Trinity College Dublin
Coláiste na Tríonóide, Baile Átha Cliath
The University of Dublin

PHD THESIS

Expert-Free Online Transfer Learning in Multi-Agent Reinforcement Learning

---

Author:
ALBERTO CASTAGNA

Supervisor:
Dr. IVANA DUSPARIC

18th May 2024

Abstract

Reinforcement Learning (RL) enables an intelligent agent to optimise its performance in a task by continuously taking action from an observed state and receiving a feedback from the environment in form of rewards. RL typically uses tables or linear approximators to map state-action tuples that maximises the reward. Combining RL with deep neural networks (DRL) significantly increases its scalability and enables it to address more complex problems than before. However, DRL also inherits downsides from both RL and deep learning. Despite DRL improves generalisation across similar state-action pairs when compared to simpler RL policy representations like tabular methods, it still requires the agent to adequately explore the state-action space. Additionally, deep methods require more training data, with the volume of data escalating with the complexity and size of the neural network. As a result, deep RL requires a long time to collect enough agent-environment samples and to successfully learn the underlying policy. Furthermore, often even a slight alteration to the task invalidates any previous acquired knowledge.

To address these shortcomings, Transfer Learning (TL) has been introduced, which enables the use of external knowledge from other tasks or agents to enhance a learning process. The goal of TL is to reduce the learning complexity for an agent dealing with an unfamiliar task by simplifying the exploration process. This is achieved by lowering the amount of new information required by its learning model, resulting in a reduced overall convergence time.

TL approaches can be divided in Task-to-Task (T2T) and Agent-to-Agent (A2A) transfer. In T2T, an agent with expertise in a specific task partially or totally reuses its learning model or belief to address a different novel previously unobserved task. In A2A, an agent transfer part of its knowledge to a target agent addressing an equally defined task, hence with the same state-action domain and alike reward model. Based on the timing of transfer, A2A can be further classified into online and offline. In online transfer, a novel agent may continuously access knowledge from another agent throughout its entire learning phase. On the other hand, in offline transfer, sharing happens exclusively at initialisation time.

State-of-the-art approaches in online A2A TL follow teacher-student paradigm, in which expert agents transfer their expertise to novices during training through advice-sharing following the teacher-student paradigm. Knowledge transferred can influence either the action decision process of an agent or the learnt weight of an action. Having an optimal teacher to provide advice under the teacher-student framework leads to state of the art performance. In fact, effective transfer relies on the degree of expertise of the teacher. As the student improves its policy, the advice provided by the teacher may become outdated and may overwrite better policies learnt by receiving agents, i.e., a certain advice provided by the teacher although initially helpful, might prevent the agent from exploring better actions. To mitigate the outdated advice shortcoming, previous work introduced advising strategies to regulate the transfer process assuming that an expert agent is used as the source of advice.

This thesis proposes Expert-Free Online Transfer Learning (*EF-OnTL*), a novel framework for experience sharing. It enables online transfer learning in multi-agent systems enabling mutual online knowledge exchange between the learning agents by selecting the most suitable source of transfer at each time. As a result, each target agent receives a customised stream of knowledge tailored to its specific knowledge gaps. Thus, agents are expected to improve their performances by reducing the exploration phase, leading to faster convergence times. *EF-OnTL* is a novel framework for online transfer learning as it facilitates the reciprocal exchange of knowledge across multiple agents without the need for a fixed expert, unlike existing methods that share actions as advice. Furthermore, in *EF-OnTL*, the target's performance is not capped at the teacher's expertise.

Without the presence of a fixed known expert, successful transfer relies on agents correctly and fine-grainedly estimating their confidence in the knowledge samples that they do have. To this effect, this thesis also introduces a new epistemic uncertainty estimator State Action Reward Next-state Random Network Distillation (*sars-RND*): an estimator based on full RL interactions. Compared to a state-visit counter, *sars-RND* enables fine-grained estimation during the training phase by taking into account additional information.

We evaluate *EF-OnTL* across 4 different benchmark environments. First, in three standard RL benchmarks environments of increasingly complexity: Cart-Pole, Multi-Team Predator-Prey, Half Field Offense, and a real-world simulated environment Ride-Sharing

Ride-Requests Simulator. *EF-OnTL* has demonstrated better or equal performance compared to the benchmark TL baselines. We have observed that the degree of improvement correlates with the complexity of the environment addressed. In simpler environments, the improvement is relatively modest, while in more complex ones, the improvement is significantly greater.

Declaration

I declare that this thesis has not been submitted as an exercise for a degree at this or any other university and is entirely my own work.

I agree to deposit this thesis in the University's open access institutional repository or allow the Library to do so on my behalf, subject to Irish Copyright Legislation and Trinity College Library conditions of use and acknowledgement.

I consent to the examiner retaining a copy of the thesis beyond the examining period, should they so wish (EU GDPR May 2018).

Signed:

Alberto Castagna, 18th May 2024

List of Publications

- [1] **Alberto Castagna**, Maxime Guériau, Giuseppe Vizzari, and Ivana Dusparic. Demand-responsive zone generation for real-time vehicle rebalancing in ride-sharing fleets. In 11th International Workshop on Agents in Traffic and Transportation (ATT 2020) at European Conference on Artificial Intelligence ECAI, 2020.

- [2] **Alberto Castagna**, Maxime Guériau, Giuseppe Vizzari, and Ivana Dusparic. Demand-responsive rebalancing zone generation for reinforcement learning-based on-demand mobility. AI Communications , 34(1):73–88, 2021.

- [3] **Alberto Castagna** and Ivana Dusparic. Multi-agent transfer learning in reinforcement learning-based ride-sharing systems. In International Conference on Agents and Artificial Intelligence (ICAART), 2022.

- [4] **Alberto Castagna** and Ivana Dusparic. Expert-free online transfer learning in multi-agent reinforcement learning. European Conference on Artificial Intelligence ECAI, 2023.

- [5] Cengis Hasan, Alexandros Agapitos, David Lynch, **Alberto Castagna**, Giorgio Cruciata, Hao Wang, and Aleksandar Milenovic. Continual model-based reinforcement learning for data efficient wireless network optimisation. European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases, 2023.

Acknowledgements

First and foremost, I would like to thank Martina for taking a leap into the unknown with me in a foreign city two weeks before a global pandemic hit, and for her unconditional support throughout the entire journey and beyond.

I am deeply grateful to my supervisor, Dr. Ivana Dusparic, who is truly one of a kind, for her guidance, encouragement, and availability throughout my doctoral studies. This thesis would not have been possible without her expertise, patience and insightful feedback.

I would like to extend my sincere gratitude to the members of my viva committee, Dr. Enda Howley, Dr. Vinny Cahill and Dr. Siobhán Clarke, for their dedication to improving my work and for the pleasant discussion during the viva. I thank CRT-AI for funding my PhD, and Trinity College Dublin, particularly the DSG team, for welcoming and hosting me.

I extend my heartfelt gratitude to my family for believing in me and for their emotional and financial support throughout all these years of my academic journey.

Last but not least, I am thankful to all the people I have met during these years and who generously contributed their time, expertise, and insights to this research endeavour. In particular, I would like to thank Giuseppe, Maxime, Lara, Mohit, Teejay, Jose, Christian, Francesco and Giorgio.

<div align="center">Alberto Castagna.</div>

Contents

List of Figures

List of Tables

1 Introduction

Reinforcement Learning (RL) is a branch of machine learning that collects a set of algorithms which enable an intelligent agent to learn an optimal behaviour in an unknown task via trial and error [6]. Initially, RL was primarily used in academic settings to demonstrate the feasibility of learning a small unknown task through tabular representation or linear approximators.

However, with the latest advancements in deep learning, RL has seen a sharp increase, as shown in Figure 1.1, where the number of RL publications has grown fourfold between 2017 and 2019, and this growth trend has continued thereafter. The figure shows the publication trend of RL-based research work when looking for *"Reinforcement Learning"* through Google Scholar search engine.

In the latest years, RL has been used to control agents in a variety of tasks, game-based or real-world tasks. Examples of applications include, robotics [11–13], financial [14, 15], asset management/allocation, large-scale optimisation such as, traffic [16–18], 5g networks [19, 20], medical decisions support [21–23], and natural language processing [24, 25].

Despite the remarkable results collected in the aforementioned fields, RL still struggles to be efficiently used in certain real world applications for a variety of reasons, including the complexity of the application domain, settings characterised by non-stationarity, and the continuous evolution of the environment. Next Section further discusses the main issues behind RL applicability.
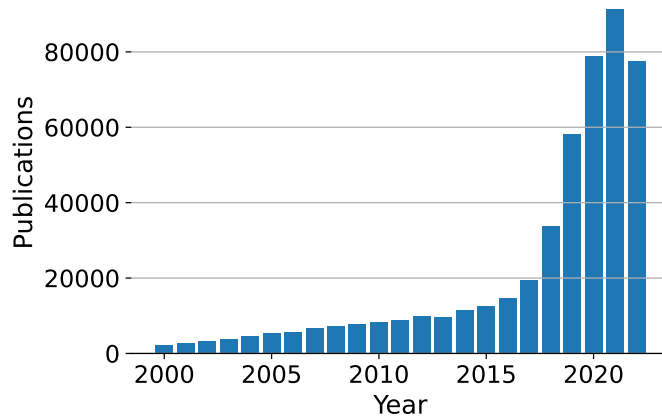
Figure 1.1: RL research work published over the latest years (2000 - 2022) and available through Google Scholar search engine.

1.1 Motivation

RL enables an intelligent agent to learn an optimal decision policy in an unknown task by evaluating its current status. Agent's optimal policy can be defined as a sequence of actions that lead the agent from an initial state to a final state while cumulating the highest possible reward. Such policy is learnt through interactions with an environment. Agent observes the current state, samples an action by following its policy, and finally receives a feedback that expresses how good was the action taken alongside the next observation [6]. States and actions are well defined over observation space and action space, while reward has a dual impact on agent's policy, making it more challenging to model. Reward expresses the immediate return of a single action taken by the agent based on the current environment state. However, the agent also needs to learn the long-term effect of a certain action through cumulative reward. For instance, an action might yield a low immediate reward but in the long term might lead the agent to achieve the best possible cumulated reward as the state is followed by other states that yield high rewards [6].

A major challenge intrinsic to RL is the trade-off between exploration and exploitation [6]. During exploration, an agent samples an action randomly and observes the outcome. Exploration enables the agent to discover new state-action pairs that might yield a higher cumulative return. On the other hand, exploitation enables the agent to consolidate its belief by bootstrapping its current policy. The success of an agent lies in achieving the

balance between these two stages.

An intuitive family of RL algorithms are the Temporal-Difference (TD) learning methods. In these methods, updates to the agent's policy are based on the difference between the estimates at two successive time points [6]. For TD based RL algorithms, policy can be represented through a table where entries are defined over states and actions.

One of the major shortcoming of RL algorithms is the lack of generalisation and flexibility as policy maps state to actions without providing a way of modifying it [6], i.e., expanding observation or control space after a policy is learnt. Instead, if a change in the task is required, the agent will need to start the learning from scratch. Furthermore, when moving towards real-world applications, the state-space complexity increases and two problems arise:

- the increasing complexity of state-space makes tabular methods and linear approximators obsolete and moves the research towards the combination of RL with non-linear approximators [6], such as Neural Network (NN);
- the combinatorial complexity of state-action space introduced a need to bootstrapping knowledge to facilitate the exploration process with ad-hoc instructions, aiding agents in converging towards a successful policy.

The coupling of NN and RL enables the applicability of RL to more complex tasks, and allows a generalisation of the state-space based on past collected experience. As a result, an agent does not need to explore all the possible state-action combinations to infer an optimal policy, as it selects actions based on past information collected across similarly encountered states [6]. The combination of NN and RL shaped a new family of RL algorithms: Deep Reinforcement Learning (DRL).

DRL improves RL thanks to its ability of generalising and approximate state-action space, however, it also inherits deep learning weaknesses.

The deep learning component in DRL makes the algorithm data hungry as it requires a significant amount of agent-environment interaction to learn effective policies. Furthermore, the architecture underneath might be wide and complex. Thus, a NN needs substantial amounts of data to tune layer after layer and to capture patterns. Furthermore, given the amount of data and the number of updating steps to be done on the network, DRL requires

a long time to learn a successful policy. Moreover, a change within the task definition invalidates all the acquired knowledge and requires the agent to learn again from scratch.

To overcome these issues, Transfer Learning (TL) offers a method of enhancing the traditional learning process with external knowledge. TL is a versatile and very broad concept, allowing the integration of external knowledge in various ways within the RL framework. Typically, the flow of transferred knowledge goes from a source agent, which provides advice or knowledge, to a target agent, which learns the task by leveraging the shared information from the source agent. Some applications of TL include:

- using a learnt policy in a novel task [26–37];
- distilling policies from multiple tasks into a single master policy able to achieve similar performance [34–36];
- pre-training an agent with selected experiences prior to allowing the agent to interact with the environment online [38–40];
- enhancing the exploration phase by letting an expert agent act as a teacher to a novice [41–51];
- mitigating the exploration cost by enabling real-time knowledge sharing across multiple agents [52, 53].

In RL the types of knowledge or objects that can be transferred to expedite the learning process, and thus reducing the learning time, depends on the specific transfer objective. For instance, when an expert agent is available to provide on-demand guidance to a novel agent, object transferred is usually an action that is expected to yield a high cumulative reward. Furthermore, the object transferred could be a feedback, given from expert supervisor to the novel agent, which augments the standard reward returned by the environment. The provided signal reflects the expert's estimation of the expected return for a specific action in a given step, based on their knowledge and experience.

In tasks that no agent has previously encountered, there are no experts capable of providing feedback or tutoring novel ones. However, in situations where multiple agents are addressing the same task, they can share selected knowledge, leading to a reduction in the exploration cost for each individual agent.

1.1.1 Multi-Agent Systems

Multi-Agent System (MAS) consists of multiple intelligent agents interacting with an environment while simultaneously learning a policy [54]. These agents can learn independently or collaborate with others in either a centralised or decentralised manner. Depending on the environment's space and the number of agents involved in the task, the communication may be limited to a certain number of agents, taking into account factors such as their distance and communication cost [55–58]. In a centralised setting, each agent shares information during training and contributes towards the update of a common policy [59–62]. When learning independently, each agent optimises its policy merely upon its collected experiences and without any additional interactions with other agents [7, 63]. Independent learning mitigates scalability and communication issues as the learning is distributed and independent, thus reducing the need for synchronisation processes across agents.

On the other hand, Multi-Agent Reinforcement Learning (MARL) algorithms that enable agents to collaborate achieve usually better performance in multi-agent system at a higher communication cost to collaborate, i.e., by sharing data to be used by a common learning process [62, 64–67] or policy parameters to be merged into a centralised one [68, 69].

In real-world scenarios, the n-to-1 communication cost in centralised approaches makes them unsuitable. The time and resources required for agents to communicate and process information centrally result in impractical and lengthy training times to achieve a satisfactory policy.

Communication in Expert-Free Online Transfer Learning (*EF-OnTL*) relies on n-to-n interactions among agents, potentially posing a weak link due to the complexity and resource-intensive nature of such communication requirements.

This thesis assumes fault-free transmission between agents, disregarding potential breakdowns in communication channels to ensure seamless data exchange and collaboration among all system components. Furthermore, in practice, communication may be restricted to neighbouring agents, significantly reducing the overall communication cost and the likelihood of breakdowns in the transmission channels. Finally, should communication disruptions occur, whether due to faulty agents or unreliable transmission channels, *EF-OnTL* is

designed to ensure that each agent retains the ability to continue its learning task autonomously.

1.2   Research Questions

Sharing advice in the form of action from a source agent with an optimal policy, which guarantees the best possible options in any situation, enhances the performance of a target agent and encourages the fulfilment of a goal. However, overriding the policy of a target agent might not be the best option when it is not guaranteed that source has an optimal policy or when no expert is available.

In fact, when a suboptimal agent acts as source of advice, it might limit the target performance on the long term as the provided advice might not lead the target agent to an optimal decision.

Generally, RL is applied to unknown tasks, and therefore, it is not very common to have an agent with a policy that achieves a satisfactory level of performance, let alone an oracle agent that consistently outperforms a learning agent at any step of training. However, if multiple agents are learning the same task simultaneously, they can share knowledge with each other as they learn, thus improving their performance and leveraging their local expertise.

Ilhan et al. [52] have shown a reduced training time by enabling the sharing of actions in MAS where no-expert is available. When an agent seeks guidance, other agents provide action-advice based on the target's state and their individual knowledge.

However, on top of communication and synchronisation cost, overriding the target's policy by forcing it to follow an action suggested by others might not be the best option and could result in delaying the target to converge.

To prevent external agents interfering with a target agent exploration process, this thesis investigates the impact of transferring selected agent-environment interactions in MAS. Agents are homogeneously defined over state and action space with same goal. Furthermore, agents have equal capacity and ability to learn a policy within the given task. When transferring online, no general expert can be identified during the learning cycle. In these

settings, overriding the policy of an agent might result in a delay in the learning process. Thus, following the intuition of pre-training a learning model with selected pre-collected agent-environment tuples, which has already shown a reduced overall training time [38–40], this work enables the share of experience batches from a temporary teacher. Once teacher is identified, the remaining agents have their own knowledge gaps. As a result, target agents need to isolate incoming experience that is expected to fill their current policy gap.

Therefore, this thesis investigates the transfer of agent-environment interactions as a means to enhance the overall system performance in MAS, i.e., reducing training time and improve agents performance. Specifically, the research questions are:

- **RQ1** - *Can, and if so to what extent, online transfer learning through sharing of experience across homogeneous agents with no fixed expert contribute to improving the system performance?*
- **RQ2** - *What criteria can agents use to identify the suitable agent to be used as source of transfer?*
- **RQ3** - *What criteria should an agent use to filter incoming knowledge?*

To fulfil the research aims, the main objectives of this thesis are the following:

- develop a common methodology that enables the selection of an agent, to be used as source of advice, from multiple candidates that has learnt an established policy;
- Investigate multiple criteria that an agent could use to filter incoming knowledge and identify the tuples expected to bridge the gap between the target policy and that of another agent;
- investigate the performance impact of transferring agent-environment tuples to a target agent while varying the frequency and the quantity of shared tuples across diverse environments.

With solid and well-defined research questions to guide this research work and having identified the primary objectives that drive this thesis, the following section introduces the significant contributions that have resulted from this study.

1.3 Thesis Contribution

The main contribution of this thesis is *EF-OnTL*, an expert-free transfer learning framework for online experience sharing between agents in multi-agent context. *EF-OnTL* enables RL-based agents to reduce exploration complexity and hence learn more quickly by sharing knowledge gathered in different parts of the system. Transferred object consists of a subset of experiences, which are past agent-environment interactions, specifically represented as (s_t, a_t, r_t, s_{t+1}) tuples. During each transfer iteration, an agent is selected as source of transfer based on Source Selection Criteria (SS), and a portion of its collected agent-environment interactions is transferred to other agents. Finally, a target agent samples a batch of shared tuples according to a collectively defined Transfer Content Selection Criteria (TCS) criteria computed across source and target agent.

This thesis relies on two criteria to identify worthy experience to transfer, *uncertainty* and *expected surprise*. Uncertainty is independent from the learning form used and aims to analyse the epistemic confidence of an agent. Assuming that all agents use a common uncertainty estimator methodology, the discrepancy between source and target uncertainty, can then be used as metric to identify relevant tuples. Therefore, to estimate epistemic uncertainty from a certain agent-environment tuple, this thesis introduces State Action Reward Next-state Random Network Distillation (*sars-RND*) an extension of Random Network Distillation (RND) [70]. On the other hand, expected surprise [71] is defined over target agent and is approximated through Temporal Difference error (TD-error).

EF-OnTL, the online transfer learning framework introduced in this thesis, represents a novel approach by enabling the reciprocal exchange of knowledge among multiple agents without the need for a fixed expert. This mutual exchange of knowledge uniquely positions it apart from current state-of-the-art methods that facilitate online transfer learning, which generally do not support such dynamic knowledge exchange. Furthermore, while the sharing of experiences is a common practice in offline transfer learning, online contexts generally prefer the sharing of actions.

In contrast to traditional action-advice methods that rely on a fixed expert, where the transfer is typically focused on either the source or the target, *EF-OnTL* introduces a

more adaptive and responsive transfer mechanism. The knowledge assimilated by an agent through *EF-OnTL* is precisely tailored to its specific knowledge gaps, as the algorithm dynamically balances between the source's expertise and the target's needs.

Without the presence of a fixed known expert, successful transfer relies on agents correctly and fine-grainedly estimating their confidence in the knowledge samples that they do have. To this effect, this thesis also introduces a new epistemic uncertainty estimator *sars-RND*: an estimator based on full RL interactions. Compared to a state-visit counter, *sars-RND* enables fine-grained estimation during the training phase by taking into account additional information.

1.4   Evaluation

EF-OnTL impact has been evaluated across 4 different environments of increasing complexity:

- Cart-Pole − to evaluate its impact in a simple minimalistic environment with independent agents [10];
- Multi-Team Predator-Prey (MT-PP) − to evaluate its impact into a collaborative but yet competitive MAS where predators are divided into two teams and a single team is enable to collaborate through experience sharing [72];
- Half Field Offense (HFO) − a complex environment with continuous action space where agents have to collaborate to fulfil their team's goal [73];
- Ride-Sharing Ride-Requests Simulator (3R2S) with real-world data − a simulation of a real-world case study replicating the ride-requests served by yellow and green taxi within the Manhattan area [9]. This environment is unique as long term reward is stronger than immediate and it is hard to establish the quality of an action. Furthermore, the reward function only captures partially the performance of an agent. Thus, an action might seem counterproductive in the short term but might lead the agent to cover more ride-requests while optimising the vehicle usage. The simulator is built upon Simulation of Urban MObility [74] (SUMO) to replicate the same network infrastructure and simulate realistic traffic patterns.

