learners in cooperative markov games: a survey regarding coordination problems. Knowl Eng Rev 27(1):1–31. https://doi.org/10.1017/S0269888912000057

Mnih V, Kavukcuoglu K, Silver D, Rusu AA, Veness J, Bellemare MG, Graves A, Riedmiller M, Fidjeland AK, Ostrovski G, Petersen S, Beattie C, Sadik A, Antonoglou I, King H, Kumaran D, Wierstra D, Legg S, Hassabis D (2015) Human-level control through deep reinforcement learning. Nature 518:529 EP –. https://doi.org/10.1038/nature14236

Mnih V, Badia AP, Mirza M, Graves A, Lillicrap T, Harley T, Silver D, Kavukcuoglu K (2016) Asynchronous methods for deep reinforcement learning. In: Balcan MF, Weinberger KQ (eds) Proceedings of The 33rd international conference on machine learning, PMLR, New York, New York, USA, Proceedings of machine learning research, vol 48, pp 1928–1937. http://proceedings.mlr.press/v48/mniha16.html

Moerland TM, Broekens J, Jonker CM (2018) Emotion in reinforcement learning agents and robots: a survey. Mach Learn 107(2):443–480. https://doi.org/10.1007/s10994-017-5666-0

Mordatch I, Abbeel P (2018) Emergence of grounded compositional language in multi-agent populations. https://aaai.org/ocs/index.php/AAAI/AAAI18/paper/view/17007

Nair R, Tambe M, Yokoo M, Pynadath D, Marsella S (2003) Taming decentralized pomdps: towards efficient policy computation for multiagent settings. In: Proceedings of the 18th international joint conference on artificial intelligence, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, IJCAI'03, pp 705–711. http://dl.acm.org/citation.cfm?id=1630659.1630762

Narvekar S, Sinapov J, Leonetti M, Stone P (2016) Source task creation for curriculum learning. In: Proceedings of the 2016 international conference on autonomous agents & multiagent systems, international foundation for autonomous agents and multiagent systems, Richland, SC, AAMAS '16, pp 566–574. http://dl.acm.org/citation.cfm?id=2936924.2937007

Nedic A, Ozdaglar A (2009) Distributed subgradient methods for multi-agent optimization. IEEE Trans Autom Control 54(1):48–61

Ng AY, Russell SJ (2000) Algorithms for inverse reinforcement learning. In: Proceedings of the seventeenth international conference on machine learning, Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, ICML '00, pp 663–670. http://dl.acm.org/citation.cfm?id=645529.657801

Ng AY, Harada D, Russell S (1999) Policy invariance under reward transformations: theory and application to reward shaping. In: In Proceedings of the sixteenth international conference on machine learning, Morgan Kaufmann, pp 278–287

Nguyen DT, Kumar A, Lau HC (2017a) Collective multiagent sequential decision making under uncertainty. https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14891

Nguyen DT, Kumar A, Lau HC (2017b) Policy gradient with value function approximation for collective multiagent planning. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems 30, Curran Associates, Inc., pp 4319–4329. http://papers.nips.cc/paper/7019-policy-gradient-with-value-function-approximation-for-collective-multiagent-planning.pdf

Nguyen DT, Kumar A, Lau HC (2018) Credit assignment for collective multiagent rl with global rewards. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds) Advances in neural information processing systems 31, Curran Associates, Inc., pp 8102–8113. http://papers.nips.cc/paper/8033-credit-assignment-for-collective-multiagent-rl-with-global-rewards.pdf

Nguyen TT, Nguyen ND, Nahavandi S (2020) Deep reinforcement learning for multiagent systems: a review of challenges, solutions, and applications. IEEE Trans Cybern 50(9):3826–3839

Oliehoek FA, Amato C (2016) A Concise Introduction to Decentralized POMDPs, 1st edn. Springer Publishing Company, Berlin

Oliehoek FA, Spaan MTJ, Vlassis N (2008) Optimal and approximate q-value functions for decentralized pomdps. J Artif Int Res 32(1):289–353. http://dl.acm.org/citation.cfm?id=1622673.1622680