As a preliminary case study, this thesis investigates the offline transfer learning scenario in two environments: MT-PP and 3R2S. Offline transfer has a reduced set of challenges to be addressed when compared to online TL. The transferred object consists of a sub-set of uncertainty-labelled agent-environment tuples collected by trained agents during their training phase. Subsequently, a new set of agents, referred as target agents, spawn and sample a batch of selected experiences from the given buffer before starting their RL training phase. The primary objective of this study is to evaluate the feasibility of positive transfer assuming the availability of an expert and examining the relation between quality of experience, based upon uncertainty, and the transfer outcome.

After demonstrating the feasibility of positive transfer from expert agent, the focus of this thesis shifts towards *EF-OnTL* in the online scenario, where no fixed expert is available. To address the limitation of RND in the online scenario, *sars-RND* is introduce. Subsequently, an extensive evaluation study is conducted on Cart-Pole and MT-PP environments to validate whether the findings from offline TL hold when transitioning to the online setting. The experiments assess various criteria for filtering incoming knowledge on the target agent and investigating the impact of selecting the source of transfer based on different criteria. Finally, the effectiveness of *EF-OnTL* is validated against to selected baseline methods on more complex and real-world scenario, HFO and 3R2S.

To benchmark the impact of *EF-OnTL*, this thesis compares the proposed method against the following TL baselines:

- *No-Transfer* − where agents learn independently without knowledge sharing;
- Online Confidence-Moderated Advice Sharing (*OCMAS*) [52] − where agents share action-advice. The most uncertain agent seeks guidance from others by asking for an action-advice. Consequently, the advice seeker takes the action by majority voting. This baseline is used to compare the impact of overriding target's policy with actions versus sharing selected experience through *EF-OnTL*;
- Requesting Confidence-Moderated Policy (*RCMP*) [42] − where an expert agent is available on-demand to guide a novel agent during its exploration phase. Target agent estimates its epistemic confidence to decide whether to ask for advice. This baseline is used to assess the impact of transferring from imperfect learning agents, selected

dynamically through *EF-OnTL*, versus fixed expert agents.

While *OCMAS* and *No-Transfer* work as upper and lower limit to benchmark *EF-OnTL* against, *OCMAS* should achieve similar level of performance. Although, *OCMAS* has a greater communication cost as it requires all the agents to synchronise at each decision-step to verify whether an agent is the one with lowest uncertainty within the group.

1.5 Dissemination

As part of the dissemination process, the findings of this research were presented in various venues.

Starting with Castagna et al. [1], presented at the 11th internation workshop on Agents in Traffic and Transportation (ATT 2020) at ECAI 2020 conference, and its extended journal version Castagna et al. [2], published in AI Communication. These two publications serve as motivation work and groundwork to investigate the optimisation problem of mapping ride-requests to vehicles while enabling ride-sharing and TL. Specifically, we had 200 vehicles utilising RL agents to learn, which gave rise to intuition that we need a knowledge exchange mechanism for agents to enhance each others learning since all were learning to accomplish the same task. Furthermore, our research aimed to develop a scalable solution that could allow for the dynamic allocation of vehicles and could be resilient to minor changes in the road-network and demand patterns. This problem sparked our interest towards TL to enable the transfer of piece of knowledge or entire policies to new agents to adapt to novel scenarios.

Castagna and Dusparic [75] in 2022, published in the proceedings of the 14th Internation Conference of Agents and Artificial Intelligence (ICAART), discussed the preliminary evaluation on experience sharing in an offline context, Chapter 3 is almost based on this publication.

The publication that mostly fully encompasses final contributions of this thesis is Castagna and Dusparic [4], published in the proceedings of the 26th European Conference on Artificial Intelligence (ECAI 2023). The core contribution of this thesis, *EF-OnTL*, and presented a subset of the evaluation reported in Chapter 6.

Another publication related to TL but not specifically *EF-OnTL*, resulted from my PhD

internship in the Intelligent Automation team at the Huawei Ireland Research Center. This collaboration led to a research paper, Cengis et al. [5], published in the proceedings of the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2023). This paper demonstrates a continual learning approach to facilitate policy adaptation in wireless networks with an expanded control space.

1.6   Roadmap

To summarise, this chapter introduced TL as a means to mitigate some of the weaknesses typical of DRL models, i.e., the lack of generalisation and extensive training time required to adequately explore an environment and learn an effective policy. In detail, $EF\text{-}OnTL$ enables the sharing of selected experiences among a set of imperfect agents that are learning a certain task simultaneously. The following chapters of this thesis are structured as follows: in Chapter 2, a comprehensive exploration into the background of RL and TL is given, presenting open challenges and the latest research work in these areas. Subsequently, before introducing the design details of $EF\text{-}OnTL$, Chapter 3 shows a feasibility study, an offline transfer context, by presenting preliminary results to assess whether experience sharing can effectively improve target performance. Further, Chapter 4 introduces and discusses $EF\text{-}OnTL$ while providing details about the design choices made in its development. Then, Chapter 5 intricately examines the foundational technologies that underpin $EF\text{-}OnTL$, and provides an insightful analysis of the simulators employed to comprehensively evaluate its capabilities. To follow, Chapter 6 introduces the evaluation objectives and the evaluation studies. It meticulously analyses the findings in relation to the research questions posed in the thesis. Finally, Chapter 7 concludes this thesis by summarising its contributions and outlining open challenges that need to be addressed in future work.

2 Background and Related Work

This chapter is organised in three main sections, each covering a specific aspect of the background work addressed in this thesis. Firstly, Section 2.1 presents the background of Reinforcement Learning (RL) and deep RL introducing also the challenges and potential limitations when applied to complex tasks. Then, Section 2.2 reviews transfer learning to overcome the limitations of deep RL. Finally, Section 2.3 concludes the chapter by clarifying the concept of uncertainty in RL and reviewing the techniques used to estimate uncertainty in RL.

2.1 Reinforcement Learning

A Markov Decision Process (MDP) is a stochastic sequential control process used to model decision-making [76] based on the Markovian property which states that the probability of going from the current state at time t to another at time $t+1$ is independent by the history of previous visited states: $P(s_{t+1}|s_t, s_{t-1}, \ldots, s_0) = P(s_{t+1}|s_t)$.

A MDP is defined as a 5-tuple (S, A, T, R, S_0).

- $S -$ is a collection of states that could be visited. S could be finite or not;
- $A -$ is a set of actions and can be either finite or not;
- $T - T : P(s_{t+1}|s_t, a_t)$ defines the transition probabilities between states, specifically, the probability that an action a_t leads to s_{t+1} from a given state s_t;
- $R - R(s_t, a_t, s_{t+1})$ expresses the immediate reward for transitioning from s_t to s_{t+1} through a_t;
- $S_0 -$ starting state set is the set of possible starting states.

Based on the observed state, a MDP can be classified as *Fully-Observable*, when the

state representation passed to the agent is accurate and reliable [6]. *Partially-Observable Markov Decision Process (POMDP)*, when an observation is an estimation of the current state [77].

As a MDP is a sequential decision process, the action-decision process over a sequence of consecutive time steps can be defined as Horizon (H) [6]. Based on the length of H, a MDP can be categorised into three categories: 1) finite-horizon tasks, where the number of action-decision is fixed to a certain number of time steps; 2) indefinite-horizon tasks, where the number of time steps are upper limited by a certain fixed value; and 3) infinite-horizon task, in which interaction is continuous and does not terminate.

For horizon of length n, it exists an associated *Discounted Reward (R_n)*. R_n is the sum of immediate rewards r_k, received at time step k, weighed by a discount $\gamma \in [0, 1]$, hence: $R_n = \sum_{k=0}^{n-1} \gamma^k r_k$. γ regulates the impact of future reward versus the greedy reward received at the current time step.

When γ is close to 0, the immediate reward holds more significance, and the influence of future rewards diminishes significantly. As a consequence the decision process prioritises greedy choices. On the other hand, when γ is close to 1, future rewards weigh more into the decision process and lead to prioritise a long-term planning that aims to maximise the overall return.

A MDP provides a mathematical framework to model tasks that require sequential decision-making. Among the requirements to define a task as a MDP is the availability of both the transition function T and the reward function R that define how the environment responds to the actions taken by an agent. However, transition function is often unknown. To overcome this limitation, RL enables an agent to learn the transition function by randomly interacting with the environment.
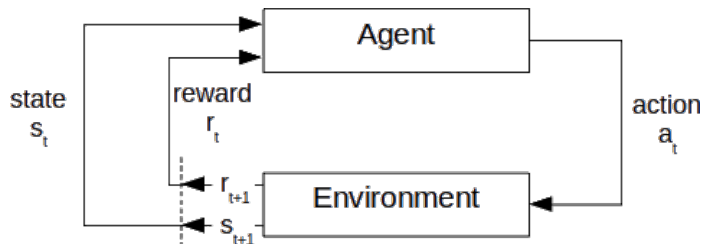
Figure 2.1: Single step of Reinforcement Learning process [6].

RL is a cyclic four-step process shown in Figure 2.1. These four steps are repeated at every time step t:

1. Observation − an agent observes the surrounding environment and along with its internal state maps it to the current state s_t;

2. Action Selection − the agent selects an action a_t to take based on perceived state;

3. Actuation − the agent takes a step within the environment by executing a_t sampled earlier;

4. Evaluation − an agent observes the updated observation of the state s_{t+1} alongside a signal r_{t+1} that expresses how good was the action a_t taken from state s_t.

RL can be categorised as *model-based* and *model-free* [6]:

- model-free − it enables an agent to learn a policy just upon the experience (s_t, a_t, r_t, s_{t+1}) collected by interacting with the environment;

- model-based − it enables an agent to learn a model of the environment, including the transition dynamics and the reward function, and use that model to plan actions and maximise the expected reward from the taken action. Hence, model-based learns by planning [6].

This thesis focuses on model-free RL.

Learning in RL can be represented as *value*-based and *policy*-based. Value-based RL uses agent-environment interactions to update a single value, e.g., table entry, representing the state-action combination. When exploiting its internal policy, an agent selects the best action as the one with highest value across all actions for the current state. On the other hand, in policy-based RL, an agent uses experience to optimise hidden parameters of a function that maps states to actions through probabilities. Over time, an agent updates these parameters to maximise the expected reward via gradient-based optimisation methods. For a certain state, an agent will then follow an action with maximum probability.

Actor-Critic methods combine both policy and value representations. These algorithms are composed by two parts: an actor, which maps states to actions and is used to select an action from a state; and a critic, which provides an estimation of the state-value, representing the expected discounted cumulative reward obtained by following the policy from the

observed state [78].

During the learning process, an agent needs to balance two phases: *exploration* and *exploitation*. While exploring, the agent samples random actions to visit new state-action combinations. On the other hand, exploitation enables the agent to strengthen its belief by leading the agent to follow its policy. At the initial stage of the learning phase, an agent follows merely a trial-and-error approach. Over time the probability of taking random actions decreases in favour of exploiting the learnt behaviour.

One of the ways to exploring new state-action pairs is by following an ϵ-greedy action selection [6], i.e., an agent samples an action by following its policy with a $1 - \epsilon$ probability and a random action otherwise. ϵ slowly decays to 0 over time reducing the probability of exploring in favour to exploiting. ϵ anneals by the ratio between the difference across starting ϵ (ϵ_{start}) and ending (ϵ_{end}) and the total number of episodes that the agent will undertake, hence $\frac{\epsilon_{start} - \epsilon_{end}}{episodes}$. Generally, ϵ is initialised to a value close to 1 and decays towards 0, e.g, 0.95 to 0.05.

For a MDP with low-dimensional state and action space, RL can be expressed using a tabular representation where rows and columns are defined over possible states and actions. Their intersections are updated based upon the value returned by the environment with Eq. 2.1. This family of RL algorithms is called *tabular methods* and one popular method is *Q-learning* [79].

$$Q(s_t, a_t) = (1 - \alpha)Q(s_t, a_t) + \alpha\left(r_t + \gamma \max_a Q(s_{t+1}, a)\right) \qquad (2.1)$$

Learning rate α weights the impact of new knowledge ($r_t + \gamma \max_a Q(s_{t+1}, a)$) against the current agent knowledge ($Q(s_t, a_t)$). α is defined in the interval of $[0, 1]$. When α is 0, the knowledge of the agent is not influenced by incoming information. On the other hand, setting α to 1 the agent fully overrides its knowledge with the newly information retrieved. γ is the *discount factor* and is defined within the $[0, 1]$ interval. It weighs the impact of immediate reward and future rewards.

Given a state and an action, its *Q-value* ($Q(s, a)$) is updated by a value computed over the received reward at time t and the maximum possible Q-value from the following state.

RL has demonstrated significant potential in tackling various tasks in different domains, including robotics [80–83], game playing [84, 85], optimisation tasks [16, 86]. However, there are a number of open challenges that limit the applicability of RL. The following section introduces the open challenges in RL.

2.1.1   Reinforcement Learning - Challenges

RL has proved to be a viable approach to learn a successful policy in tasks where feedback is available to judge single-step action. However, a number of challenges remain, some of which are outlined below:

- *Exploration-Exploitation trade-off* − Exploration and exploitation are fundamental processes for RL and a right balance is required to achieve a nearly optimal policy. The agent needs enough exploration to visit action-state combinations and to observe their outcome. On the other hand, the agent needs to exploit its policy to strengthen the learnt knowledge by taking the sequence of actions, expected to be optimal, and to observe the long term horizon outcome. A right trade-off is required depending on the ongoing learning phase. At the beginning, an agent is expected to prefer exploring than exploiting while eventually exploitation takes the lead. Balancing these two phases is challenging as it depends on the complexity of state-action space combination.

- *Reward Function Design* − A reward function, or reward model, supports a RL-agent to learn a policy by providing feedback after a single-step interaction. In the long-term, an agent is expected to learn a policy that maximises the long-term cumulative reward. Thus, a reward model plays a crucial role towards the definition of optimal policy. While these functions are usually well defined in games and benchmarking environments, i.e., OpenAI Gym [87], when it comes to real-world, it is very likely that a reward function is not available. For instance, in a ride-sharing enabled mobility on-demand system, the reward model could prioritise the revenue maximisation, could punish long delays due to re-routing and enabling ride-sharing or could support the vehicle utilisation. Depending on the desired goal, the reward function should be designed to lead the agent towards the accomplishment of the primary goal while

having a fair trade-off on the secondary goals. Thus, designing a proper reward function demands careful attention and deep knowledge of the addressed task [88].

- *Sparsity of Rewards* − Sparse-reward tasks provide feedback to an agent very infrequently, and in a lot of cases only upon the accomplishment of a goal. Consequently, for most state-action pairs, the agent will not receive any feedback. To expedite the learning process, it may be better to enhance the reward model by providing earlier feedback.

- *State-Action Dimensional Complexity* − Complexity of a task is directly influenced by its state and action space [88]. A larger set of actions, requires the agent to explore an increased number of state-action combinations.

- *Lack of Generalisation* - RL optimises a policy to a specific task. A change within any of the MDP tuple is very likely to invalidate some or even all acquired experiences. Furthermore, agents must learn a generalised policy allowing for various environment configurations. For instance, considering a mobility on-demand context where a RL agent controls a vehicle aiming to maximising the number of passengers served. In this scenario, the agent needs to learn a policy that can effectively adapt to different demand patterns, ensuring optimal performance across a diverse range of request distributions and traffic conditions.

- *Exploration Cost* − On top of the exploration-exploitation trade-off, exploring towards a real-world task can be dangerous and expensive. When exploring actions in a real-world a physical agent requires certain time to fulfil an action and observe the outcome, introducing a delay, [88]. Furthermore, safety concerns need to be taken into account when performing arbitrary actions. A failure might lead the agent to damage its system or external entities [89].

- *Partial-Observability and Non-Stationarity* − Real-world sensors might provide information that are not fully reliable because of environment conditions or malfunctioning [88]. Furthermore, real-world systems are often influenced by external factors or stochastic elements [88]. Therefore, the transition dynamics of the environment might change over time leading the task to be non-stationary.

Tabular representation for RL limits the applicability to domains with low-dimension

state-action spaces due to the combinatorial complexity. However, discretisation functions are often combined with tabular RL to enable the handling of high-dimensional and continuous state-action spaces. As a result, the combinatorial complexity is lowered and tables can be used to fit the state-action combinations. Nevertheless, that might still not be sufficient to learn a successful policy for complex tasks, for example, a self-driving car.

Tabular RL lacks generalisation across similar state-action pairs as each update within the model is uniquely related to a specific action and state combination. To overcome the generalisation limit, intrinsic in tabular representations, researchers have focused on combining Neural Network (NN) and RL.

2.1.2 Neural Networks

An Artificial neural network is a learning model inspired by a human brain. The first tracked attempt towards their development is dated back to 1943. Warren McCulloch and Walter Pitts [90] proposed a model of a neuron and demonstrated how multiple neurons could be interconnected to perform simple logical operations. However, the wider development and use of NNs started from 1986 when Rumelhart, Hinton and Williams [91] introduced a function to update an internal representation by back-propagating errors. From early 2000s, NNs have gained popularity due to their successful applications in various topics, including classification, clustering, patter recognition and prediction across multiple sectors [92].

NNs are composed of interconnected layers, with each layer containing a specific number of neurons. Each neuron computes a linear function as follow:

$$y = w * x + b \tag{2.2}$$

The bias term b is a constant value that allows the neuron to have an effect even when the input signal x is 0. w is a scalar value that weighs x and y is the output of the neuron. w and b are adjusted during the training process based on the output of the network. The output of a layer is defined as collection of individual neuron outputs belonging to that specific layer.

The output of a layer is processed through an activation function to introduce non-

linearity and enable the network to learn complex relationships. Some of the most common are *ReLU* [93], *TanH* [94], *Sigmoid* [95] and *SoftMax* [96]. Each activator applies a function and decides whether a neuron output is active or not. For instance, the *ReLU* activation function clips any negative value to 0 and leaves any positive input unchanged.

A NN is a sequence of layers and activation functions. Commonly the computational flow follows a sequential direction from first layer to the second and so on to the output layer. That is known as *forwarding* step. Computation can be divided into multiple streams at any point of the network and layers might be connected in different ways according to specific needs.

There are multiple type of layers available based on their inner function. *Linear* or *Dense* layer using Eq. 2.2, *Convolutional* layers, that are used to process multi-channel input such as images and finally, *Recurrent* layers, that enable a model to retain additional information by building an inner memory for past events.

Defining the internal NN structure is the first step towards deploying a NN to tackle a task. Although, defining the architecture along with the activation functions is not enough to design a learning model. All parameters inside the layers need to be adjusted during a training process to learn some high level features and to generate the desired output.

The process of updating the parameters based on the training data is known as *back-propagation* [91]. Back-propagation enables a NN to learn from the error produced by the model. Back-propagation consists of computing the loss between predicted output and the true outcome. The network error is computed through a *loss function*, such as Mean Squared Error (MSE) defined at Eq. 2.3.

$$MSE = \frac{\sum_{i=1}^{n}(y_i - \widehat{y_i})^2}{n} \tag{2.3}$$

$\widehat{y_i}$ is the true outcome and is compared against the network output y_i.

The computed error is propagated backward through the network, from last layer to the first, to update the parameters using an optimisation algorithm, e.g., *Adadelta* [97], *Adam* [98] or *RMSprop* [99].

When a NN has 3 or more hidden layers, it is generally classified as deep NN [93]. Having

multiple hidden layers enables the learning of more complex representations. Thus, deep NN have established a new state-of-the-art performance across multiple fields such as speech recognition, visual object recognition, object detection, drug discovery and genomics [100].

The rapid advancement in deep learning and NN enabled researchers to shift from RL to Deep Reinforcement Learning (DRL). DRL is the result of merging deep NN with RL enabling the tackling of increasingly complex tasks that were previously intractable due to the constraints of combinatorial state-action spaces [101].

2.1.3 Deep Reinforcement Learning

Deep Reinforcement Learning (DRL) mitigates the challenges of RL by using a deep NN to learn the mapping between states and actions. DRL enables the learning of an optimal policy in tasks with high-dimensional state-action spaces and furthermore, it generalises across similar states and actions. In DRL, an artificial network is used to learn the mapping between states and actions. Therefore, the agent in Figure 2.1 is equipped with a NN that is used to learn the high-level features that are used for the action decision process. Figure 2.2 shows the agent-environment interaction with a NN.
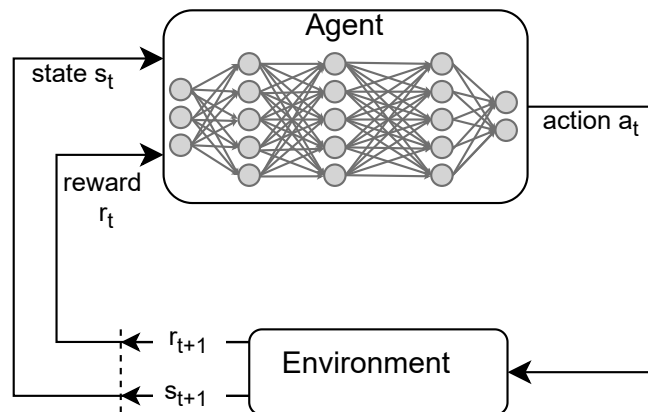
Figure 2.2: Single step DRL process with deep NN.