Omidshafiei S, Pazis J, Amato C, How JP, Vian J (2017) Deep decentralized multi-task multi-agent reinforcement learning under partial observability. In: Precup D, Teh YW (eds) Proceedings of the 34th

international conference on machine learning, PMLR, International Convention Centre, Sydney, Australia, Proceedings of machine learning research, vol 70, pp 2681–2690. http://proceedings.mlr.press/v70/omidshafiei17a.html

Omidshafiei S, Kim DK, Liu M, Tesauro G, Riemer M, Amato C, Campbell M, How JP (2019) Learning to teach in cooperative multiagent reinforcement learning. Proc AAAI Conf Artif Intelli 33(01):6128–6136

Oroojlooyjadid A, Hajinezhad D (2019) A review of cooperative multi-agent deep reinforcement learning. ArXiv arxiv: abs/1908.03963

Oudeyer PY, Kaplan F (2007) What is intrinsic motivation? A typology of computational approaches. Front Neurorobotics 1:6–6

Palmer G, Tuyls K, Bloembergen D, Savani R (2018) Lenient multi-agent deep reinforcement learning. In: Proceedings of the 17th international conference on autonomous agents and multiagent systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, AAMAS '18, pp 443–451. http://dl.acm.org/citation.cfm?id=3237383.3237451

Palmer G, Savani R, Tuyls K (2019) Negative update intervals in deep multi-agent reinforcement learning. In: Proceedings of the 18th international conference on autonomous agents and multiagent systems, pp 43–51

Panait L, Luke S (2005) Cooperative multi-agent learning: the state of the art. Auton Agent Multi-Agent Syst 11(3):387–434. https://doi.org/10.1007/s10458-005-2631-2

Panait L, Sullivan K, Luke S (2006) Lenient learners in cooperative multiagent systems. In: Proceedings of the fifth international joint conference on autonomous agents and multiagent systems, association for computing machinery, New York, NY, USA, AAMAS '06, pp 801–803. https://doi.org/10.1145/1160633.1160776,

Papoudakis G, Christianos F, Rahman A, Albrecht SV (2019) Dealing with non-stationarity in multi-agent deep reinforcement learning. CoRR arxiv: abs/1906.04737,

Pathak D, Agrawal P, Efros AA, Darrell T (2017) Curiosity-driven exploration by self-supervised prediction. In: Precup D, Teh YW (eds) Proceedings of the 34th international conference on machine learning, PMLR, International Convention Centre, Sydney, Australia, Proceedings of Machine Learning Research, vol 70, pp 2778–2787. http://proceedings.mlr.press/v70/pathak17a.html

Peng P, Yuan Q, Wen Y, Yang Y, Tang Z, Long H, Wang J (2017) Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. CoRR arxiv: abs/1703.10069,

Pérolat J, Leibo JZ, Zambaldi V, Beattie C, Tuyls K, Graepel T (2017) A multi-agent reinforcement learning model of common-pool resource appropriation. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R (eds) Advances in neural information processing systems 30, Curran Associates, Inc., pp 3643–3652. http://papers.nips.cc/paper/6955-a-multi-agent-reinforcement-learning-model-of-common-pool-resource-appropriation.pdf

Peysakhovich A, Lerer A (2018) Prosocial learning agents solve generalized stag hunts better than selfish ones. In: Proceedings of the 17th international conference on autonomous agents and multiagent systems, international Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, AAMAS '18, pp 2043–2044. http://dl.acm.org/citation.cfm?id=3237383.3238065

Pinto L, Davidson J, Sukthankar R, Gupta A (2017) Robust adversarial reinforcement learning. In: Precup D, Teh YW (eds) Proceedings of the 34th international conference on machine learning, PMLR, International Convention Centre, Sydney, Australia, Proceedings of machine learning research, vol 70, pp 2817–2826. http://proceedings.mlr.press/v70/pinto17a.html

Pinyol I, Sabater-Mir J (2013) Computational trust and reputation models for open multi-agent systems: a review. Artif Intell Rev 40(1):1–25. https://doi.org/10.1007/s10462-011-9277-z