2.1.3.1 Value-based Methods

Value-based DRL methods use a NN to approximate a value function used to select the best action to take given a state. One of the first algorithms is Deep Q-Network (DQN),

an evolution of Q-learning where tabular representation is replaced by an artificial neural network.

The first DQN version consists of a single deep NN used to approximate the value function, presented in 2013 by Mnih et al. [101], to solve seven Atari games using raw pixels as input. The use of a single online NN to estimate the value from the next state (Q-value) and select the optimal action resulted in instability during the learning process. Although the algorithm achieved state-of-the-art results in six out of the seven games tested, the training process was not stable. Instability is caused by the correlation between consecutive observations and non-stationarity of the environment. In 2015 Mnih et al. [102] overcame the instability issue by introducing an *experience replay buffer*, to break the correlation of consecutive observations, and a *target network*.

An experience replay buffer consists of storing n agent-environment interactions and randomly sampling a subset during training. Furthermore, an experience replay buffer improves sample efficiency by allowing an agent to reuse previously visited interactions multiple times without requiring redundant sampling of the same action within the same state.

The target network shares the same structure of the online network and its parameter are periodically updated based on the online parameters of the network. Two methods, namely *hard-copy* and *soft-copy*, are used to update the target network. In the hard-copy method, the parameters of the online network are copied to the target network after every n learning iterations. On the other hand, in the soft-copy method, the target network's parameters are slowly updated towards the value of the online network parameters during each learning iteration. This update is controlled by a parameter τ, which regulates the strength of the update. While online network is optimised, the target network is frozen. Having a target network to estimate the Q-values from next state reduces the correlations with the Q-values, on the current state, estimated by the online network and therefore helps achieving a stable learning process [102]. DQN loss function ($\mathcal{L}$) using two networks is reported in Eq. 2.4, where $Q(s, a)$ is the Q-value obtained using the online network while $\widehat{Q}(s, a)$ is the target estimation.

$$\mathcal{L} = MSE\Big(Q(s_t, a_t), (r + \gamma \max_{a_{t+1} \in A} \widehat{Q}(s_{t+1}, a_{t+1}))\Big) \tag{2.4}$$

Despite the improvements brought by the use of a target network and experience replay buffer, DQN remains computationally demanding to find an optimal policy. Additionally, while experience replay improves sample efficiency, it relies on random sampling, which may result in insufficiently capturing the crucial experiences. The uniform probability of selecting each tuple means that significant experiences that hold a high impact on learning might not be adequately processed by the underlying learning model.

Furthermore, DQN suffers *overestimation bias* as it uses the maximum Q-value over all possible actions in the next state ($\gamma \max_{a_{t+1} \in A} \widehat{Q}(s_{t+1}, a_{t+1})$). Selecting the maximum Q-value can lead to overestimation of the Q-values where the true Q-values are high [103].

To overcome the aforementioned issues and thus to improve the learning stability of DQN, *Double DQN*, *Prioritised Experience Replay* and *Dueling networks* have been introduced.

Double DQN modifies Eq. 2.4 by using the online network to select the next action a_{t+1} while taking its associated Q-value from the target's. Thus, $\widehat{Q}(s, a)$ is estimated as $\widehat{Q}\Big(s_{t+1}, \operatorname*{argmax}_{a_{t+1} \in A} Q(s_{t+1}, a_{t+1})\Big)$. Using the online network to select the next action and taking the Q-values estimated by the target network overcomes the overestimation bias [103].

Prioritised Experience Replay improves the random sampling of experience replay buffer by assigning a weight to each visited agent-environment interaction. Thus, experiences that surprise the most will have a higher probability to be sampled. In the context of Temporal-Difference (TD) learning models, surprise is commonly defined as the TD error. As the learning progresses, the surprise values are updated based on the training loss.

By prioritising selected experiences that held more information for the learning model, prioritised experience replay leads to accelerated learning and improved performance for the agent [104].

Dueling Q-network modifies the architecture of DQN by forking computation stream into two branches as shown in Figure 2.3. The underlying idea is to estimate the intrinsic value $V(s)$ of being in a certain state s and the advantage $A(s, a)$ of taking a certain

DQN
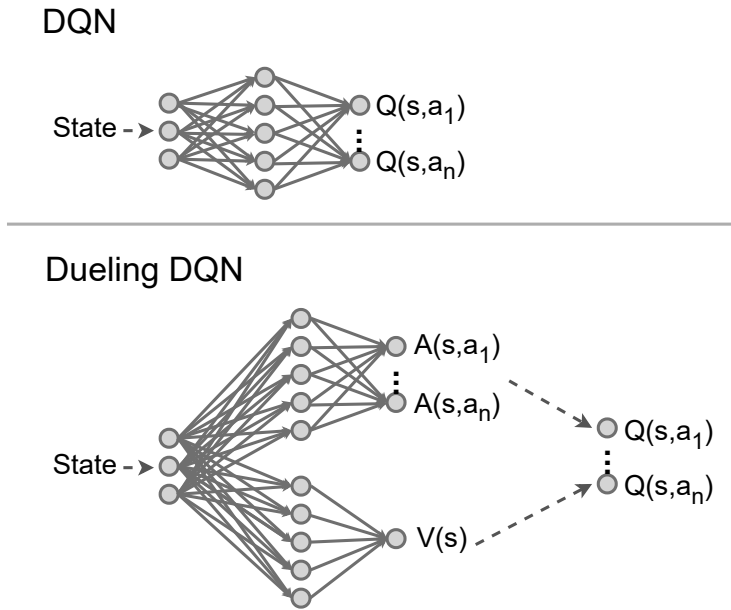
Dueling DQN

Figure 2.3: DQN architecture at the top and dueling DQN architecture at the bottom.

action a from s. The decomposition of Q-values into state-value and action advantages, allows the agent to identify states that are valuable and to learn the advantage brought by each action. The decoupling improves the learning efficiency and stability by enabling more efficient exploration and exploitation of the environment. However, forking the architecture into two streams is not sufficient to decouple the two estimators. To ensure that the model will learn the decoupled representations of $V(s)$ and $A(s,a)$, the mean value of advantages has to be zero. Such constraint need to be included within the loss function, usually done through advantage normalisation, to ensure that the model focuses on the relative differences between actions. For dueling Q-network, the Q-values can be defined as the sum between normalised advantage and state value as reported in Eq. 2.5 [105].

$$Q(s,a) = V(s) + A(s,a) - \frac{1}{|A|} \sum_{a_i \in A} A(s, a_i) \qquad (2.5)$$

2.1.3.2 Policy-based Methods

Policy-based methods focus on learning a policy, which is a mapping function from states to actions. The learning model outputs a probability distribution over actions from a given state. When exploiting the policy, an agent takes the action with highest probability in

a certain state. The distribution is given by the output layer and therefore is affected by the network parameters. Policy-based methods enable the agent to explore based on the probability distribution defined over the actions from states.

Policy-based methods are generally optimised using policy gradient methods. The gradient (∇J), computed with Eq. 2.6, identifies the direction in which parameters should be optimised to maximise the expected cumulative reward.

$$\nabla J = \mathbb{E}[Q(s, a) \nabla \log \pi(a|s)] \tag{2.6}$$

Policy-based methods can be divided into *Off-policy* and *On-policy* based on how they handle the collected agent-environment interactions. On-policy methods, update the learning model based on most-recent gathered experiences collected by the latest version of the policy. A batch of samples is collected, used for updating the policy, and then discarded. The constraint of using collected experience only once can be a limitation of on-policy methods given that they cannot improve sample efficiency with a replay buffer.

2.1.3.3 Actor-Critic Methods

Actor-Critic combines value-based and policy-based methods. These algorithms are composed by two components, an actor which is learning a policy, and a critic, estimating a value function. The two most common Actor-Critic algorithms in DRL are Proximal Policy Optimisation (PPO) [7] and Deep Deterministic Policy Gradient (DDPG) [8].

PPO is an actor-critic on-policy algorithm that updates the policy gradient using the trust region policy optimisation method. The trust region method ensures that the probability ratio (r_t), defined as the ratio $r_t(\theta)$ between the probabilities of selecting a certain action under the new updated policy π_θ and the old policy $\pi_{\theta_{old}}$ ($r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)}$), lies within a certain interval. This constraint prevents sudden changes between the updated and old policies, ensuring stable and controlled policy updates.

A common approach to enforce the interval constraint on $r_t(\theta)$ in PPO is by using the clipped version defined in Eq. 2.7. This function is used to compute the clipped loss L^{CLIP} during the training process[7].

$$L^{CLIP}(\theta) = \widehat{\mathbb{E}}_t\left[\min\left(r_t(\theta)\widehat{A}_t, clip(r_t(\theta), 1-\epsilon, 1+\epsilon)\widehat{A}_t\right)\right] \qquad (2.7)$$

$\widehat{A}_t$ is an estimator of the advantage function at time t that expresses how good was a certain action taken in a given state. The first term inside the min $(r_t(\theta)\widehat{A}_t)$, is to encourage the policy π_θ to take actions that maximises the expected return. However, as $(r_t(\theta)\widehat{A}_t)$ might lead the updated policy to shift too much from the old one, the second term constrains $r_t(\theta)$ within an ϵ-interval. Here, ϵ is the clipping factor, which ensures that the updates to the parameters θ are kept within a controlled range during training, ensuring stability in the learning process.

The PPO clipped version objective function balances between improving the policy and preventing sudden changes that may lead to instability or divergence. While the PPO actor is updated using a surrogate function, i.e., Eq. 2.7, the critic is updated by MSE between the predicted state-value and the true discounted rewards obtained from the collected agent-environment data. Figure 2.4 shows PPO architecture with actor and critic networks.
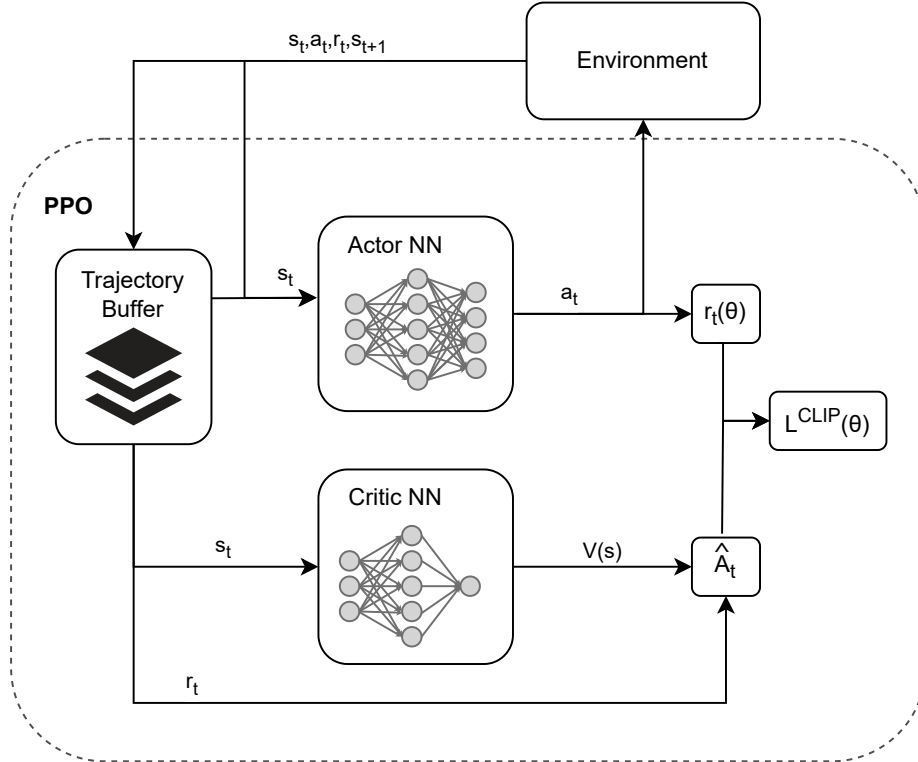
Figure 2.4: PPO architecture [7].

DDPG is an actor-critic off-policy method used for tasks with continuous control space.

Both actor and critic are implemented through a deep model. Actor is used to estimate an action that maximises the estimated expected reward from a certain state. Critic estimates the state-action value used to evaluate the actor performance. The output of the actor is the mean value of a Gaussian distribution that is used to generate the action to take from a given state. To ensure exploration, action sampling is enriched by a noise value $\mathcal{N}$ sampled by a distribution: $\mu(s_t) = \mu(s_t|\theta_t^\mu) + \mathcal{N}$. First part of the sum is the output of the actor for state s_t and with the current network parameters used to compute the mean. DDPG is off-policy as it uses an experience replay buffer to gather experience and sample afterward a batch of tuples to break the correlation between consecutive samples and improve the learning stability [106].

Although DDPG is a popular algorithm for continuous control tasks, there are improved versions that might be more suitable for different tasks requirements.
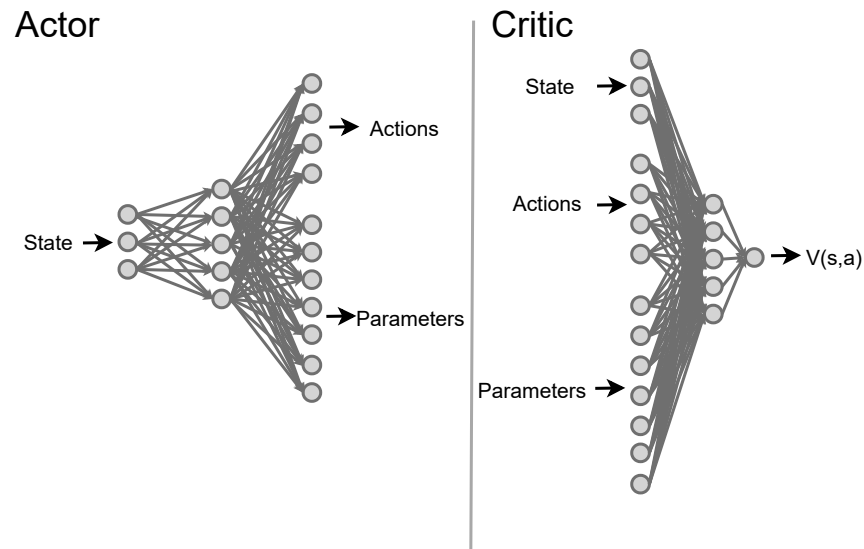
Figure 2.5: Parameterised Action Deep Deterministic Policy Gradient (PA-DDPG) [8] Actor-Critic architecture.

In scenarios where a single action controls multiple underlying parameters, which are often defined within a continuous domain, *Parameterised Action Deep Deterministic Policy Gradient (PA-DDPG)* [8] extends DDPG for such tasks with parameterised action space. PA-DDPG actor has two output layers and selects simultaneously a discrete action and the parameters needed to parametrise the chosen action, as shown in Figure 2.5. The critic computes state-action value from visited state and the outputs from the actor network [8].

2.1.4   Deep Reinforcement Learning - Challenges

DRL enables an agent to tackle more complex tasks and perform as well as traditional RL methods when applied to simple tasks. Therefore, DRL overcomes the state-action dimensional complexity and mitigates the lack of generalisation by generalising across similar state-action tuples. However, DRL leaves unsolved the other RL weaknesses, introduced earlier in Section 2.1. The unsolved challenges that remain to be addressed are the following: 1) exploration-exploitation trade-off; 2) reward function design; 3) sparsity of rewards; 4) lack of generalisation, across similar but different tasks; 5) exploration cost; 6) partial-observability and non-stationarity.

Furthermore, the use of deep NN to learn the state-action mapping introduces two additional challenges to DRL.

- The capacity of a deep NN learning model is influenced by the number of parameters present within it. Increasing the number of neurons and layers expands the model ability to learn a wide range of features. Although, it is important to note that while larger networks are capable of addressing more complex problems, an agent may not need that much capacity for a certain task. In general, larger NN require a larger amount of data compared to smaller networks in order to achieve similar level of performance for a given task. Hence, a trade-off is required to ensure sufficient granularity while keeping the total number of neurons as low as possible. In addition to network width and depth, the choice of layer types used within the network also impact the network size. Different layers require different amount of parameters to be optimised during the learning phase.
- Deep NNs are generally known to be data-hungry as they require enormous amount of data to optimise the inner parameters. Furthermore when coupled with RL, impact of new knowledge is usually scaled by the learning rate and that amplifies the neediness of data.

To address the limitations of DRL, one effective approach is to grant the learning agent access to an external source of knowledge. This process is known as Transfer Learning (TL) and, when applied to DRL, some of its benefits are the following:

- TL improves the sample efficiency by transferring crucial experiences across multiple agents;

- TL improves the generalisation lack by re-using policies across similar tasks;

- TL lowers the learning complexity related to deep models by adapting previous acquired policies;

- TL lowers the exploration cost by guiding the exploration of an agent.

The next section delves into the state of the art of TL applied to DRL to overcome its limitations.

2.2 Transfer Learning

TL is a branch of Machine Learning (ML) that focuses on the reusability of knowledge acquired in various contexts, tasks, or by different agents. When applied to RL, it can reduce the learning time to learn an effective policy by enabling knowledge reuse. The broad idea behind TL applied to RL is to optimise resources consumption, i.e., computation power and time required to learn a policy, by re-using existing knowledge rather than learning from tabula rasa and updating the knowledge by re-sampling agent-environment interactions.

In TL, the communication can be conceptualised as a one-way channel from a *source* agent to a *target* agent. The goal of TL is to use source knowledge to speed-up the learning process of a target by reducing, and in some cases removing, the exploration complexity.

Transferring knowledge from source to target can result into two possible outcomes. *Positive transfer* occurs when the target agent shows an improvement upon some metrics compared to tabula rasa learning. On the other hand, *negative transfer* occurs when the use of external knowledge delays or hinders the learning process of the target agent. This negative transfer might happen when the transferred knowledge is outdated or when the source and target task are defined on incompatible MDPs.

Transferring in RL can be divided in two categories: *Agent-to-Agent (A2A)* and *Task-to-Task (T2T)*. In A2A TL, both the source and targets agents address tasks defined over the same MDP. Thus, no mapping is needed to transfer knowledge from one agent to another. On the other hand, T2T TL occurs when the source and target tasks are defined over

different MDPs. In T2T, a mapping function is required to enable the transfer of knowledge across the tasks or different agents.

2.2.1   Task-to-Task Transfer Learning

Transferring from a known source task to a previously unseen task is a pillar concept towards the development of a *lifelong learning* model where an agent has to adapt throughout time to fit an evolving task. For instance, a task might evolve on the control space by expanding the action space.

The objective of T2T TL is to fill the lack of generalisation typical of RL approaches. When applied to DRL, the high data dependency is reduced and the training time, required to learn an effective policy, is reduced.

The complexity of transferring across tasks results from their MDPs discrepancy. Tasks might differ by the state and action domains, the environment dynamics T or the reward model [107]. As examples of these challenges,

- $S_s \neq S_t$ − Two tasks can have a different representation of the same state-space [26, 27, 29, 30, 39, 108, 109], e.g., graphical representation vs raw feature vectors, or even different sensors. A policy may need to be resilient to sensors failure or upgraded sensors.

- $A_s \neq A_t$ − Over time a task may need to adjust the control space dimension. As a result, tasks might have a different set of actions available to the agent. [26, 27, 30, 34–36]. As an example, $A_s \subset A_t$ or $A_t \subset A_s$ or $A_s \cap A_t \neq \emptyset$.

- $R_s \neq R_t$ - Two tasks can share the same state-action space but have distinct reward structures or even divergent ultimate objectives [26–32, 37, 110].

- $T_s \neq T_t$ - Two tasks defined within the same domain can differ in their probability transition function T. Consequently, an action from a particular state may result in a different next state for each task.

An effective approach for T2T knowledge transfer involves leveraging an existing learning model for a new task. Thus, the model can be utilised partially or entirely for the new task [26–37].

When it is not possible to re-use the model but sampled agent-environment interactions are available or provided by a human demonstrator, a mapping function can be defined to map incompatible features of source and target task [39, 109–113]. Thus, samples gathered in one task can be adapted and re-used in a novel task to pre-train a learning model. Another application of T2T TL is when a task can have multiple different representation forms for the current state, a latent space can be learnt to enable cross-modal input [108].

2.2.2   Agent-to-Agent Transfer Learning

Multiple agents addressing the same task, or tasks defined over the same MDP, are generally defined as homogeneous agents. An example of a system with multiple homogenous agents are multi-agent systems where multiple agents interact with the same environment to achieve a shared goal. This section overviews challenges and methodologies to enable knowledge transfer across homogeneous agents.

The primary objective of A2A transfer learning is to expedite the learning process for an agent by integrating external knowledge, thus reducing the overall learning time compared to learning without additional input. To achieve this objective, A2A TL aims to minimise exploration complexity and enhance the overall performance of the system.

A common trait observed in A2A TL is the use of external expertise to influence the learning or action decision process of a target agent, as opposed to adapting an existing optimal model as in T2T. The external input can be provided by either a human or an artificial agent, and the knowledge provided can be integrated into the exploration process or can be used as initial phase prior to exploration.

An approach to categorise A2A work is by analysing the shared object, which can be in the form of the following:

- *Action* - an agent provides on-demand advice to an exploring novel agent. Advised action overrides the target policy and ideally leads the novel agent towards the expected objective [41–45, 48, 50–52, 114, 115].
- *Feedback* - an expert provides feedback, in the form of a reward signal, on decisions made by a target agent during its learning process. The expert signal extends the

environment reward and is used by the target agent to update the policy [112].