Potter MA, De Jong KA (1994) A cooperative coevolutionary approach to function optimization. In: Davidor Y, Schwefel HP, Männer R (eds) Parallel problem solving from nature - PPSN III. Springer, Berlin, pp 249–257

Qu G, Wierman A, Li N (2020) Scalable reinforcement learning of localized policies for multi-agent networked systems. PMLR, The Cloud, Proceedings of machine learning research, vol 120, pp 256–266. http://proceedings.mlr.press/v120/qu20a.html

Rabinowitz N, Perbet F, Song F, Zhang C, Eslami SMA, Botvinick M (2018) Machine theory of mind. In: Dy J, Krause A (eds) Proceedings of the 35th international conference on machine learning, PMLR, Stockholmsmässan, Stockholm Sweden, Proceedings of machine learning research, vol 80, pp 4218–4227. http://proceedings.mlr.press/v80/rabinowitz18a.html

Raghu M, Irpan A, Andreas J, Kleinberg B, Le Q, Kleinberg J (2018) Can deep reinforcement learning solve Erdos-Selfridge-Spencer games? In: Dy J, Krause A (eds) Proceedings of the 35th international conference on machine learning, PMLR, Stockholmsmässan, Stockholm Sweden, Proceedings of machine learning research, vol 80, pp 4238–4246. http://proceedings.mlr.press/v80/raghu18a.html

Raileanu R, Denton E, Szlam A, Fergus R (2018) Modeling others using oneself in multi-agent reinforcement learning. In: Dy J, Krause A (eds) Proceedings of the 35th international conference on machine learning, PMLR, Stockholmsmässan, Stockholm Sweden, Proceedings of machine learning research, vol 80, pp 4257–4266. http://proceedings.mlr.press/v80/raileanu18a.html

Ramchurn SD, Huynh D, Jennings NR (2004) Trust in multi-agent systems. Knowl Eng Rev 19(1):1–25. https://doi.org/10.1017/S0269888904000116

Rashid T, Samvelyan M, Schroeder C, Farquhar G, Foerster J, Whiteson S (2018) QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In: Dy J, Krause A (eds) Proceedings of the 35th international conference on machine learning, PMLR, Stockholmsmässan, Stockholm Sweden, Proceedings of machine learning research, vol 80, pp 4295–4304. http://proceedings.mlr.press/v80/rashid18a.html

Russell S, Zimdars AL (2003) Q-decomposition for reinforcement learning agents. In: Proceedings of the twentieth international conference on international conference on machine learning, AAAI Press, ICML'03, pp 656–663. http://dl.acm.org/citation.cfm?id=3041838.3041921

Schaul T, Horgan D, Gregor K, Silver D (2015) Universal value function approximators. In: Proceedings of the 32nd international conference on international conference on machine learning - volume 37, JMLR.org, ICML'15, pp 1312–1320

Schmidhuber J (2010) Formal theory of creativity, fun, and intrinsic motivation (1990–2010). IEEE Trans Auton Ment Dev 2(3):230–247. https://doi.org/10.1109/TAMD.2010.2056368

Schmidhuber J, Zhao J, Wiering M (1996) Simple principles of metalearning. Tech. rep

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

Sen S, Weiss G (1999) Multiagent systems. MIT Press, Cambridge, MA, USA. http://dl.acm.org/citation.cfm?id=305606.305612

Sequeira P, Melo FS, Prada R, Paiva A (2011) Emerging social awareness: exploring intrinsic motivation in multiagent learning. In: 2011 IEEE international conference on development and learning (ICDL), vol 2, pp 1–6. https://doi.org/10.1109/DEVLRN.2011.6037325

Shalev-Shwartz S, Shammah S, Shashua A (2016) Safe, multi-agent, reinforcement learning for autonomous driving. CoRR arxiv: abs/1610.03295,

Shapley LS (1953) Stochastic games. Proc Nat Acad Sci 39(10):1095–1100

Shoham Y, Leyton-Brown K (2008) Multiagent systems: algorithmic, game-theoretic, and logical foundations. Cambridge University Press, USA

Shoham Y, Powers R, Grenager T (2003) Multi-agent reinforcement learning: a critical survey. Tech. rep

Silva FLD, Taylor ME, Costa AHR (2018) Autonomously reusing knowledge in multiagent reinforcement learning. In: Proceedings of the twenty-seventh international joint conference on artificial intelligence, IJCAI-18, International joint conferences on artificial intelligence organization, pp 5487–5493. https://doi.org/10.24963/ijcai.2018/774,

Silver D, Huang A, Maddison CJ, Guez A, Sifre L, van den Driessche G, Schrittwieser J, Antonoglou I, Panneershelvam V, Lanctot M, Dieleman S, Grewe D, Nham J, Kalchbrenner N, Sutskever I, Lillicrap T, Leach M, Kavukcuoglu K, Graepel T, Hassabis D (2016) Mastering the game of go with deep neural networks and tree search. Nature 529:484 EP –. https://doi.org/10.1038/nature16961

Silver D, Hubert T, Schrittwieser J, Antonoglou I, Lai M, Guez A, Lanctot M, Sifre L, Kumaran D, Graepel T, Lillicrap T, Simonyan K, Hassabis D (2018) A general reinforcement learning algorithm that masters chess, shogi, and go through self-play. Science 362(6419):1140–1144

Singh A, Jain T, Sukhbaatar S (2019) Learning when to communicate at scale in multiagent cooperative and competitive tasks. In: International conference on learning representations. https://openreview.net/forum?id=rye7knCqK7

Son K, Kim D, Kang WJ, Hostallero DE, Yi Y (2019) Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In: International conference on machine learning, pp 5887–5896

Song J, Ren H, Sadigh D, Ermon S (2018) Multi-agent generative adversarial imitation learning. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds) Advances in neural information processing systems, Curran Associates, Inc., vol 31, pp 7461–7472. https://proceedings.neurips.cc/paper/2018/file/240c945bb72980130446fc2b40fbb8e0-Paper.pdf

Song Y, Wang J, Lukasiewicz T, Xu Z, Xu M, Ding Z, Wu L (2019) Arena: A general evaluation platform and building toolkit for multi-agent intelligence. CoRR arxiv: abs/1905.08085,

Spooner T, Savani R (2020) Robust market making via adversarial reinforcement learning. In: Proceedings of the 19th international conference on autonomous agents and multiagent systems, pp 2014–2016

Srinivasan S, Lanctot M, Zambaldi V, Perolat J, Tuyls K, Munos R, Bowling M (2018) Actor-critic policy optimization in partially observable multiagent environments. In: Bengio S, Wallach H, Larochelle

H, Grauman K, Cesa-Bianchi N, Garnett R (eds) Advances in neural information processing systems 31, Curran Associates, Inc., pp 3422–3435. http://papers.nips.cc/paper/7602-actor-critic-policy-optimization-in-partially-observable-multiagent-environments.pdf

Stone P, Veloso M (2000) Multiagent systems: a survey from a machine learning perspective. Auton Robots 8(3):345–383. https://doi.org/10.1023/A:1008942012299

Strouse D, Kleiman-Weiner M, Tenenbaum J, Botvinick M, Schwab DJ (2018) Learning to share and hide intentions using information regularization. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds) Advances in neural information processing systems 31, Curran Associates, Inc., pp 10249–10259. http://papers.nips.cc/paper/8227-learning-to-share-and-hide-intentions-using-information-regularization.pdf

Sukhbaatar S, szlam a, Fergus R (2016) Learning multiagent communication with backpropagation. In: Lee DD, Sugiyama M, Luxburg UV, Guyon I, Garnett R (eds) Advances in neural information processing systems 29, Curran Associates, Inc., pp 2244–2252. http://papers.nips.cc/paper/6398-learning-multiagent-communication-with-backpropagation.pdf

Sukhbaatar S, Kostrikov I, Szlam A, Fergus R (2017) Intrinsic motivation and automatic curricula via asymmetric self-play. CoRR arxiv: abs/1703.05407,