- *Q-values* - when agents are based on a TD learning model, expert agent provides Q-values to a target agent. These values are then integrated into the action decision process of the target agent to influence the final decision [46, 47, 49, 50, 116].

- *Experience* - a target agent receives agent-environment interactions obtained by a source agent. Such experience can be used to pre-train the target learning model before exploring, or to extend the standard RL exploration phase [38, 53, 117–122].

Moreover, a learning model can still be entirely re-used and potentially fine-tuned as suggested for T2T TL.

The majority of A2A TL research work is based on the teacher-student framework initially proposed by Torrey and Taylor [45]. In this framework, a teacher guides a student agent during the learning phase, and their interaction is often constrained by a budget.

2.2.3   The Teacher-Student Framework

The teacher-student framework presented by Torrey and Taylor [45] considered two homogenous agents: a teacher and a student. The teacher is regarded as an expert who has already learnt a successful policy for a certain task. On the other hand, student has no prior knowledge about the problem and requires an exploration phase. During the learning phase of the student, the teacher is available on-demand to provide action as advice, assisting the student in achieving an optimal policy. Despite the initial use case of the framework, the teacher-student paradigm has been adapted for *imitation learning*, where student learns to imitate expert policy, e.g., Hester et al. [38] and Taylor [112], or to provide on-demand support to novices during exploration without providing action as advice [43, 50, 51, 117].

The goal of teacher-student framework is to accelerate the learning of a novice agent by reducing the exploration complexity. As a result, teacher-student framework is generally coupled with action-advice, where teacher supports on-demand a novice agent by suggesting an action to take in certain states, [41–45, 48–52]. However, instead of overriding the target policy, the action-decision process can be influenced by providing advice in form of Q-values [46, 47, 50]. Finally, advice can also be provided in the form of an experience batch [116].

Within the teacher-student framework, as the communication between the two agents is generally limited by a budget, one of the initial challenges is to devise a strategy that maximises the positive impact of advice while respecting the budget. An effective strategy should focus on identifying and prioritising situations in which the student is in pivotal need of advice. Some of the initial strategies designed to regulate the teacher-student interaction include:

- *Importance Advising* - aims to identify states that are more important towards the achievement of a goal by analysing the impact of specific actions. For a state, its importance is measured as difference between the maximum and minimum impact brought by available actions. Such advising strategy has been proposed by Torrey and Taylor [45] and adopted in a few approaches [48, 112];

- *Reward Shaping* - teacher continuously provides a reward signal on the actions chosen by the student following its policy. This additional signal extends the reward returned by the environment and aims to promote policy imitation [112];

- *Mistake Correction* - the teacher constantly monitors the student and provides advice when the action chosen by the student differs from the action expected by the teacher policy. Such technique requires the teacher to have an optimal policy [45, 112];

- *ϵ-decay Probability* - probability of giving advice is reduced as the agent explores by a small ϵ value [43, 44, 51];

- *Early Advice* - student utilises the full allocated budget by receiving guidance during the first n steps of learning where it is expected to be in the greatest need [45, 112];

To improve the budget utility and aiming to effectively identify situations where advice can significantly impact the development of the target policy, Taylor et al. [41] proposes *Predictive Advising* aiming to lower the high communication cost of *Mistake Correction* by enabling the source agent to predict the action sampled by the target agent during the learning process. Da Silva et al. [48] enables transfer from an agent that has successfully completed an episode. As a result, all target agents benefit from the full trajectory followed by the source agent, which led to the accomplishment of the episode. In addition, other sophisticated advising strategies may rely on state-visit counter [46, 48, 49], confidence of student and teacher [42, 47, 52, 114–116] and agent performance [49, 53]. Finally, when the

source agent is an expert with optimal policy, the budget utility can be further increased
by re-using previous advice across similar states as proposed by Zhu et al. [43].

Table 2.1 summarises the related work based on the Teacher-Student paradigm categor-
ised by the following features:

- *Advice Source* − regardless of the form of advice, advice can be provided by a human
  operator (*human*) or by an intelligent agent. An agent that has already encountered
  the task can be considered as an expert agent (EA). On the other hand, if the agent
  has not yet converged for the given task it is considered as an imperfect agent (IA),
  e.g., agents that learn simultaneously the same task. Lastly, advice can be given by
  a supervised model (*classifier*).

- *Advice Type* − indicates the form of the advice shared. This can vary between one
  or multiple actions (a), experience or demonstration (e), scalar reward used by target
  agent to complement the reward returned by the environment (*reward*) and finally
  network output logit, Q-values in case of TD-learning method, (Q).

- *Advising Policy* − indicates the advising strategy followed by the agents, which can be
  one of the following: importance advising (ia), reward shaping (rs), mistake correc-
  tion (mk), ϵ-decay probability (ϵp), early advice (ea), predictive advising (pa), sharing
  successful episodes (se), target confidence (t-$conf$) (evaluated on target agent), source
  and target confidence (st-$conf$) (evaluated on both target and source agent), based
  on state-visit counters (svc), based on TD-error (td-$error$) and performance-based
  metrics defined over state (pms).
  Finally, for some advising policy may be not applicable (na), i.e., when transfer hap-
  pens on initialisation of new agents and knowledge is used to pre-train a model [50].

- *DRL* − reports whether the framework is applied on a DRL algorithm or only tabular.

- *Re-using Advice* − indicates whether the target has some memory to retain the advice
  received to increase the budget utility.

- *MARL* − reports whether the framework is applied in a multi-agent context.

The majority of the teacher-student algorithms, reported in Table 2.1, provides advice
in form of action. Furthermore, action is the form of transfer preferred when the source is an
experienced agent [41–43, 45]. When the source agent is not optimal, advice is provided in

Table 2.1: Summary of work based on the Teacher-Student paradigm.

Reference	Advice Source	Advice Type	Advising Policy	DRL	Advice Reuse	MARL
Taylor et al., 2014 [41]	EA	a	$ea, aa, ia,$ mk, pa	✓	✗	✗
Da Silva et al., 2020 [42]	EA	a	$t\text{-}conf$	✓	✗	✗
Zhu et al., 2020 [43]	EA	a	ϵp	✓	✓	✗
Norouzi et al., 2021 [44]	$classifier$	a	ϵp	✗	✗	✓
Torrey and Taylor, 2013 [45]	EA	a	ea, ia, mk	✗	✗	✗
Zhu et al., 2021 [46]	IA	Q	svc	✓	✗	✓
Liang and Li, 2020 [47]	IA	Q	$st\text{-}conf$	✓	✗	✓
Da Silva et al., 2017 [48]	IA	a, e	ia, svc, se	✓	✗	✓
Taylor et al., 2019 [49]	IA	Q	pms, svc	✗	✗	✓
Liu et al., 2022 [50]	IA	a, Q	na	✓	✗	✓
Subramanian et al., 2022 [51]	IA	a	ϵp	✓	✗	✓
Ilhan et al., 2019 [52]	IA	a	$st\text{-}conf$	✓	✗	✓
Taylor and Borealis, 2018 [112]	$human$	$reward$	rs	✗	✗	✗
Chen et al., 2020 [116]	$human$	e	$t\text{-}conf$	✓	✗	✗

form of Q-values that are combined on target side to influence the action-decision process [46, 47, 49].

The teacher-student paradigm achieves state-of-the-art performance when transferring from expert agents to novices, and the effectiveness of the transfer depends on the quality of expert policy. Advisor must not provide advice that is out of date or leads to worsening of target agent learning experience. Therefore, teacher must have an optimal policy that cannot be outperformed by other agents in any way.

Recent developments on the teacher-student framework aimed to relax the optimality constraint by establishing metrics and methodologies to identify when an exploring agent is in need of advice.

Liu et al. [50] adapt the framework to work online with non-optimal teachers and an increasing number of agents. Existing agents distil their knowledge into novel agents that join later on, resulting in less experienced agents benefiting from others. Similarly, Norouzi et al. [44] use a subset of collected demonstration to train a centralised super-entity. The central-entity benefits from experience gathered by all the agents and can be queried to provide advice in specific states. Despite quality of advice being improved over time, this learning paradigm results in an additional cost to gather the RL experience and to train the underlying model.

Da Silva et al. [42], and Chen et al. [116] rely on agent's epistemic uncertainty to decide whether an agent has adequately explored a certain state. The use of confidence has a two-fold advantage, firstly it enables the agent to prevent redundant advice leading to an overall better use of the allocated budget, and secondly it enables a student agent to outperform the teacher by allowing target agent to potentially learn actions better than source of advice.

Despite the better budget utilisation by estimating the confidence on student side, these methods do not assess the advisor proficiency. Ilhan et al. [52] and Liang and Li [47] provide both target and sender with a confidence estimator model that allows for confidence comparison between the two agents. As a result, the target agent can still benefit from an imperfect agent as advisor while lowering the possibility of being negatively impacted by following the advice in states where the target has higher uncertainty than the teacher.

Finally, establishing standardised criteria, including confidence, state visit counters, and performance-based metrics, facilitates the comparison of learning progress among multiple agents in diverse states. Comparing the learning progress of each agent enables bi-direction advice transfer across learning agents, each agent can benefit from knowledge gathered by the other agents [46–49, 52].

2.2.4   Experience Sharing in Agent-To-Agent Transfer Learning

Feeding samples to neural network as a pre-trained stage is an established approach to boost the learning performance of a novel model across different fields, e.g., computer vision [123–125] and natural language process [126, 127].

RL models have been using similar strategies to perform an addition pre-training step before letting the agent explore [128–131].

More generally, there is a recent trend in RL on the transition from online to offline RL [40, 132–134]. While in online RL an agent collects data by interacting with the environment, agent trained in offline RL relies on available data collected by using an unknown policy. Although this thesis does not focus on offline RL, there are some overlapping challenges needed to be addressed with the sharing of experience from one agent to another. For instance, predicting the impact of specific agent-environment interactions on a target

agent, as well as exploring the trade-off between the quantity of tuples provided to a novel agent and its resulting impact on performance.

To follow, this thesis provides a summary of relevant work in A2A TL based on experience sharing in Table 2.2. The category of *Online TL* differentiates whether the transferred experiences are used in an offline context, like a pre-training step, or online to enhance a learning process.

Table 2.2: Summary of experience sharing work in A2A TL.

Reference	Experience Source	Selection Policy	DRL	Advice Reuse	MARL	Online TL
Hester et al., 2018 [38]	*human*	*na*	✗	✗	✗	✗
Gerstgrasser et al., 2022 [53]	*IA*	*td-error*	✓	✗	✓	✓
Wang and Taylor, 2017 [115]	*classifier*	*t-conf*	✓	✗	✗	✗
Wang and Taylor, 2018 [114]	*classifier*	*t-conf*	✓	✗	✗	✓
Nair et al., 2018 [117]	*human*	*na*	✓	✗	✗	✗
Vecerik et al., 2017 [118]	*human*	*na*	✓	✗	✗	✗
Rajeswaran et al., 2017 [119]	*human*	*na*	✓	✗	✗	✗
Cruz et al., 2017 [120]	*human*	*na*	✓	✗	✗	✗
Gabriel et al., 2019 [121]	*human*	*na*	✓	✗	✗	✗
Gao et al., 2018 [122]	*IA*	*na*	✓	✗	✗	✗

Most of the TL work based on experience sharing focuses on transferring agent-environment interactions collected from a human demonstrators to be used as a pre-training step for a RL agent [38, 117–121]. Wang and Taylor and Borealis [114, 115] use pre-collected agent-environment interactions to train a classifier that provides on-demand action-advice to a learning agent. The decision to follow the advice is based on the confidence provided by the source of advice. Wang and Taylor [114] also enable continuous improvement of the classifier by collecting further online demonstrations from uncertain states. Finally, Gerstgrasser et al. [53] enable experience sharing across multiple agents by sharing agent-environment interactions with high TD-error to all the agents.

2.2.5   Research Gaps and Design Challenges

Based on the discussions in the previous two sections, the application scope of the teacher-student framework [46–49, 52] is evolving towards an online scenario where learning agents have a dual roles as both students and teachers. This bi-directional transfer allows for dynamic and effective learning process by enabling agents to fill their knowledge gap. In

fact, when multiple agents address simultaneously a task, it is highly probable that they will quickly develop expertise in narrow and specific areas of the system. Therefore, by sharing their knowledge with one another, it is possible to lower the average policy training time. However, sharing knowledge is a very delicate process and negative transfer is very likely to happen when teacher is less than optimal.

In order to design a successful online teacher-student framework in multi-agent systems requires a number of key challenges to be addressed. First, as all agents are learning, there must be a careful balance between the communication cost associated with the transfer process and the potential influence that the shared knowledge may have on the target agent. The system must be able to identify suitable teachers, from whom the target agent will benefit by gaining new knowledge. Consequently, as the goal of transfer learning is to maximise the positive impact of external knowledge, the system must be able to filter relevant advice that is expected to bring a positive impact to target agent, and update the target policy accordingly.

Therefore, the following challenges must be taken into account:

- *Identify suitable teacher* - the system must be able to dynamically filter any potential candidate from which the target agent will benefit by transferring knowledge.
- *Balance communication cost and transfer influence* - while ultimately the goal of transfer learning is to maximise the positive impact brought by the external knowledge, it is also essential to consider and limit the communication cost involved. In an online scenario, synchronising all of the agents to check whether the target agent will eventually benefit by an advice is unfeasible as both sender and receiver need to synchronise for each action-selection process.
- *Identify relevant knowledge* - the system must select the form of shared advice and decide whether is relevant for the target agent's current level of knowledge. As an example, a more experienced teacher that is transferring knowledge might not bring any support to a student by suggesting an action that would already be selected by the target policy anyway.
- *Policy update based on received knowledge* - the impact of shared knowledge and the method of integrating it into an agent policy may vary depending on the type of

advice exchanged among the agents. For instance, an action could both override the target policy or could increase the likelihood of the action being selected by the agent by influences its weight.

To address these challenges, this thesis proposes Expert-Free Online Transfer Learning (*EF-OnTL*) an online sharing framework based on the teacher-student paradigm where teacher is dynamically selected, based upon some fixed criteria, across available learning agents. The advice is replaced by a batch of agent-environment interactions collected by the teacher. Target agents filter the incoming batch to sample a sub-set of tuples that are expected to improve their policy.

2.2.6   Baselines

Based on the analysis of the TL approaches presented in Section 2.2.3 and Section 2.2.4, it can be concluded that expert-based teacher-student frameworks positively impact the target agent due to the reliable pool of knowledge available to be leveraged.

This thesis shows *EF-OnTL* impact against two action-advice based baselines: 1) Online Confidence-Moderated Advice Sharing (*OCMAS*) [52] - an online confidence moderated action-advice sharing that adapts the teacher-student framework to multiple learning agents. The agents synchronise at each step and the most uncertain agent relies on others to select the appropriate action to take by majority voting. The majority voting along with the uncertainty comparison mechanism ensure that advice is provided in situations where the seeker is really in need of advice and reduce the budget utilisation when compared to classical teacher-student methods. 2) Requesting Confidence-Moderated Policy (*RCMP*) [42] - provides action-advice support to a learning agent by relying on an expert agent. *RCMP* lowers advice frequency as the target agent gradually increase its confidence by exploring the environment. *RCMP* has demonstrated state-of-the-art performance while reducing communication costs, in comparison to a traditional action-advice strategy for the teacher-student framework. The consequence of this, however, is that effectiveness of *RCMP* is strictly related to the quality of teacher used to provide guidance. Advice is moderated by target uncertainty regardless of teacher confidence in a certain state. Therefore, by overriding the actions chosen by the student policy, the teacher may restrict the student performance and

prevent exploration for better solution.

2.2.7   Relationship of Transfer Learning to Collaborative Multi-Agent Reinforcement Learning

In an online context, multiple agents, which interact in the same environment, are enabled to transfer part of their knowledge through TL, which enables these agents to share and leverage a portion of their accumulated knowledge.. Similarly, Multi-Agent Reinforcement Learning (MARL) is typically applied to multi-agent scenarios where multiple agents interact with a single instance of an environment. As a consequence, MARL introduces a collaborative framework, wherein each agent develops its individual policy to address its specific goals while taking into consideration both actions and behaviours of agents in the proximity.

To provide a more precise contextual position of TL and *EF-OnTL*, in this section we discuss similarities and differences between TL and MARL. MARL is a subfield of RL that addresses scenarios where multiple agents interact within a shared environment. The goal of MARL is to enable multiple agents to learn concurrently by learning a policy that takes into account the behaviours of other agents. Generally, in MARL each agent has its own policy and RL approaches that can be used in MARL can be organised into three categories:

- Independent Learners (IL), each agent acts independently by taking decisions based on the state returned by the environment. The policy is updated based on its own experience and all the other agents are treated as part of the environment. As example, Q-learning [82], DQN [101] and PPO [7] can be used to learn independent policies;

- Centralised Training with Decentralised Execution (CTDE), agents' policy are trained jointly in a centralised manner using shared information. However, during execution, each agent makes independent decision based on its own policy without any need for information from other agents. For instance, Multi-Agent Deep Deterministic Policy Gradient [61] (MADDPG) is an extension of DDPG to enable centralised learning in multi-agent context. In MADDPG, each agent selects an action based on its independent actor network, while the critic is centralised. Centralised critic allows for the evaluation of the joint actions of all agents facilitating the learning of cooperative

behaviour across agents.

- networked agents, agents communicate one to the other during training to learn to take joint actions that maximise the global reward while learning an independent policy. As example, QMIX [59] and Value Decomposition Network (VDN) [60] define a global Q-function as a composition of single agents' Q-function. The global Q-function evaluates the impact of the join actions taken by the agents and is used during the training phase to induce collaborative behaviours on the single agent policies. As a result, the Q-function of a single agent is influenced by the actions of other agents.

While the objective of MARL is to encourage agents to learn cooperative behaviours through interaction and information exchange, *EF-OnTL* aims to improve the overall system performance by enabling agents to learn independent policies while sharing batch of agent-environment interactions to transfer a subset of knowledge that is expected to improve the target policy. Furthermore, while MARL allows agents to have different reward models and objectives, *EF-OnTL* is not specially designed to transfer between agents with different goals. To enable the transfer across agents with different reward models, a mapping function is required to map the experiences and rewards from one agent to another.

2.3   Uncertainty in Reinforcement Learning

In RL there are two types of uncertainties, *aleatoric* and *epistemic*. *Aleatoric* uncertainty comes from the environment and is generated by stochasticity in observation, reward and actions, *epistemic* uncertainty comes from the learning model and indicates whether the agent has adequately explored a certain state. In RL, uncertainty has been largely used to identify unfamiliar states and hence to enhance overall agent exploration [70, 135–139]. Most recent TL frameworks follow this insight by relying on epistemic uncertainty to determine whether an agent requires guidance or not [42, 49, 117].

One feasible way to approximate agent epistemic uncertainty within a specific task is by counting the number of visits per each state. In fact, Taylor et al. [49], Da Silva [48] and Nourouzi et al. [44] estimate uncertainty from state-visits counter. Zhu et al. [46] improve the estimation by relying on state-action pair. However, state-space might be continuous or

very large making the naive counting unfeasible. Thus, state-visits could be approximated by Random Network Distillation (RND) [70]. Despite RND originally being introduced to encourage exploration, it has already been exploited as uncertainty estimator for TL-RL [52]. RND consists of two networks, a target with unoptimised and random initialised parameters and a main predictor network. Over time, the target network is distilled within the predictor and uncertainty is defined as prediction error between the two outputs.

Other sophisticated models can be used to estimate the epistemic uncertainty. Wang et al. [115] propose uncertainty estimation through neural network, decision tree and Gaussian process. NN is also used by Da Silva et al. [42], where the agent learning model is expanded by replacing control-layer with an ensemble later used to approximate agent uncertainty.

Despite the different underlying technique used to estimate uncertainty, all the afore-mentioned methods rely uniquely on visited state and thus, might lack crucial information needed to estimate uncertainty in states where the goal is close but not yet achieved. For instance, in sparse reward environment and tasks with continuous control space, action must be taken into account when estimating the intrinsic agent uncertainty when they are far from optimality. While taking into account actions explored along states might not be necessary when bootstrapping expert knowledge, it is important to discriminate between sampled actions over states when transferring from learning agents.

2.4   Summary of Background

A Markov Decision Process (MDP) is a mathematical framework to model tasks that require sequential decision making. MDPs are based on the Markovian property, which states that the probability of transitioning from one state to the next depends solely on its current state and it is independent of its past states.

Reinforcement Learning (RL) is a machine learning framework that enables an intelligent agent to learn the transition function of a MDP through continuous interaction with the environment. RL is a cyclic four-step process composed by: 1) Observation ; 2) Action Selection; 3) Actuation; and 4) Evaluation. This cycle enables a RL agent to learn a policy by updating its knowledge based on the new sampled evidence.

One of the major limitations of RL lies in the representation used to learn the mapping between states and actions. The state-action combinatorial complexity limits the applicability to domains with low-dimension state-action space.

Deep Reinforcement Learning (DRL) partially overcomes RL limitations by using a Neural Network (NN) to learn the high-level features that are used for the action decision process. Furthermore, the use of NNs enables an agent to generalise across similar visited states reducing the complexity that arises from the exploration of state-action space to collect new evidence. Despite the exploration cost being mitigated by the capacity of NNs to generalise across similar states, this along other RL limitations remain unsolved.

Transfer Learning (TL) can partially overcome the DRL limitations as follow:

- TL improves the sample efficiency by including external expertise within a DRL agent;
- TL allows for policy re-use across similar tasks;
- TL lowers the learning complexityy by partially or fully adapting previous acquired policies;

TL enables part of knowledge to be transferred from one agent, named source, to another to improve the learning phase of a target agent. Depending on the objective of TL, the form of knowledge transferred can vary, i.e., action, agent-environment interaction, reward and policy. Other factors that influences the designing of a TL framework are the availability of an expert or a trained policy and the access to pre-collected experiences.

To provide support to a learning agent, the predominating framework used is the Teacher-Student paradigm, where an experienced teacher provides advice to a novel agent. In the teacher-student framework, the advice generally influences the action-decision process, i.e., action or Q-values, or the weight of learnt state-action pair, i.e., reward.

The teacher-student framework proposed by Da Silva et al. [42], Requesting Confidence-Moderated Policy ($RCMP$), has demonstrated state-of-the-art performance. In this approach, a target agent explores the environment and estimates the epistemic uncertainty for the states it visits. If this uncertainty falls below a predefined threshold, the target agent seeks advice from an experience teacher, which provides an action to be followed by the target.

The teacher-student framework can be extended to multiple learning agents where the roles of teacher and student are swappable over time. For instance, Ilhan et al. [52] propose Online Confidence-Moderated Advice Sharing (*OCMAS*). In this approach, an uncertain agent asks other agents for an action-advice. Finally, the advice seeker follows the final action based on majority voting of the suggested actions.

Wang and Taylor [114] propose a TL framework where the teacher, an external entity trained on previously stored agent-environment interactions, provide action as advice to a learning agent. Furthermore, in uncertain states, the teacher can request for additional demonstrations to improve the quality of the advice given. Gerstgrasser et al. [53] propose a TL framework where the tuples that exhibit high TD-error within a specific agent are forwarded to all the learning agents.

This thesis focuses on the transferability of knowledge across homogeneous learning agents by introducing *EF-OnTL*. The objective of this thesis is to investigate the impact of experience sharing against the teacher-student framework. *EF-OnTL* is compared against two action-advice baselines: *OCMAS* [52] to compare the impact of action-advice against experience sharing by *EF-OnTL*; and *RCMP* [42] to benchmark *EF-OnTL* capability against an action-advice baseline from experienced teachers. In the *RCMP* version implemented in this thesis, the single optimal teacher is replaced by multiple trained agents to prevent any bias coming from the use of a single non-optimal teacher. Consequently, the action followed by a target agent is picked by majority voting on the actions provided by the teachers.

EF-OnTL enables agents to share collected experience autonomously to promote continuous improvement of a multi-agent system. Similarly, Multi-Agent Reinforcement Learning (MARL) algorithms are designed to enhance the learning of multi-agent systems by enabling the agents to learn cooperative behaviours. MARL enables collaboration among learning agents through Centralised Training with Decentralised Execution (CTDE) and networked agents. In CTDE, the agents collaborate during their training process to enable the learning of complex and collaborative policies. For instance, in the multi-agent variant of Deep Deterministic Policy Gradient (DDPG), known as MADDPG, the critic is centralised and shared across all the agents. The centralised critic takes in input the state-action of each agent to estimate the Q-value of the joint actions.

In networked agents, communication among between agents plays a crucial role in co-ordinating their actions and making collaborative decisions. As example of networked agents, QMIX [59] and Value Decomposition Network (VDN) [60] combine the Q-values of independent agents to learn a global Q-function that considers the states and actions of each agent.

3   Preliminary Studies

This chapter serves as a feasibility study towards the development of Expert-Free Online Transfer Learning (*EF-OnTL*). While the objective of *EF-OnTL* is to facilitate online Transfer Learning (TL) among multiple agents learning simultaneously, the content of this chapter focuses on TL in the offline context. In comparison to *EF-OnTL*, the challenges addressed by experience sharing in offline TL are relatively reduced, as, in this context, the source of transfer is a nearly optimal agent. This situation implies that the target agent should benefit from the knowledge provided by the teacher due to the difference in the exploration time.

Through this offline evaluation, our aim is to investigate and observe the implications of offline TL in environments where we plan to assess the online approach. The evaluation of this preliminary stage is carried out in two distinct environments: 1) a Single-Team Predator-Prey (ST-PP) scenario, in which experiences, selected based on their associated uncertainty, are transferred to a new agent; and 2) Ride-Sharing Ride-Requests Simulator (3R2S), a setting where experiences are transferred across agents operating with diverse demand patterns

The remainder of this chapter is structured as follows: the architecture used for the experiments is detailed in Section 3.1; the environments used to benchmark the transfer of experience are presented in Section 3.2. To follow, the impact of experience sharing in offline TL is analysed in Section 3.3. Lastly, Section 3.4 evaluates Random Network Distillation (RND) as an uncertainty estimator model, highlighting its limitations when used in an online context like *EF-OnTL*, and introduces State Action Reward Next-state Random Network Distillation (*sars-RND*) to overcome the identified shortcomings.

3.1 Proximal Policy Optimisation Architecture

This section introduces the architectural design for the Deep Reinforcement Learning (DRL) algorithm implementation used to learn the agent's policy. For the preliminary experiments on offline experience sharing, we extended the classical Proximal Policy Optimisation (PPO) algorithm [7] to enable learning from external experience.

PPO clipped version, introduced earlier in Chapter 2, has been used to carry out the preliminary experiments to assess the impact of experience sharing in Multi-Team Predator-Prey (MT-PP) and 3R2S.

Unlike off-policy algorithms that use two different policies to sample and optimise, PPO assumes that the samples used to optimise a policy are collected by the current version of the very same policy. To enable the use of external experience in PPO, the model needs to be expanded as depicted in Figure 3.1.

The learning stage of the expanded PPO model is divided in two parts. Firstly, the learning model is optimised based on part of the transferred experiences. The agent samples B tuples from the transfer buffer, ensuring that each tuple's uncertainty is below a predefined threshold. The selected tuples are then used to populate an experience buffer, which is finally used to perform a few steps of gradient descent optimisation. Secondly, the agent interacts with the environment without relying any longer on the external transferred knowledge and updates its policy based on the sampled experience.

While the expanded version of PPO used for the preliminary experiments requires the use of a transfer buffer, there are no changes within the internal model architecture shown in Figure 2.4. For the preliminary experiments conducted in two environments, ST-PP and 3R2S, the PPO models' architecture for both actor and critic networks, as well as the learning parameters, are reported in Table 3.1. The learning parameters specify the number of policy updates performed over each mini-batch and the clipping ratio used for the clipped PPO.

However, for TL applications in an online setting like *EF-OnTL*, we transitioned from using PPO to alternative models. This transition is necessary due to the nature of PPO being an on-policy method, which makes it unsuitable for mixing external information with
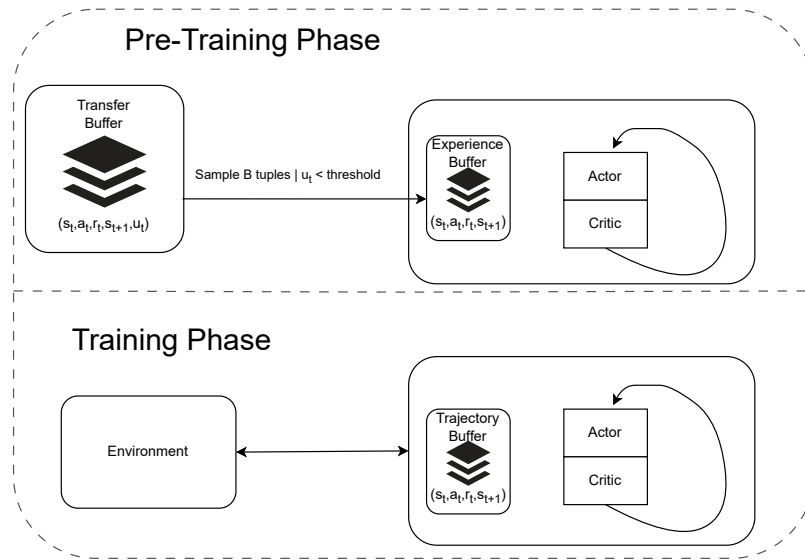
Figure 3.1: PPO extended model.

Table 3.1: PPO parameters setup for preliminary experiment.

Parameter	ST-PP	3R2S
actor		
Input Layer	Linear(18, 64)	Linear(26, 128)
Hidden Layer(s)	Linear(64, 64)	Linear(128, 128) Linear(128, 128) Linear(128, 128) Linear(128, 64)
Output Layer	Linear(64, 5)	Linear(64, 5)
Activation	Tanh	Tanh
critic		
Layer	Linear(18, 64)	Linear(26, 128)
Output Layer	Linear(64, 1)	Linear(128, 1)
Activation	ReLU	ReLU
Optimiser	Adam	Adam
Learning Rate	1e-3	1e-4
Betas	(.9, .999)	(.9, .999)
#epochs	10	10
Clipping ratio	.2	.2
Mini-batch Size	32	32

the data sampled by the current policy. On-policy methods presume that the experience used to update the policy is sampled using the current policy version. However, in offline TL, the optimisation steps conducted on external knowledge are carried out prior to updating the model based on the sampled experience.

3.2   Benchmark Environments

To evaluate the impact of experience sharing in an offline TL scenario, we used two bench-
mark environments: ST-PP, introduced in Section 3.2.1 and 3R2S, introduced in Sec-
tion 3.2.2.

3.2.1   Predator-Prey

Predator-Prey is a task originating from biology where certain species hunt other indi-
viduals for survival [140]. Different versions are available and this thesis uses a grid-world
implementation based on an existing mini-grid environment compatible with OpenAI gym
APIs [141].
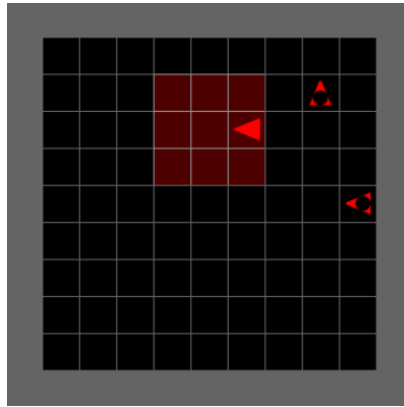
Figure 3.2: 9x9 predator-prey grid with one predator and two prey.

The configuration used for this study consists of a $9x9$ obstacle-free grid with a single
predator and two prey, as shown in Figure 3.2, where the predator is illustrated as a filled
triangle and the prey as a pierced triangles. The orientation of each entity matches the
direction of the triangle.

The objective of a predator is to catch all the prey available within the grid. To accom-
plish such a goal, a predator has 5 possible actions to take:

1. *hold* − the predator stays within the same cell and does not change its rotation;

2. *forward* − if the facing cell is empty, the predator moves from the current cell to the
   next;

3. *rotate left* − the predator orientation is changed by −90°;

4. *rotate right* − the predator orientation is changed by $+90°$;

5. *catch* − the agent catches the object in the facing cell. A catch succeeds if and only if the object within the consecutive cell is a prey.

The control level for a prey is analogous to that of the predator, with the exception of the *catch* action. Prey escapes predator by sampling random actions with the following probability distribution: *hold* with a probability of 10%, rotate $±90°$ with a probability of 25%, and *forward* with a probability of 40%.

3.2.2 Ride-Sharing Ride-Requests Simulator

3R2S consists of a mobility on-demand scenario that enables ride-sharing while serving real-world ride-requests. Ride-requests are replicated by a real-world taxi request dataset obtained from the Manhattan area [9].

While the demand pattern is replicated from the Manhattan yellow cab dataset, traffic patterns and road network infrastructures are simulated through the open-source Simulation of Urban MObility [74] (SUMO) simulator. SUMO is largely adopted because of its realistic and fully customisable simulation for urban mobility scenarios and provides high granularity in modelling urban transportation elements.

A screenshot taken from the SUMO platform is shown in Figure 3.3. The upper image displays the Manhattan road infrastructure, while the lower one shows an intersection with a single taxi vehicle and its perceived requests that could potentially be picked-up. The red or green coloured bars intersecting the road lanes in proximity of an intersection represent the status of the traffic lights at the intersection. The vehicles are divided into two groups: traffic vehicles, represented in blue, which are travelling around to simulate traffic patterns; and taxi vehicles, which are coloured in yellow. Finally, the passenger requests are depicted as a stylised top-view human shape and coloured in red.

The problem of matching ride-requests and driver while enabling ride-sharing can be described as a fleet of vehicles travelling around a certain area to maximise the number of satisfied requests while optimising the constrained resources by enabling car sharing. In this scenario, there is a one-to-one mapping in which each agent is uniquely associated with a
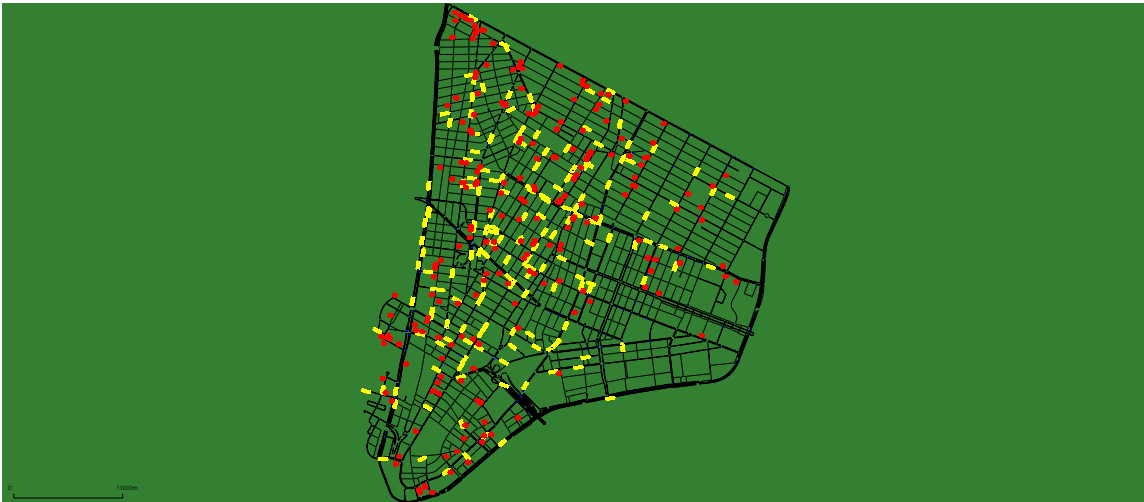
(a) Ride-sharing enabled taxi cabs are depicted as yellow boxes and ride-requests as red circles.
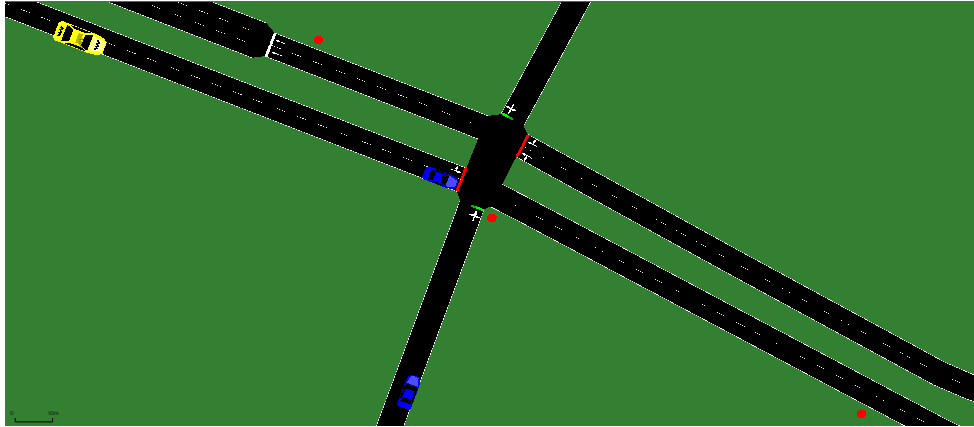
(b) Close view of an intersection with its traffic lights. Taxi cabs are depicted in yellow, regular vehicles in blue, and potential passengers are shown in red.

Figure 3.3: SUMO simulator screenshots.

single taxi vehicle, and each taxi vehicle is exclusively controlled by an agent.

From a single agent perspective, the goal is to maximise the amount of ride-requests served throughout its life time while minimising the cumulated delay for the vehicle and any on-board passenger. Control is distributed at vehicle level and there is no communication nor coordination between agents.

In the request dispatcher system, each request is broadcast to the fleet as it occurs and remains active for 15 minutes. If the request is not engaged within this time frame, it expires and is registered as not served.

A request is considered engaged when a vehicle begins to drive towards the pickup point to pick up the passengers. As a constraint, the estimated time for the vehicle to reach the

request point must fall within the request window time. However, the actual driving time could be longer due to varying traffic conditions compared to the initial estimation. Once the passengers are boarded, they are either travelling towards their destination or detouring to pick-up additional requests to be served in ride sharing by the vehicle.

The state of an agent is composed of internal vehicle status alongside the perception of the agent on the closest requests that could potentially be served. At a specific time, given a certain vehicle and a specific request, the request is eligible to be served by the vehicle if there is enough room in the cab to accommodate the incoming passengers and the estimated travel time to reach the pick point of the request is compliant with the request expiration time. Internal observation is composed of current position, updated in real-time, and destination with number of empty seats that can be used to welcome new passengers. The destination of a vehicle is updated at the end of an event, i.e., vehicle arrives to pick-up or drop-off location. On top of that, a vehicle may decide to take a detour to pick additional passengers while travelling towards a certain drop-off location. This evaluation step is allowed whenever the vehicle is further from the current destination point and has at least a free seat. A representation of the state of a vehicle and its perception is reported in Figure 3.4a.

In the experiments carried out for this research, agent perceives up to 3 closest requests, as depicted in Figure 3.4b. Consequently, control space consists of 5 actions: 1) *parked* - vehicle stays idle for a predefined amount of time; 2) *drop-off* - vehicle drives towards its destination until it reaches the drop-off point, and offload the passenger(s) there; 3-5) *pick-up* - agent updates its destination to match the origin point of the selected request and drives towards it and passengers are on-boarded when it arrives. Note that agent can decide to pick one of the three available requests and as a result, agent has three different pick-up options, resulting in 3 available actions for pick-up. All vehicles within the fleet are synchronised under a global clock and action decision process follows a FIFO queue. Considering the nature of the task, an episode begins as vehicles enter the simulation and terminates when all taxis have left after successfully driving all the passengers on-board, and with no pending requests left to be served.

As introduced earlier, the reward scheme is designed to incentivise vehicles to accomplish

internal feature
position (x,y) [*float, float*]
destination (x,y) [*float, float*]
free seats [*int*]

Closest requests

observed feature
#passengers [*int*]
origin (x,y) [*float, float*]
destination (x,y) [*float, float*]
detour_time to_pick [*float*]
detour_time to_drop [*float*]

observed feature
#passengers [*int*]
origin (x,y) [*float, float*]
destination (x,y) [*float, float*]
detour_time to_pick [*float*]
detour_time to_drop [*float*]

(a) Perception schema Ride-Sharing Ride-Requests Simulator (3R2S).

(b) Internal vehicle state and agent's observation over the 3 closest requests that could be engaged by the vehicle.

Figure 3.4: Vehicle internal state and perception.

as many requests as possible within the constraints of limited resources, while minimising the accumulated delays of vehicles and passengers. Therefore, reward is generally based on the elapsed SUMO steps from the beginning of an action to the decision time for the subsequent action. First, when it is not possible to fulfil an action, agent receives a penalty of -1. Second, when agent decides to remain parked, it receives a negative reward R_p defined over the elapsed parked time as defined in Eq. 3.1, where x is set to 1 with empty vehicle and 5 otherwise.

$$R_p = \frac{-x}{x + elapsed\_time} \tag{3.1}$$

Third, on drop-off reward R_d is defined by Eq. 3.2 if the request has been served in ride-sharing. $noRStime$ is an estimation of the time that the same vehicle would have taken to fulfil the ride-request with similar traffic condition without enabling ride-sharing. $noRStime$ is the estimated travel time needed to travel from the current position to a target location based on the current state of the network. When the request is not served in ride-sharing, R_d is set to 5.

$$R_d = \max\left(-1, 5 - \frac{elapsed\_time - noRStime}{noRStime}\right) \tag{3.2}$$

Last, on pick-up, reward R_{up} is defined as reported by Eq. 3.3 where x is set to 1 with empty vehicle and to 2 otherwise. $elapsed\_time$ captures the time spent to drive towards the pick location from the moment of the assignment of the request.

$$R_{up} = \frac{x}{x + elapsed\_time} \tag{3.3}$$

To emulate a real-world scenario, this thesis utilises ride-request data generously provided by Guériau and Dusparic at [86]. NYC taxi trips [9] from 50 consecutive Mondays between July 2015 and June 2016 in Manhattan are aggregated and divided based on time-base schedule resulting into 4 datasets: morning, afternoon, evening and night. In this thesis, all experiments are based on the morning peak slot (from 7 to 10 am) and the evening (from 6 to 9 pm). A depiction of the different demand trend and ride-requests imbalance is reported
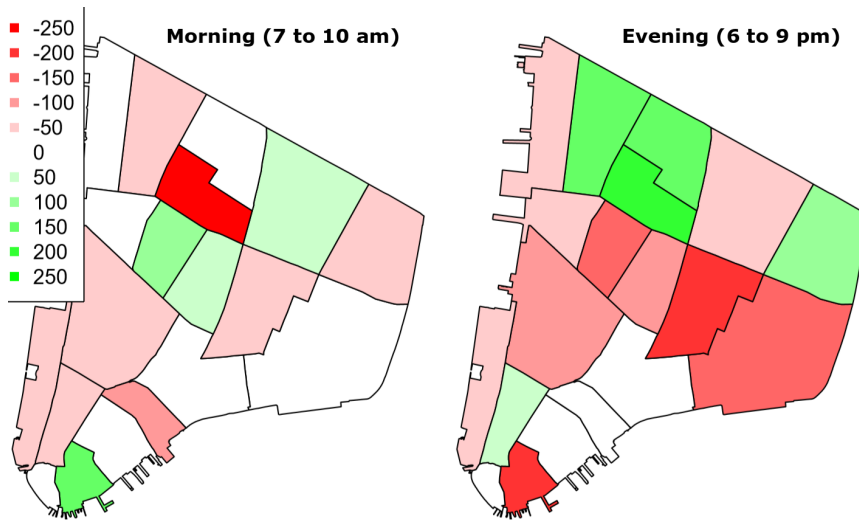
Figure 3.5: Observed demand imbalance in New York Taxi dataset [9] trips between morning (7-10am) and evening (6-9pm) peak hours in the south part of Manhattan on Tuesday, February 2nd 2016.

in Figure 3.5. In the Reinforcement Learning (RL) context agents need to generalise across different demand patterns and thus having different demand set with imbalanced zones are crucial to evaluate the generalisation aspect.