Sunehag P, Lever G, Gruslys A, Czarnecki WM, Zambaldi V, Jaderberg M, Lanctot M, Sonnerat N, Leibo JZ, Tuyls K, Graepel T (2018) Value-decomposition networks for cooperative multi-agent learning based on team reward. In: Proceedings of the 17th international conference on autonomous agents and multiagent systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, AAMAS '18, pp 2085–2087. http://dl.acm.org/citation.cfm?id=3237383.3238080

Sutton RS, Barto AG (1998) Reinforcement learning: an introduction. Adaptive computation and machine learning, MIT Press. http://www.worldcat.org/oclc/37293240

Sutton RS, Precup D, Singh S (1999) Between mdps and semi-mdps: a framework for temporal abstraction in reinforcement learning. Artif Intell 112(1):181–211

Svetlik M, Leonetti M, Sinapov J, Shah R, Walker N, Stone P (2017) Automatic curriculum graph generation for reinforcement learning agents. https://aaai.org/ocs/index.php/AAAI/AAAI17/paper/view/14961

Tacchetti A, Song HF, Mediano PAM, Zambaldi V, Kramár J, Rabinowitz NC, Graepel T, Botvinick M, Battaglia PW (2019) Relational forward models for multi-agent learning. In: International conference on learning representations. https://openreview.net/forum?id=rJlEojAqFm

Tampuu A, Matiisen T, Kodelja D, Kuzovkin I, Korjus K, Aru J, Aru J, Vicente R (2017) Multiagent cooperation and competition with deep reinforcement learning. PLoS ONE 12(4):1–15. https://doi.org/10.1371/journal.pone.0172395

Tan M (1993) Multi-agent reinforcement learning: Independent vs. cooperative agents. In: In Proceedings of the tenth international conference on machine learning, Morgan Kaufmann, pp 330–337

Tang H, Hao J, Lv T, Chen Y, Zhang Z, Jia H, Ren C, Zheng Y, Fan C, Wang L (2018) Hierarchical deep multiagent reinforcement learning. CoRR arxiv: abs/1809.09332,

Taylor A, Dusparic I, Cahill V (2013) Transfer learning in multi-agent systems through parallel transfer. In: in Workshop on theoretically grounded transfer learning at the 30th international conference on machine learning (Poster

Taylor ME, Stone P (2009) Transfer learning for reinforcement learning domains: a survey. J Mach Learn Res 10:1633–1685. http://dl.acm.org/citation.cfm?id=1577069.1755839

Tesauro G (2004) Extending q-learning to general adaptive multi-agent systems. In: Thrun S, Saul LK, Schölkopf B (eds) Advances in neural information processing systems 16, MIT Press, pp 871–878. http://papers.nips.cc/paper/2503-extending-q-learning-to-general-adaptive-multi-agent-systems.pdf

Tumer K, Wolpert DH (2004) Collectives and the design of complex systems. Springer, Berlin

Tuyls K, Weiss G (2012) Multiagent learning: basics, challenges, and prospects. AI Mag 33(3):41

Vezhnevets AS, Osindero S, Schaul T, Heess N, Jaderberg M, Silver D, Kavukcuoglu K (2017) FeUdal networks for hierarchical reinforcement learning. In: Precup D, Teh YW (eds) Proceedings of the 34th international conference on machine learning, PMLR, International Convention Centre, Sydney, Australia, Proceedings of Machine Learning Research, vol 70, pp 3540–3549. http://proceedings.mlr.press/v70/vezhnevets17a.html

Vezhnevets AS, Wu Y, Leblond R, Leibo JZ (2019) Options as responses: grounding behavioural hierarchies in multi-agent RL. CoRR arxiv: abs/1906.01470,

Vinyals O, Ewalds T, Bartunov S, Georgiev P, Vezhnevets AS, Yeo M, Makhzani A, Küttler H, Agapiou J, Schrittwieser J, Quan J, Gaffney S, Petersen S, Simonyan K, Schaul T, van Hasselt H, Silver D, Lillicrap TP, Calderone K, Keet P, Brunasso A, Lawrence D, Ekermo A, Repp J, Tsing R (2017) Starcraft II: a new challenge for reinforcement learning. CoRR arxiv: abs/1708.04782,