To facilitate effective policy learning for the agents, the training is structured into 10 epochs, each containing 10 episodes of single-vehicle training. In a given episode, an agent can serve the unmet requests that previous vehicles could not fulfil within that specific epoch. At the beginning of a new epoch, the demand set is fully restored. This strategy emulates a scenario of concurrent exploration by multiple vehicles without any competition among them in serving customers.

All the experiences gathered during training are processed by a single learning process. Consequently, all the agents contribute to the same learning process, optimising the utilisation of accumulated knowledge and accelerating the learning process. In cases where a vehicle fails to perform an update or is unavailable to serve requests in a particular location, the others can continue seamlessly. Once training is completed, the acquired knowledge is propagated to all vehicles within the fleet. Furthermore, this allows new vehicles to join the fleet without requiring additional training efforts.

3.3   Impact of Experience Sharing in Offline Reinforcement Learning

The objective of this offline TL study is to analyse the impact of experience sharing and comparing it against the teacher-student framework presented in Chapter 2. This study serves as an introductory step towards the development of *EF-OnTL* in an online sharing context. In this offline experience sharing scenario, the set of challenges to be addressed is reduced as there is no source selection process as the source of transfer is a trained agent. Specifically, the aim of this study is threefold:

1. identify and study how pivotal parameters influence the outcomes of transfer, i.e., source uncertainty and quantity of transferred tuples;

2. assess and quantify the extent of positive transfer achieved through the sharing of external experiences;

3. compare experience sharing influence against action-advice based teacher-student baselines that provide tailored action to take in critical situations.

The following experiments, demonstrating the impact of experience sharing in an offline-TL context, are based on the PPO model presented in Section 3.1. The method and the baselines are applied to the ST-PP environment and 3R2S.

The experience transfer buffer is provided by an external agent that has already encountered and solved the task. The sharing of experience and the integration process within the target learning model happens at initialisation time. The target agent receives a buffer with tuples labelled by source agent's uncertainty and samples a subset of the available collected knowledge. To estimate confidence, source agent is equipped with a RND model that enables the agent to approximate its epistemic uncertainty while learning the task. The source agent stores all the tuples visited during training along with the associated epistemic uncertainty estimated by the uncertainty model.

Compared to online TL, this offline experience sharing method has a reduced complexity as it assumes the existence of a pre-identified teacher agent, which is certainly more qualified than the novel target agent. To benchmark the impact of sharing experience as pre-

training phase, the offline transfer approach is compared against *no-transfer*, where a target agent does not utilise external knowledge, and action-advice based teacher-student framework following various advising strategies, i.e., advice at the beginning (*BS: adv_ at_ begin*), confidence based ϵ-decay [44] (*BS: confidence_ based_ e-decay*) and mistake correction (*BS: mistake_ correction*). In *BS: confidence_ based_ e-decay*, a target agent asks for advice following an ϵ-decay probability and when its more uncertain than the teacher in the visited state.
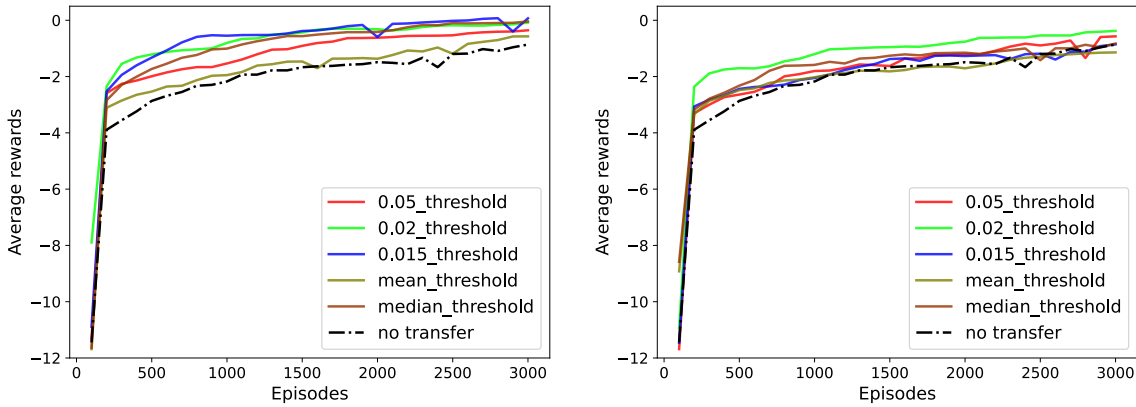
(a) Average rewards per different threshold with pre-training over $5,000$ agent-environment interactions.

(b) Average rewards per different threshold with pre-training over $10,000$ agent-environment interactions.

Figure 3.6: Comparison of offline experience sharing in ST-PP across different filtering thresholds. Results are shown as average learning curve across 50 independent simulations.

Figure 3.6 shows two different graphs with the average rewards over $3,000$ episodes of the pre-training enabled algorithm across different filtering thresholds and a *no transfer* baseline. Figure 3.6a enable an agent to sample $5,000$ interactions during the pre-training step, while Figure 3.6b the pre-training budget is doubled to $10,000$. These experiments use three fixed thresholds: 0.05, 0.02 and 0.015, and two adaptive thresholds based on the content of the buffer available: *mean* and *median*. The *mean* threshold enables a target agent to sample tuples with an uncertainty level lower than the average uncertainty computed across all interactions in the buffer. On the other hand, the *median* threshold enables a target agent to sample tuples with an uncertainty level below the median uncertainty of all interactions.

These experiments show that pre-training a learning model with external experience coming from a trained agent successfully improves the performance of a target agent. The positive impact appears to be unmatchable by a *no transfer* agent. In fact, while jumpstart

may not be consistent across the TL enabled agents, there seems to be an asymptotic improvement brought by the pre-training phase that enables the target agent to keep a higher level of performance from the beginning to the last episode. Finally, although there are minor differences based on the filtering threshold selected, the predominant feature, from these graphs, is the quantity of shared experiences.

Intuitively, the more information is provided to a novel agent, the better it is expected to adapt and learn the task. However, the results provided by Figure 3.6 empirically contradict this intuitive assumption by showing an opposite trend. In fact, in Figure 3.6a, all the thresholds show an improvement against *no transfer*, while in Figure 3.6b, the impact given by the selected threshold is milder, and a few scenarios show no improvement at all or even negative transfer when compared against *no transfer*.

To follow, the threshold that enabled the TL target agent to achieve higher overall performance and the *mean* threshold are compared against the TL baselines to compare the impact of transferring agent-environment interactions against action-advice based baselines.
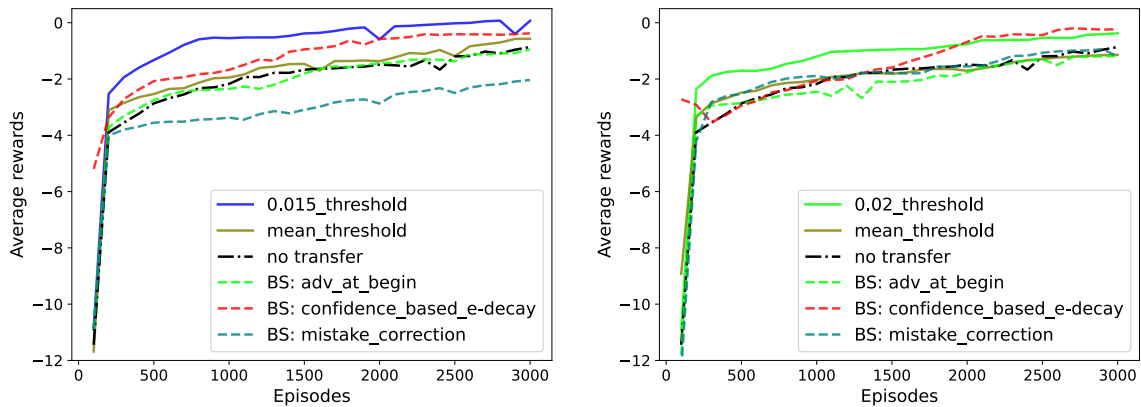
(a) Average rewards per different algorithm type with pre-training over $5,000$ agent-environment interactions.

(b) Average rewards per different algorithm type with pre-training over $10,000$ agent-environment interactions.

Figure 3.7: Comparison of offline experience sharing in ST-PP against the teacher-student framework.

Figure 3.7 shows two different graphs for the total reward over $3,000$ training episodes. The leftmost graph shows the TL algorithms with a budget up to $5,000$ while the rightmost graph depicts the experiments with a budget up to $10,000$.

These comparisons suggest that experience-based TL can compete against action-advice

based baseline. Specifically, three main findings arise, as follows. Firstly, providing action as advice by taking into account the confidence measure of both source and target allows for a better budget utilisation and it leads to better performance when compared to *BS: mistake_correction* and *BS: adv_at_begin*. Secondly, despite the remarkable jumpstart exhibited by the *BS: confidence_based_e-decay* compared to *no transfer* within the initial 50 episodes, the *0.015_threshold* agent achieves a similar level of rewards in a shorter number of episodes. Furthermore, the interactions processed as a pre-training step are general and not tailored to overcome a specific situation. On the other hand, the advice provided by action-advice based baseline are specific to overcome a certain situation. Thirdly, while the filtering threshold plays a role into influencing the target agent performance, the predominant feature driving the transfer outcome is the amount of transferred interactions.

As second part of the initial offline evaluation phase, transfer of experience is applied to the 3R2S environment while varying the underlying demand pattern. Therefore, the following scenarios are evaluated:

- *Train & test 7-10am* − agent is trained and tested on the same set of requests, the morning peak hours. This is considered as a baseline as in this scenario there is no transfer;

- *Train & test 6-9pm* − agent is trained and tested on the evening peak hours. This is a transfer-free baseline;

- *Train 7-10am test 6-9pm* − agent is trained on the morning peak hours and tested on the evening demand set. In this baseline a policy trained on a demand set is tested onto another to assess the capability of a policy with no experience sharing to generalise across different demand patterns;

- *TL-enabled test 6-9pm* − agent is trained and tested on the evening demand set. However, this scenario enables TL as the agent accesses a transfer buffer containing the experiences collected during a training of the morning peak hours to perform a pre-training step.

These scenarios evaluate whether and to what extent the transfer of experiences among agents, defined within the same domain but based on different underlying data, can result in positive transfer to a target agent.

Table 3.2: Performance metrics across all 4 evaluated approaches in 3R2S environment.

Scenario	%Served requests	%RS	σ pass	$\overline{d_t}$ (km)	$\overline{D_r}$
Train & test 7-10am	93	93	76.61	140	9.4
Train 7-10am test 6-9pm	77	99	21.65	87	9
Train & test 6-9pm	76	99	19.35	87	9.11
TL-enabled test 6-9pm	79	93	41.12	82	5.23

The averaged results are presented by Table 3.2 while Figure 3.8 shows the distribution across the fleet and requests. In detail, it reports the waiting time distributions for requests in *(a)*, passengers distribution in *(b)*, and mileage distribution in *(c)* across the fleet.

The first improvement resulting of sharing experience in the 3R2S environment is represented by the increase of served requests, as reported in Table 3.2. When an agent samples a subset of pre-collected experiences to pre-train its learning model, (*TL-enabled test 6-9pm*), it enables a fleet to satisfy an additional 3% of ride-requests when compared to a similar agent that does not utilise external knowledge on same demand set, (*Train & test 6-9pm*). Despite the additional served requests, the average distance travelled ($\overline{d_t}$) is lowered by 5 km per vehicle, 82km against 87km, when compared to a no-transfer agent (*Train & test 6-9pm*). Furthermore, passenger metrics improved as well, by lowering the detour ratio ($\overline{D_r}$). $\overline{D_r}$ is defined as the ratio between the estimated driving distance from the pick-up to the drop-off location with no detour and the actual travelled distance by the vehicle while the passengers are onboard. Finally, passengers distribution (σ pass) in the TL enabled scenario, (*TL-enabled test 6-9pm*), is unbalanced compared to no-transfer scenarios tested on the same dataset, (*Train & test 6-9pm*) and (*Train 7-10am test 6-9pm*), as variance is almost doubled, 41.12 against 19.35 and 21.65, respectively.

In the *TL-enabled test 6-9pm*, the average waiting time is increased by less than 1 minute, as a result of serving additional passengers. However, as shown in Figure 3.8a, only a few requests have an increased waiting time above 15 minutes and those are very likely the cause of the increased waiting time.

(a) Distribution of waiting time over the requests.

(b) Distribution of passengers served per vehicle.

(c) Distribution of distance travelled across the fleet.
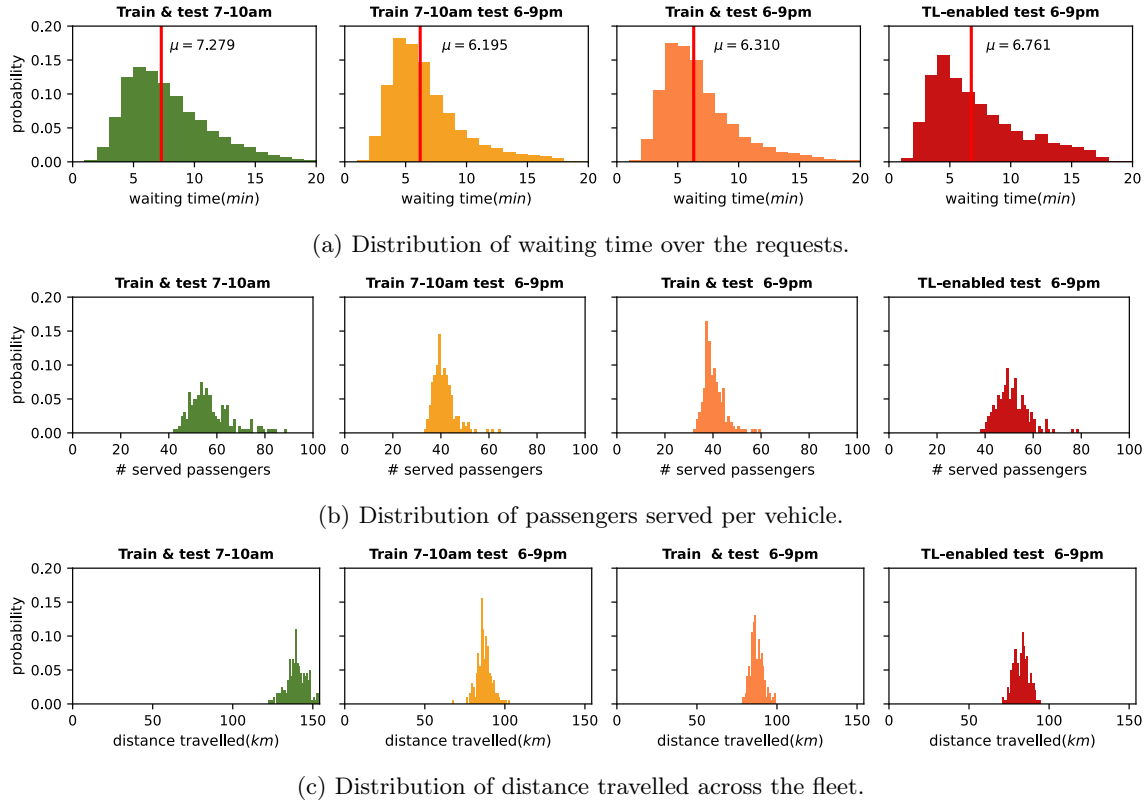
Figure 3.8: Evaluation metrics in the 3R2S environment across 4 different scenarios.

3.4   Random Network Distillation Considerations

The preliminary experiments have shown that scalar estimate of agent epistemic uncertainty influences the performance of a target agent when transferring external experiences.

Therefore, this section discusses Random Network Distillation (RND), introduced in Section 2.3. RND is a two-network model used to approximate agent epistemic uncertainty on visited states. Consequently, this section highlights RND shortcomings when used as uncertainty estimator in an online TL scenario. Finally, this section presents *sars-RND*, our proposed extension of RND, to overcome the intrinsic RND limitations.

Before jumping into RND limitations, this section evaluates RND estimation capability based on the amount of encoded features compared between the two networks. The encoder size matches the number of neurons used within the output layer.

While a bigger encoder size enables the estimator to be more sophisticated, it might lead to overfitting when estimating the output of the other network and, consequently,

when estimating the uncertainty. On the other hand, a smaller encoder size does not allow for a sufficient number of features to be compared between the two networks. To define the size used within this thesis, Figure 3.9 presents a RND sensitivity analysis while varying the encoder size on the MT-PP problem. In detail, the figure shows the maximum uncertainty registered within the first $1,000$ episodes. Encoder size identifies the number of features that are used to compute the Mean Squared Error (MSE) between the frozen random network and the optimised one. A general trend observed from the graph is that uncertainty is sharply decreased within the first episodes and thereafter anneals towards a small value close to 0.

Regardless of the chosen encoder size, the uncertainty model presents a descending trend. A higher number of features makes the estimator more sensitive to newer states while a lower number generalises too much across different states. Therefore, for all the experiments presented in this thesis, the encoder size is set to $1,024$ as a balanced trade-off which provides sufficient granularity to generalise across similar states while allowing the model to react to new or infrequently visited states.
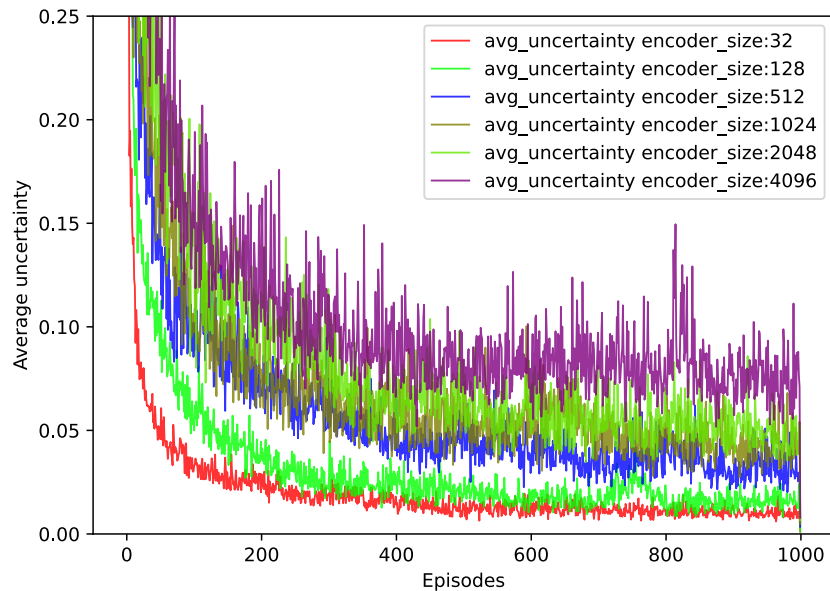
Figure 3.9: Uncertainty estimator with different encoder size for RND model in MT-PP.

In Section 3.3, we demonstrated that RND can be successfully used to identify meaningful agent-environment interactions that result in increased performance to a novel agent.

The tuples processed by the target agent are selected based on a threshold level. Given the observed decreasing trend registered in Figure 3.9, the tuples processed by the target agent are very likely to be collected within the latest training episodes of the source agent. This hypothesis is supported by the experiments with a smaller transfer budget shown in Figure 3.6a, where a lower threshold (0.015) increases the chances that a shared tuple is sampled from the latest part of the training process. Similarly, increasing the threshold (0.02 and 0.05) reduces the probability of sampling from the latest part of training and as a result, it gradually deteriorates the performance. However, when doubling the transfer budget, from $5,000$ to $10,000$, this hypothesis is not true as the lower threshold that was performing the best earlier is now the worst among the three.

The deterioration of performance with an increased budget might be due to the environment that leads predator to visit very often similar states that hold little or no information while taking different actions. Given the sparse reward function and the high probability that an agent is in a state with no information, there might be many tuples transferred that are related to these situations that add no value to the agent's knowledge. For instance, in the MT-PP game screenshot provided by Figure 3.2, the predator is in a not useful state. The predator, filled triangle in cell $(6,3)$ from the top-left corner, will transition to a same perceived situation with any of the available actions (*hold*, *forwards*, *rotate right*, *rotate left* and *catch*). As an agent perceives a $3x3$ grid-based on the next consecutive cell of the controlled predator, there is no way of perceiving anything different from empty cells by taking a single step in this specific setup.