Vinyals O, Babuschkin I, Czarnecki WM, Mathieu M, Dudzik A, Chung J, Choi DH, Powell R, Ewalds T, Georgiev P, Oh J, Horgan D, Kroiss M, Danihelka I, Huang A, Sifre L, Cai T, Agapiou JP, Jaderberg M, Vezhnevets AS, Leblond R, Pohlen T, Dalibard V, Budden D, Sulsky Y, Molloy J, Paine TL, Gulcehre C, Wang Z, Pfaff T, Wu Y, Ring R, Yogatama D, Wünsch D, McKinney K, Smith O, Schaul T, Lillicrap T, Kavukcuoglu K, Hassabis D, Apps C, Silver D (2019) Grandmaster level in starcraft ii using multi-agent reinforcement learning. Nature 575(7782):350–354. https://doi.org/10.1038/s41586-019-1724-z

Wang JX, Kurth-Nelson Z, Tirumala D, Soyer H, Leibo JZ, Munos R, Blundell C, Kumaran D, Botvinick M (2016a) Learning to reinforcement learn. CoRR arxiv: abs/1611.05763,

Wang JX, Hughes E, Fernando C, Czarnecki WM, Duéñez Guzmán EA, Leibo JZ (2019) Evolving intrinsic motivations for altruistic behavior. In: Proceedings of the 18th international conference on autonomous agents and multiagent systems, International Foundation for Autonomous Agents and Multiagent Systems, Richland, SC, AAMAS '19, pp 683–692. http://dl.acm.org/citation.cfm?id=3306127.3331756

Wang S, Wan J, Zhang D, Li D, Zhang C (2016b) Towards smart factory for industry 4.0: a self-organized multi-agent system with big data based feedback and coordination. Comput Netw 101:158–168. https://doi.org/10.1016/j.comnet.2015.12.017. http://www.sciencedirect.com/science/article/pii/S1389128615005046, industrial Technologies and Applications for the Internet of Things

Wang T, Dong H, Lesser VR, Zhang C (2020a) ROMA: multi-agent reinforcement learning with emergent roles. CoRR arxiv: abs/2003.08039

Wang T, Wang J, Wu Y, Zhang C (2020b) Influence-based multi-agent exploration. In: International conference on learning representations. https://openreview.net/forum?id=BJgy96EYvr

Wang T, Wang J, Zheng C, Zhang C (2020c) Learning nearly decomposable value functions via communication minimization. In: International conference on learning representations. https://openreview.net/forum?id=HJx-3grYDB

Wei E, Luke S (2016) Lenient learning in independent-learner stochastic cooperative games. J Mach Learn Res 17(84):1–42. http://jmlr.org/papers/v17/15-417.html

Wei E, Wicke D, Freelan D, Luke S (2018) Multiagent soft q-learning. https://www.aaai.org/ocs/index.php/SSS/SSS18/paper/view/17508

Wei Ren, Beard RW, Atkins EM (2005) A survey of consensus problems in multi-agent coordination. In: Proceedings of the 2005, American control conference, 2005., pp 1859–1864 vol. 3. https://doi.org/10.1109/ACC.2005.1470239

Weiß G (1995) Distributed reinforcement learning. In: Steels L (ed) The biology and technology of intelligent autonomous agents. Springer, Berlin, pp 415–428

Weiss G (ed) (1999) Multiagent systems: a modern approach to distributed artificial intelligence. MIT Press, Cambridge

Wiegand RP (2004) An analysis of cooperative coevolutionary algorithms. PhD thesis, USA, aAI3108645

Wolpert DH, Tumer K (1999) An introduction to collective intelligence. CoRR cs.LG/9908014. http://arxiv.org/arxiv: abs/cs.LG/9908014

Wu C, Rajeswaran A, Duan Y, Kumar V, Bayen AM, Kakade S, Mordatch I, Abbeel P (2018) Variance reduction for policy gradient with action-dependent factorized baselines. In: International conference on learning representations. https://openreview.net/forum?id=H1tSsb-AW

Yang E, Gu D (2004) Multiagent reinforcement learning for multi-robot systems: a survey. Tech. rep