When transferring agent-environment interactions, based on uncertainty, from an experienced agent that has already completed the training phase, the redundant visits to states with little information may not be a problem. In fact, this issue can be mitigated by selecting tuples based on uncertainty, thereby prioritising recent experiences, and by regulating the transfer budget allowing for a limited number of redundant visits. Nevertheless, when transferring in an online TL scenario, adjusting the budget in an ad-hoc manner will not alleviate the issue. The source agent will not have completed its training process yet, and consequently, it will not have sufficiently visited the successful state-action combinations.

3.4.1   RND and *sars-RND* Comparison

RND has proven to be a valid uncertainty estimator model to be used within an offline TL context where tuples are selected from agents that have completed the full training process. However, when applied in an online context, RND lacks additional information that should be taken into account when selecting experiences to share from imperfect agents. This limitation arises because RND estimates confidence based solely on observed states, assigning high confidence to states the agent frequently visits regardless of the action taken. Consequently, it might fail to accurately estimate the confidence for states where the agent has rarely, or even never, explored a successful state-action combination that lead to the goal fulfilment.

To overcome the limitation of RND, which estimates epistemic uncertainty solely from visited states, this thesis proposes *sars-RND*. *sars-RND* is an extension of RND that reduces the estimation bias by introducing additional information in the uncertainty estimation. *sars-RND* estimates epistemic uncertainty from a full RL tuple $(s_t, a_t, s_{t+1}, r_{t+1})$. As a result, *sars-RND* is expected to estimate a higher uncertainty in never encountered states while generalising across similar agent-environment interactions.

To show the limitation of RND and demonstrate the effectiveness of *sars-RND*, we test both models within the MT-PP benchmark environment. In this scenario, we let the agents visiting two states, shown in Figure 3.10, while sampling various actions to analyse the uncertainty curves generated by the two models.

State 1   State 2
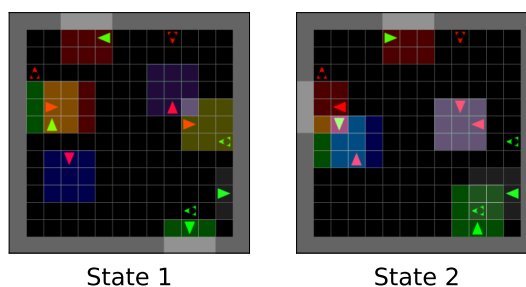
Figure 3.10: Two states in MT-PP environments.

To ensure a fair comparison, the two estimator models share the same architecture except for the input layer. *sars-RND* input layer is composed by a few extra neurons to accommodate action, reward and next state. To assess the uncertainties, agents take a fixed

action *hold* for 250 steps to strengthen the prediction across both estimators. Afterward, action sampled is changed to a new action *rotate left* for a shorter number of steps (25). Finally, agents resample *hold* for another 25 steps. The uncertainty trends are reported in Figure 3.11.
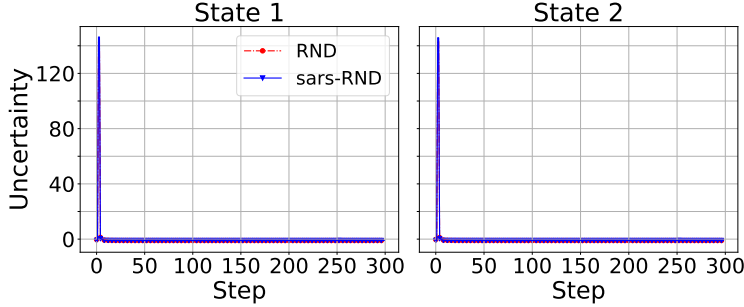
Figure 3.11: RND and *sars-RND* uncertainty curves, with their 95% confidence interval, while sampling different actions in MT-PP environment.

Overall, both RND and *sars-RND* uncertainty curves follow the same decreasing trend, as shown in Figure 3.11. The estimated uncertainty is high during the first steps and decreases asymptotically to 0 as the agents visit the states. However, by evaluating uncertainty in a narrowed interval as shown in Figure 3.11, RND keeps a flat trend while *sars-RND* registers a spike in uncertainty on the change of the action, firstly, at step (250), when action a_0 is replaced by a_1 and secondly, at 275 when a_0 is resampled. Second spike is smother compared to first as *sars-RND* recognises the already seen interaction. We observe that both for states under examination, state 1 and state 2.
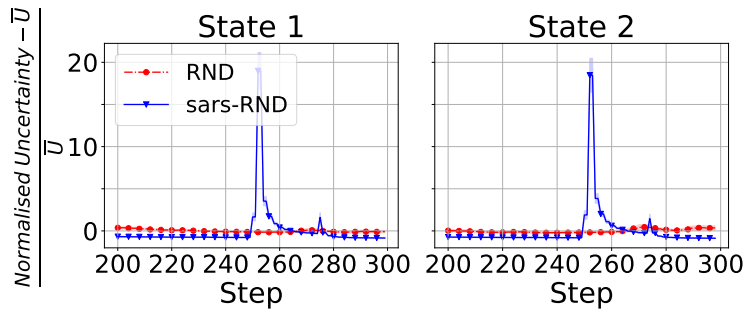
Figure 3.12: Zoomed in normalised uncertainties curves, with their 95% confidence interval, while sampling different actions in MT-PP environment.

This artificial study confirms that *sars-RND* offers a fine-grained estimation of uncertainty by considering both the executed action and the visited state. In contrast, RND is not sensitive to variations in the action taken. Moreover, the observed decreasing trend,

reported in Figure 3.11, suggests that *sars-RND* generalises effectively, as it can recognise familiar states even when sampling different actions. Consequently, the uncertainty estimated for never encountered states is expected to be higher than that for previously seen states.

Based on the experiments presented in this section, we conclude that *sars-RND* stands as a more suitable choice for online transfers in comparison to RND. The ability of *sars-RND* to estimate epistemic agent uncertainty from a full RL agent-environment interaction effectively overcomes the limitation of RND, while maintaining a comparable trend that gradually approaches 0 over time. Consequently, *sars-RND* is the model used with *EF-OnTL* in the online TL experiments *EF-OnTL*.

3.5   Conclusion

The evaluation study, presented in Section 3.3, on offline experience sharing based on the use of confidence measure establishes the foundation for the development of *EF-OnTL* for online experience sharing across multiple learning agents.

The evaluation results in MT-PP reveal that using agent confidence as a transfer decision metric can lead a target agent to improved performance. This trend is also confirmed by the action-advice based baseline *BS: confidence_based_e-decay*, as this approach enabled the target agent to achieve superior performance compared to other teacher-student framework baselines. Moreover, the sharing of batches of experiences has resulted into unmatched performance improvements for a target agent, outperforming action-advice baselines.

From MT-PP results it stands out that the quantity of shared information holds a stronger impact on a target agent compared to the filtering threshold used to select incoming experiences. Nevertheless, in an offline context, the threshold used for experience selection on the target side still contributes to facilitating positive transfer.

However, when transitioning from an offline to an online context and enabling the transfer across learning agents, relying on a fixed value to decide the tuples to be used by a target agent introduces a potential limitation as agents' confidence grow over time. Additionally, while tuning the transfer threshold can positively affect a target agent policy, this process

is tedious, requiring numerous trials due to the high sensitivity of the outcome to this para-meter, as observed in the experiments in MT-PP. Consequently, a pillar requirement for *EF-OnTL* is to enable the share of experience while removing the need for a hard-defined threshold.

When moving from offline to online experience sharing with *EF-OnTL*, we examine whether the counter-intuitive effect observed with the quantity of shared experiences remains valid.

Finally, given the promising outcomes observed during offline experience transfer between agents with different underlying demand patterns in 3R2S, we assess whether these findings are confirmed in an online experience sharing scenario.

4   *EF-OnTL* Design

This chapter presents the main contribution of this thesis, Expert-Free Online Transfer Learning (*EF-OnTL*), a novel online experience sharing framework that enables online transfer learning between agents in multi-agent systems with no fixed expert.

Section 4.1 motivates the design choices underlying *EF-OnTL*. Section 4.3 presents the *EF-OnTL* architecture. Section 4.4 introduces the necessary notation and presents the algorithm. Section 4.5 presents the transfer core engine, used to handle the knowledge transfer, by introducing both the teacher and the transfer content selection criteria. Section 4.6 presents the uncertainty estimator model used alongside *EF-OnTL* and, finally, this chapter ends with a summary in Section 4.7.

4.1   Motivation

Chapter 2 presented multiple Transfer Learning (TL) approaches differing by both the transferred object and the level of expertise possessed by the source of the transfer.

To summarise, when it is guaranteed that a dedicated expert is fully available to supervise novel agents, the most effective way to reduce the policy training time by injecting external knowledge into a novel target agent is to guide the target agent following the teacher-student framework with tailored advice, i.e., action or Q-values. The advice directly impacts the action decision process of the target agent.

On the other hand, when the provided advice is suboptimal, the target agent performance is capped by the degree of expertise of the source agent. In fact, the source's knowledge will eventually limit the capability of target agent to learn an effective policy. As a result, when

incorporating suboptimal advice within the action decision process, the risk of negative transfer overweights the potential benefit of TL.

To overcome the need of an optimal expert and enable TL across imperfect agents, this thesis presents *EF-OnTL*. *EF-OnTL* objective is to identify a subset of gathered experiences to be transferred that supplements and enhances the target learning process rather than overwrites it. Experience consists of agent-environment interactions that are visited by a source agent during its exploration process. As a result, through *EF-OnTL*, target agent enhances its own local standard Reinforcement Learning (RL) policy with external curated experience.

4.2 *EF-OnTL* Notation and Definitions

Prior to presenting in detail *EF-OnTL*, this section introduces the essential notation and definitions on which *EF-OnTL* is based. These concepts simplify the framework understanding and facilitate comprehension of the remaining chapter.

- N − number of RL-based agents available during a simulation;
- a set of *Agents* $A = \{A_1, \ldots, A_N\}$;
- a set of *Learning Processes* $LP = \{LP_1, \ldots, LP_N\}$, which are mapped to agents using the following function $f_1 = (A_i, LP_j) \in A \times LP \leftrightarrow i = j$, such as an agent has one and only one LP;
- a set of *Uncertainty Estimators* $UE = \{UE_1, \ldots, UE_N\}$, which are mapped to agents using the following function $f_2 = (A_i, UE_j) \in A \times UE \leftrightarrow i = j$, such as an agent has one and only one UE;
- a set of *Transfer Buffers* $TB = \{TB_1, \ldots, TB_N\}$, which are mapped to agents using the following function $f_3 = (A_i, TB_j) \in A \times TB \leftrightarrow i = j$, such as an agent has one and only one TB;
- B − a scalar value *Transfer Budget* that defines the exact number of experiences to be processed by a target agent during a single transfer step;
- TF − *Transfer Frequency* defined as number of episodes that occur between two consecutive transfer steps;

- *SS* − Source Selection Criteria (SS) used to select source of transfer;
- *TCS* − Transfer Content Selection Criteria (TCS) used to filter relevant knowledge to be transferred.

The notation and definitions provided above lay the base of *EF-OnTL* framework. These elements serve as a solid foundation for the architecture presentation and algorithm description presented in the following sections.

4.3   *EF-OnTL*: Architecture Design

EF-OnTL is a transfer learning framework for multi-agent systems that overcomes the need for an a priori determined experienced teacher by dynamically selecting a temporary expert at each transfer step based on the current performance of the agents, according to commonly defined metrics. *EF-OnTL* aims to improve the performance of individual agents through experience sharing. The selected agent is used as a source of transfer and some of its collected experience is made available to others. Subsequently, a target agent can filter and sample a batch of experiences to be integrated into its learning process and finally update its policy. The transferred batch contains agent-environment interactions sampled by the selected teacher and labelled with source's epistemic uncertainty (u^t): $(s^t, a^t, r^t, s^{t+1}, u^t)$. *EF-OnTL* is independent of the underlying RL algorithm used and thus, it can be exploited on a range of RL methods, both tabular and neural network-based ones.

Figure 4.1 depicts *EF-OnTL* system architecture that can be divided into three main components:

- RL Agent − It is the underlying RL algorithm used by the agent to learn a policy to address a task. Such model enables the agent to sample an action from a certain state. After taking the step, the agent forwards the observation, RL agent-environment tuple, to the learning model underneath that will be eventually updated.
- Transfer Core Engine − The Transfer Core is used to establish a connection with other agents and the main role is to filter the incoming experience that is tailored to the current policy of the agent. Transfer Core is detailed in Section 4.5.
- Uncertainty Estimator Model − The Uncertainty Estimator is used by the agent to
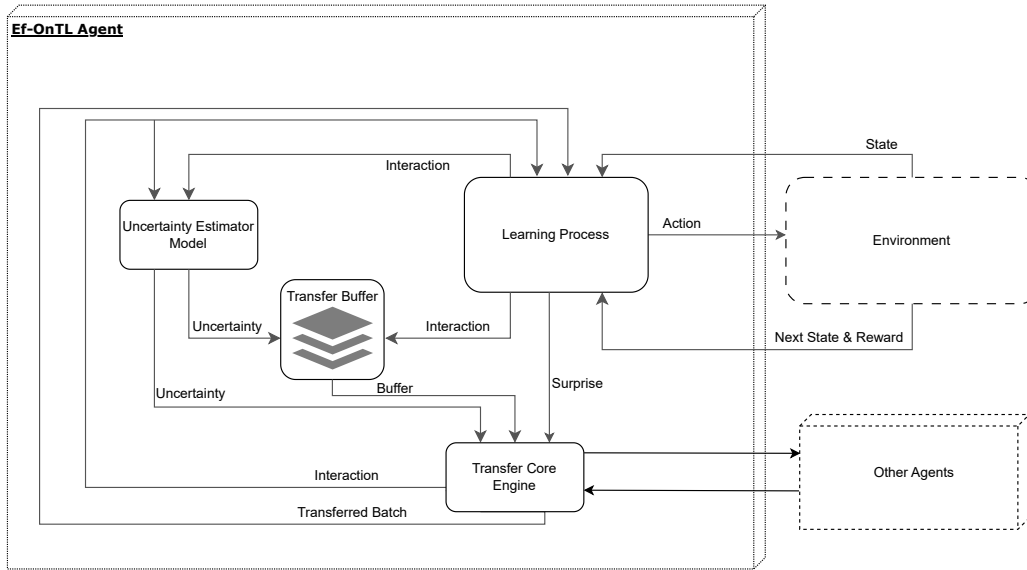
Figure 4.1: *EF-OnTL* agent components diagram.

approximate epistemic uncertainty. State Action Reward Next-state Random Network Distillation (*sars-RND*) is covered in Section 4.6.

The data processed by an *EF-OnTL* agent come from two distinct sources: an environment accessed online in a traditional RL manner, and other *EF-OnTL* agents. Beyond the traditional update of the learning process, the interactions sampled from the environment are fed into the uncertainty estimator model, which estimates the epistemic uncertainty related to each tuple. Subsequently, each tuple, along with its estimated uncertainty, is stored in an internal buffer, named *Transfer Buffer*. This process is repeated iteratively until the transfer time is reached.

When transfer is initiated, the transfer core engine mutually exchanges the internal transfer buffer with other agents. Each interaction contained within the transfer buffers is then fed into the learning process, to estimate the surprise value, and the uncertainty estimator model, to estimate the uncertainty. Based on these two metrics, a source agent is selected and a subset of tuples is filtered to shape the transferred batch. Finally, the learning process is updated based on this transferred batch.

4.4 *EF-OnTL* Framework

This section delves into the online experience sharing workflow followed by *EF-OnTL* agents. The agent exploration and learning process is independent, except during the transfer step. When the transfer begins, a common agent is selected as the source of transfer, and the reaming agents filter a subset of the incoming experiences to update their learning process based on the selected tuples.

Algorithm 1 introduces high level procedure followed by agents for sharing experience one to another throughout their simultaneous exploration processes.

Algorithm 1 Expert-Free Online Transfer Learning

1: Given: A, LP, UE, TB, B, TF, SS, TCS, f_1, f_2, f_3
2: **for** ep in *Episodes* **do**
3:     **for all** $A_i \in A$ **do**   ▷ follow normal policy
4:         get state s_i^t for A_i at time t
5:         sample an action a_i^t based on LP_i
6:         perform a step and observe $o_i^t = (s_i^t, a_i^t, r_i^t, s_i^{t+1})$
7:         estimate uncertainty $u_i^t = UE_i(o_i^t)$
8:         push (o_i^t, u_i^t) to TB_i ▷ FIFO queue
9:         optimise UE_i on o_i^t
10:        optimise LP_i
11:     **end for**
12:     **if** $ep \ \% \ TF$ is 0 **then**   ▷ start transfer-step
13:         select source agent A_s by SS
14:         **for all** $A_t \in (A \setminus A_s)$ **do**   ▷ transfer from A_s to A_t
15:             apply TCS over TB_s and sample B tuples
16:             optimise LP_t with the sampled tuples
17:         **end for**
18:     **end if**
19: **end for**

First, line 1 defines the *EF-OnTL* parameters. Then, agent takes a standard RL step at lines $3 - 6$, i.e, agents retrieve observation from an environment, sample an action based on their learning process and finally take a step within the environment.

Afterward, the related UE_i estimates i-th agent's epistemic uncertainty u_i^t over the observed agent-environment tuple $(s_i^t, a_i^t, r_i^t, s_i^{t+1})$, then UE_i model is updated based on new sampled evidence and the uncertainty-labelled tuple is published to the associated transfer buffer TB_i as shown in lines $7 - 9$. When TB_i is at full capacity, new labelled interactions replace the oldest tuples following a FIFO scheme. Learning process LP_i is then updated based on the underlying RL algorithm used, lines $10 - 12$.

Lines $14 - 20$ show a transfer step. Firstly, at line 15, source is selected among the agents w.r.t. Source Selection Criteria SS. Secondly, remaining agents, i.e., targets, apply a filtering function TCS over source transfer buffer and then sample a batch composed by a fixed number B of tuples, line 17. Finally, at line 18, each target agent updates its learning process based on the obtained batch.

While the above provides a full description of the algorithm, for further clarity, Figure 4.2 summarises $EF\text{-}OnTL$ workflow, at a single TL time-step t, for the simplest scenario with only 2 agents.
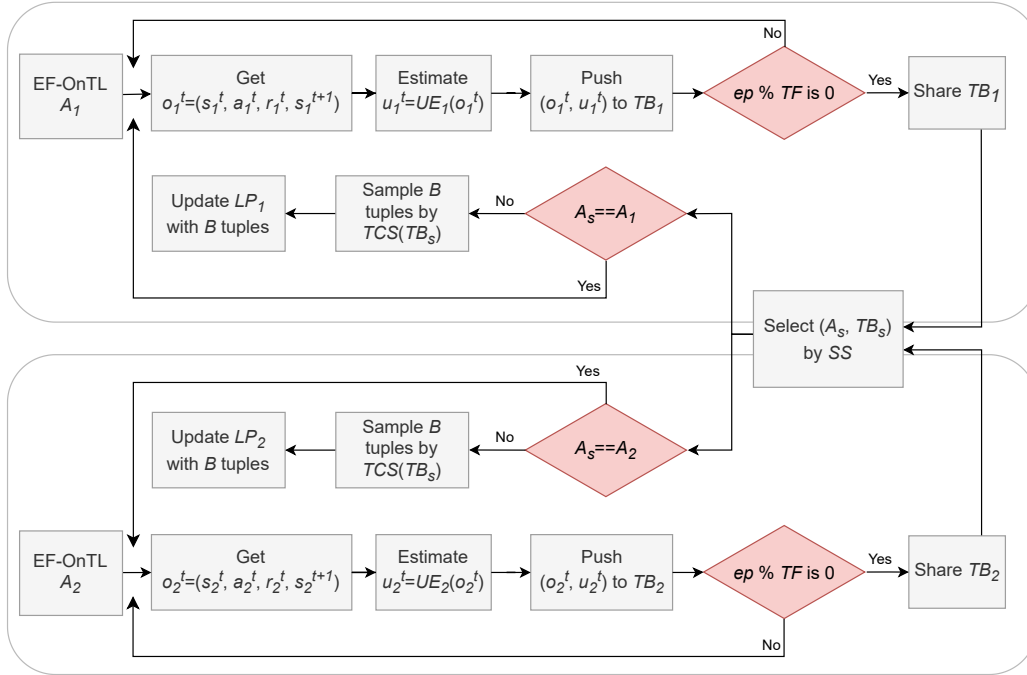
Figure 4.2: $EF\text{-}OnTL$ simplified workflow in a two agents scenario.

In detail, the source agent A_s and the related source transfer buffer TB_s are globally selected, as the SS metrics, further presented in Section 4.5.1, do not consider the target agent to identify the source of transfer. Consequently, when implementing multiple $EF\text{-}OnTL$ agents, the source selection process can be either centralised or decentralised, based on specific needs. However, both implementations are equivalent given the SS metrics. When $EF\text{-}OnTL$ is applied on more than two agents, all the agents share their transfer buffer one to another.