Yang J, Nakhaei A, Isele D, Fujimura K, Zha H (2020) Cm3: Cooperative multi-goal multi-stage multi-agent reinforcement learning. In: International conference on learning representations. https://openreview.net/forum?id=S1lEX04tPr

Yang T, Meng Z, Hao J, Zhang C, Zheng Y (2018a) Bayes-tomop: a fast detection and best response algorithm towards sophisticated opponents. CoRR arxiv: abs/1809.04240,

Yang Y, Luo R, Li M, Zhou M, Zhang W, Wang J (2018b) Mean field multi-agent reinforcement learning. In: Dy J, Krause A (eds) Proceedings of the 35th international conference on machine learning, PMLR, Stockholmsmässan, Stockholm Sweden, Proceedings of machine learning research, vol 80, pp 5571–5580. http://proceedings.mlr.press/v80/yang18d.html

Yu C, Zhang M, Ren F (2013) Emotional multiagent reinforcement learning in social dilemmas. In: Boella G, Elkind E, Savarimuthu BTR, Dignum F, Purvis MK (eds) PRIMA 2013: principles and practice of multi-agent systems. Springer, Berlin, pp 372–387

Yu H, Shen Z, Leung C, Miao C, Lesser VR (2013) A survey of multi-agent trust management systems. IEEE Access 1:35–50. https://doi.org/10.1109/ACCESS.2013.2259892

Yu L, Song J, Ermon S (2019) Multi-agent adversarial inverse reinforcement learning. In: Chaudhuri K, Salakhutdinov R (eds) Proceedings of the 36th international conference on machine learning,

PMLR, Long Beach, California, USA, Proceedings of machine learning research, vol 97, pp 7194–7201. http://proceedings.mlr.press/v97/yu19e.html

Zhang K, Yang Z, Basar T (2018) Networked multi-agent reinforcement learning in continuous spaces. In: 2018 IEEE conference on decision and control (CDC), pp 2771–2776

Zhang K, Yang Z, Liu H, Zhang T, Basar T (2018) Fully decentralized multi-agent reinforcement learning with networked agents. In: Dy J, Krause A (eds) Proceedings of the 35th international conference on machine learning, PMLR, Stockholmsmässan, Stockholm Sweden, Proceedings of machine learning research, vol 80, pp 5872–5881. http://proceedings.mlr.press/v80/zhang18n.html

Zhang K, Yang Z, Başar T (2019) Multi-agent reinforcement learning: a selective overview of theories and algorithms. ArXiv arxiv: abs/1911.10635

Zhang W, Bastani O (2019) Mamps: Safe multi-agent reinforcement learning via model predictive shielding. ArXiv arxiv: abs/1910.12639

Zheng Y, Meng Z, Hao J, Zhang Z (2018a) Weighted double deep multiagent reinforcement learning in stochastic cooperative environments. In: Geng X, Kang BH (eds) PRICAI 2018: trends in artificial intelligence. Springer International Publishing, Cham, pp 421–429

Zheng Y, Meng Z, Hao J, Zhang Z, Yang T, Fan C (2018b) A deep bayesian policy reuse approach against non-stationary agents. In: Bengio S, Wallach H, Larochelle H, Grauman K, Cesa-Bianchi N, Garnett R (eds) Advances in neural information processing systems 31, Curran Associates, Inc., pp 954–964. http://papers.nips.cc/paper/7374-a-deep-bayesian-policy-reuse-approach-against-non-stationary-agents.pdf

Zhu H, Kirley M (2019) Deep multi-agent reinforcement learning in a common-pool resource system. In: 2019 IEEE congress on evolutionary computation (CEC), pp 142–149. https://doi.org/10.1109/CEC.2019.8790001

Zhu Z, Biyik E, Sadigh D (2020) Multi-agent safe planning with gaussian processes. ArXiv arxiv: abs/2008.04452

Multi-agent deep reinforcement learning: a survey

Author: Sven Gronauer et al.
Publication: Artificial Intelligence Review
Publisher: Springer Nature
Date: April 15, 2021

Reprint Permission

Creative Commons