To further zoom into $EF\text{-}OnTL$ process, this section proceeds by showing the sequence of operations within an $EF\text{-}OnTL$ agent, providing insights into how an $EF\text{-}OnTL$ agent

operates internally. The internal agent perspective aims to provide a fine-grained analysis on how each sub-modules within *EF-OnTL* is used and interacts one to the other throughout the learning and transferring stage. To enable a detailed internal view, the diagram in Figure 4.3 divides the agent into fine-grained sub-modules.
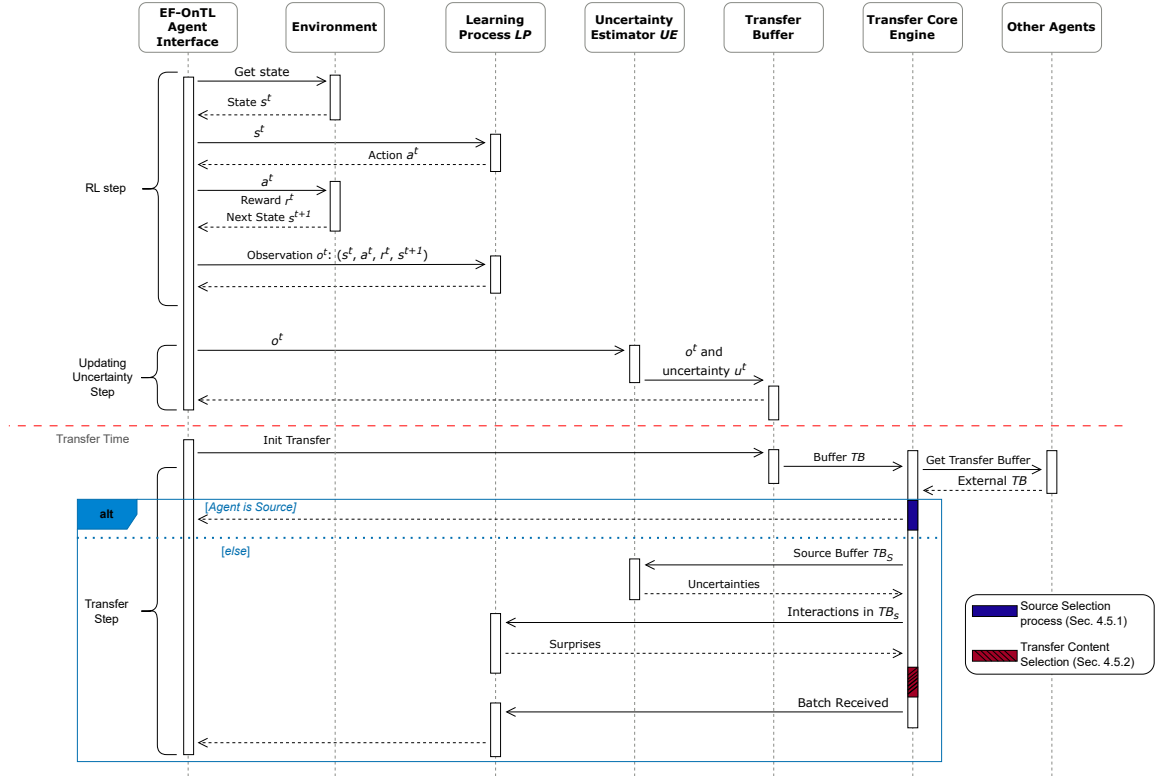
Figure 4.3: *EF-OnTL* agent sequence diagram.

There are 3 main sequence blocks that can be identified:

1. *RL step* − describes the RL agent-environment interaction, lines 4 to 6 in Algorithm 1. Agent uses its learning process *LP* to sample an action based on the observed state provided by the environment. After taking that action, agent observes the reward returned by the environment along with the next state. Finally, the full observed tuple (o^t), that will eventually be used to update the underlying learning policy, is passed to the Learning Process.

2. *Updating Uncertainty step* − o^t is forwarded to the *Uncertainty Estimator Model*, which approximates the epistemic uncertainty (u^t) and updates the underlying model as shown in line 9 of Algorithm 1. Finally, (o^t, u^t) is pushed to the associated *Transfer Buffer*.

3. *Transfer Step* − When the time to transfer comes, as per line 14 in Algorithm 1, an agent initialises the transfer by sending its transfer buffer to the *Transfer Core Engine*. The *Transfer Core Engine* collects the transfer buffers from other agents and decides whose the suitable agent that should act as teacher, as shown at line 15 of Algorithm 1.

   When an agent is chosen as source of transfer, the knowledge transfer process terminates for that agent, and it is not influenced by external knowledge. On the other hand, if the selected teacher is an external agent, the transfer begins. Both the *Uncertainty Estimator Model* and the *Learning Process* are used to estimate the uncertainties and surprises for each tuple within the *Source Transfer Buffer*. Afterward, the *Transfer Core Engine* filters and samples a certain number of tuples that are used to update the *Learning Process*. Finally, the transfer step terminates and the control is returned to the agent.

During a transfer step there are the two selection processes, source selection and transfer content selection, that are not captured in detail within this diagram. Next section focuses on the *Transfer Core Engine* presenting these two filtering functions and the criteria used to select source of transfer and relevant knowledge.

4.5   Transfer Core Engine

The *Transfer Core Engine* is the internal part of *EF-OnTL* agent that manages the transfer step. Its role is to interface with other agents to exchange the transfer buffers and to select relevant knowledge to be used to update the RL learning process. The details of its architecture are shown in Figure 4.4.

The *Transfer Core Engine* has two objectives: 1) identify the A_s, the agent used as source of transfer; 2) filter the relevant incoming knowledge to update the *Learning Process*. The remainder of this section focuses on these two modules: Section 4.5.1 presents the metrics used to select A_s alongside the related transfer buffer TB_s. Then Section 4.5.2 presents TCS, the selection criteria applied by the target agent to select relevant knowledge.
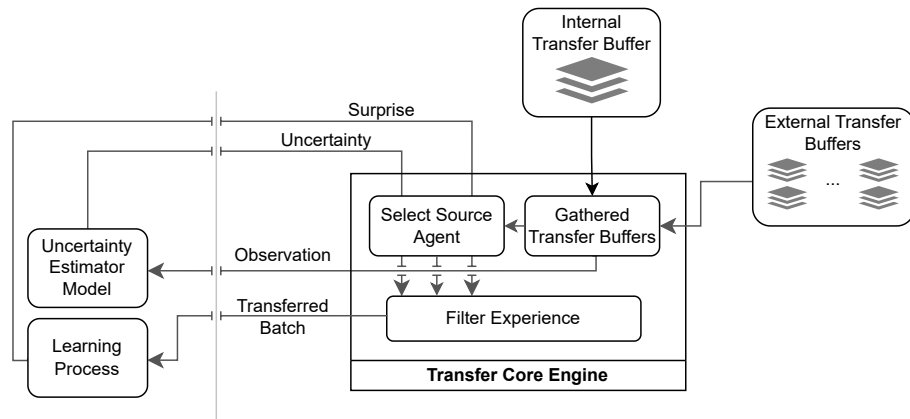
Figure 4.4: Detail of *EF-OnTL* Transfer Core diagram.

4.5.1 Source Selection

In TL approaches where an expert is readily available for unidirectional knowledge transfer, as the one described in Chapter 3, the source selection task is not required. However, in *EF-OnTL*, given the continuous online transfer across multiple agents, source selection becomes crucial to achieve positive transfer and to dynamically assign roles at the beginning of each transfer step.

One of the key challenges in *EF-OnTL* is to identify the most suitable agent among multiple candidates with similar expertise, as shown at line 13 of Algorithm 1, which allows other agents to benefit from the sharing of the source's knowledge.

To identify the transfer source from a set of candidates, this work assumes that agents can communicate one to the other with no limitations and at a fixed cost; in reality the communication cost and range will be limited confining the communication to a subset of agents.

As discussed in Chapter 2 for Agent-to-Agent (A2A) TL, the source selection decision process can rely on confidence or uncertainty metrics defined on target and sender, performance based metrics, or can be based on completion of an episode. The ultimate choice depends upon the transfer objective.

The majority of experience-based TL work does not consider the source selection problem, as demonstrations are assumed to be coming from agents that have already finished their training. The closest work related to *EF-OnTL* are Online Confidence-Moderated

Advice Sharing ($OCMAS$) [52] and Wang and Taylor [114, 115]. $OCMAS$ is an online action-advice based TL where the problem of source selection is replaced by majority voting as the final advice is the result of majority voting over a set of incoming action-advice. Wang and Taylor [114, 115] use uncertainty to select the external entity that should provide action-advice. In *EF-OnTL*, agents use uncertainty and cumulative reward to select the source of transfer. Instead of providing a single action-advice to a target agent selected by majority voting or provided by an external entity, *EF-OnTL* provides to a target agent a subset of experiences to overcome the knowledge gap between the target and the source policy. This thesis studies two different methods to select the transfer source within the Transfer Core Engine:

- Average Uncertainty ($\overline{U}$) − it selects source of transfer as the agent with lowest average uncertainty $\overline{u_i}$ on the tuples stored within its transfer buffer TB_i. Formally, that means that the source agent A_s is selected accordingly to Eq. 4.1.

$$
\begin{aligned}
A_s &\leftarrow min_{i=0}^{N}(\overline{u_i}) \\
\overline{u_i} &= \frac{\sum_{t=0}^{|TB_i|} u_i^t \in TB_i}{|TB_i|}
\end{aligned}
\tag{4.1}
$$

- Best Performance (BP) − it relies on performance achieved by agents over a predefined window of most recent episodes and it is inspired by Taylor et al. [49]. As performance measure, *EF-OnTL* uses average cumulated reward over episodes' finite-horizon undiscounted return ($\overline{R_i}$). A_s is selected out of average sum of rewards returned by the environment over episodes from initial to goal state with a finite number of steps. Eq. 4.2 reports the formula used for the source selection, where E is the number of evaluated episodes and τ_i^e represents the length of the *e-th* episode for the *i-th* agent.

$$
\begin{aligned}
A_s &\leftarrow max_{i=0}^{N}(\overline{R_i}) \\
\overline{R_i} &= \frac{\sum_{e=0}^{E} \sum_{t=0}^{\tau_i^e} r_i^{e,t}}{E}
\end{aligned}
\tag{4.2}
$$

The two SS criteria, $\overline{U}$ and BP, address the challenge of identifying a suitable agent to act as the source of transfer from a set of candidates, enabling dynamic roles for a transfer step. While these metrics are generally used to establish whether an agent is in need of advice in specific circumstances, *EF-OnTL* uses these metrics for selecting the source of transfer, which allows the sharing of a batch of experiences aiming to overcome the gap within a target policy.

In Chapter 6, both techniques described by Eq. 4.1 and Eq. 4.2 are evaluated in a mutually exclusive manner. This evaluation aims to identify the performance impact of the two techniques to establish whether one prevails over the other in certain conditions or scenarios.

4.5.2   Transfer Content Selection

Once the appropriated source of transfer is identified, the target agent has to accurately select the specific knowledge relevant to its own learning process, as shown in line 17 of Algorithm 1. Therefore, this section introduces TCS to prioritise certain tuples based on their score over a set of measures.

To simplify the readability, this section discusses the simplest case with two agents, which, at each transfer step, are referred to as source agent A_s and target agent A_t. Experience worth to be transferred is identified through two criteria, *expected surprise* as in [53] and *uncertainty* as in [114, 115].

Gerstgrasser et al. [53] use Temporal Difference error (TD-error) to select agent-environment tuples that surprised a certain agent, and consequently, these tuples are forwarded to all other agents by injecting the tuples into their replay buffer. However, this approach could potentially limit the capability of other agents to store and learn from more specific knowledge, as certain tuples might not have the same impact on multiple different policies. Wang and Taylor [114, 115] use the uncertainty to provide action as advice from a centralised entity and to decide whether to optimise the central entity with further demonstration collected online.

In *EF-OnTL*, *expected surprise* [71], is defined on the target *Learning Process* and is approximated through TD-error. Despite expected surprise is not used as much as uncertainty

or confidence to filter relevant knowledge in other TL work, we propose it as transfer criteria in this thesis based on its proven usefulness as selection criteria for prioritising experience selection in prioritised experience replay [104].

In *EF-OnTL*, *uncertainty* is estimated by an uncertainty estimator model UE with a standardised architecture. Both source and target agent have access to their own estimator, respectively UE_s and UE_t that are updated based on their local experiences. Consequently, their estimated uncertainties can be compared to select relevant tuples to be transferred. In detail, given (o_s^i, u_s^i), where o_s^i is an agent-environment tuple visited by A_s at time i and u_s^i is its associated epistemic uncertainty, then A_t estimates current uncertainty $u_t^{o_s^i}$ over the interaction o_s^i sampled by A_s. Hence, $u_t^{o_s^i} = UE_t(o_s^i)$. The discrepancy between these two estimations, delta confidence ($\Delta - conf$), can be defined as $\Delta - conf = u_t^{o_s^i} - u_s^i$. Note that $u_t^{o_s^i}$ is an epistemic estimation at the time of transfer and changes over time.

Based on these two criteria, i.e., surprise and uncertainty, A_t receives a personalised batch of experiences that aims to fill shortcomings in its policy. We define multiple filtering functions for incoming knowledge as follow:

1. *rnd* Δ-*conf* − transfer Randomly based on threshold determined by Delta Confidence. By choosing this technique, an agent randomly samples B tuples with an associated $\Delta - conf$ higher than the median value computed across all tuples.

2. *high* Δ-*conf* − transfer based on Higher Delta Confidence. Agent sorts in decreasing order the incoming tuples by the Δ-*conf* and selects the top B entries.

3. *loss & conf* − transfer based on Higher Loss and Confidence combined. While previous filters are defined only over uncertainty, this one also considers expected surprise. To balance the different scales, uncertainty and surprise values are normalised within a $[0, 1]$ interval and then weighted equally. Finally, target agent sorts the incoming tuples by the computed value and then selects the top B tuples with the highest values.

The three TCS, *rnd* Δ-*conf*, *high* Δ-*conf* and *loss & conf*, enable a target agent to filter the incoming knowledge facilitating dynamic transfer content selection. As a result, each target agent can select a subset of the incoming experiences that is expected to improve its current learning process. In Chapter 6, all the three TCS criteria are evaluated and

compared in a mutually exclusive manner. The comparison aims to identify the performance impact of filtering the incoming experience based on different selection criteria to establish whether one prevails over the other in certain conditions or scenarios.

4.6 Uncertainty Estimator Model

This section presents the uncertainty estimator model used within *EF-OnTL* to support the source selection decision process and the filtering of relevant knowledge to be integrated within the learning model of the agent.

As discussed in Section 2.3, generally, in RL, the epistemic uncertainty of an agent is approximated from the observed state returned by the environment. In simple scenarios a state-visit counter can be used to approximate the uncertainty. On the other hand, when it is impossible to keep track of all the different visited states, neural network-based models, as Random Network Distillation (RND) [70], can be used as approximators of state-visit counter. RND uses two neural networks, referred as target and predictor, to approximate the epistemic uncertainty through Mean Squared Error (MSE) of the networks' output. Over time, the predictor is optimised to predict the target output and the uncertainty is expected to decrease accordingly.
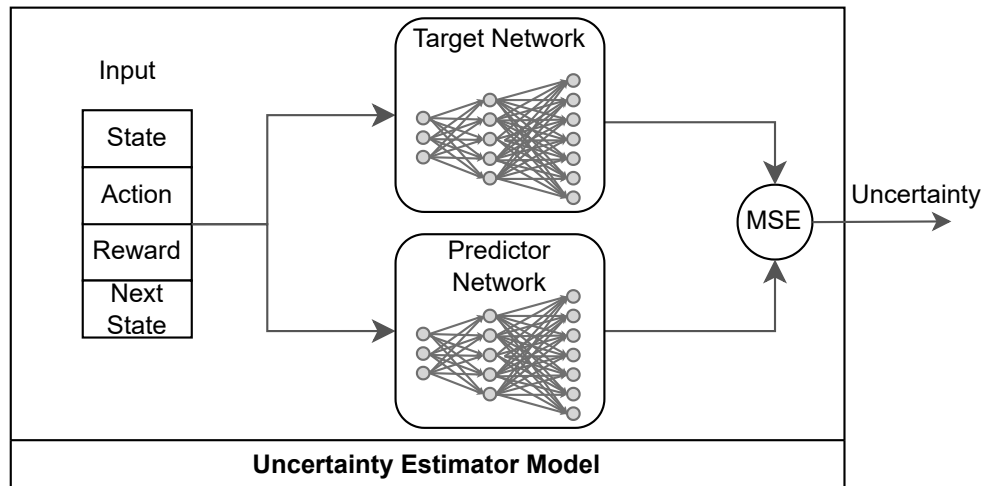
Figure 4.5: *sars-RND* architecture.

When used in an online TL scenario, RND is expected to lack crucial informations that are necessary to estimate the epistemic uncertainty of an agent, i.e., action taken and the

updated state. To address this, *sars-RND*, shown in Figure 4.5, an estimator model based on RND and expected to overcome the limitations of the base model is introduced. *sars-RND* extends RND by estimating uncertainty from a RL tuple $(s_i^t, a_i^t, r_i^t, s_i^{t+1})$, composed by state, action, reward and next state.

By taking into consideration the additional features, *sars-RND* is expected to provide a more accurate estimation throughout the learning phase when compared to RND. The extra inputs of *sars-RND* are expected to provide a more accurate estimation by enabling the agent to discriminate situations where the goal is closer but not yet explored because the action that lead to the accomplishment of the goal has never been explored.

Section 3.4, has demonstrated *sars-RND* superiority in estimating the agent epistemic uncertainty in an online scenario when compared against RND.

4.7   *EF-OnTL* Design Summary

In most Transfer Learning (TL) frameworks, knowledge is typically transferred from an optimal fixed expert to learning agents. However, if the source of advice is suboptimal, the capability of a target agent to learn an optimal policy might be constrained by the suboptimal policy. To overcome this limitation and enable TL across multiple learning agents, in this chapter we presented *EF-OnTL*. *EF-OnTL* is an online experience sharing framework that facilitates TL among multiple learning agents by transferring a buffer of agent-environment interactions.

EF-OnTL expands the architecture of a traditional RL agent by incorporating an uncertainty estimator model (UE), a transfer buffer (TB), and a transfer core engine for each agent. The UE is used to estimate the epistemic confidence of an agent. Subsequently, the observed agent-environment interaction labelled with uncertainty is stored in the transfer buffer. Finally, the transfer core engine handles the communication with other agents and finalises the transfer process.

To enable online experience sharing, the transfer core engine in *EF-OnTL* must address two challenges: 1) identifying a suitable temporary expert to act as the source of transfer, and 2) selecting the relevant experiences that contribute to the target policy.

For source selection, *EF-OnTL* defines two Source Selection Criteria (SS): Average Uncertainty ($\overline{U}$) and Best Performance (BP). $\overline{U}$ selects the source of transfer as the agent with the lowest average uncertainty within the transfer buffer, while BP selects the source agent based on the average cumulated reward over episodes' finite horizon undiscounted return.

To filter experiences relevant to a specific target agent, *EF-OnTL* defines three Transfer Content Selection Criteria (TCS) based on two criteria: expected surprise and delta confidence (Δ-*conf*). Expected surprise is approximated by the Temporal Difference error, and it is used to select tuples that are expected to surprise a target agent. On the other hand, Δ-*conf* is defined as the difference between the uncertainty of the target agent and the uncertainty of the source agent. Consequently, the three TCS used in *EF-OnTL* consist of transfer Randomly based on threshold determined by Delta Confidence (*rnd Δ-conf*), transfer based on Higher Delta Confidence (*high Δ-conf*), and transfer based on Higher Loss and Confidence combined (*loss & conf*).

The criteria used by *EF-OnTL* to select the transfer source and relevant knowledge are not new within the TL community. For instance, uncertainty has been used to identify the source of transfer [114, 115] and to identify the situation where an agent is in need of advice [52], while the idea of using performance based metrics are inspired by Taylor et al. [49]. Similarly, the surprise concept has been used by Gerstgrasser et al. [53]. However, to the best of our knowledge, *EF-OnTL* is innovative in applying these criteria to address the combined challenge of dynamically selecting the source of transfer and dynamically determining the content transferred to each individual target agent. Moreover, uncertainty has been primarily utilised to decide whether to provide an one-time advice.

To promote positive TL outcome, State Action Reward Next-state Random Network Distillation (*sars-RND*) is introduced and used as UE in this thesis. *sars-RND* is a model that enables an agent to estimate epistemic uncertainty from an agent-environment interaction. Based on the uncertainty and on other commonly defined metrics, i.e., expected surprise and cumulative reward, agents can dynamically select a source of transfer and filter the experiences from an incoming transfer buffer. This selection process aims to identify a batch of experiences that are expected to improve the target's policy.

5   Implementation

This chapter discusses the hardware and software technologies used for conducting the research presented within this thesis. Section 5.1 introduces the libraries and the hardware used to run the experiments, collect the data and process the results. Then, Section 5.2 covers the design of the Reinforcement Learning (RL) algorithms used for the evaluation. Finally, this chapter ends with Section 5.3 that discusses the implementation details of the main contribution of this thesis, Expert-Free Online Transfer Learning (*EF-OnTL*).

5.1   Technology for Experiments

This section presents the necessary libraries used to carry out the experiments presented in this work.

All algorithms and experiments are developed in Python 3. The neural networks are implemented using PyTorch version 1.13.1+cu117 to enable the models run on a GPU with CUDA. Performance and results are stored as CSV files and visualised in real time through TensorBoard V. 2.12.0. Finally, data are processed and analysed using Pandas V. 1.5.3, NumPy V. 1.23.0 and graphically displayed through Matplotlib V. 3.4.0.

5.1.1   Hardware

The results presented in this thesis have been collected on two distinct machines. The first machine used to carry out the experiments is a consumer-oriented laptop, specifically a Dell XPS model. Its hardware configuration details are reported in Table 5.1.

The second machine used is a custom rack setup with multiple GPUs. The hardware and software details are provided in Table 5.2.

Table 5.1: Laptop hardware specs used for experiments.

Parameter	Value
CPU	Intel i7-9750H @ 2.60GHz
GPU	NVIDIA GTX 1650 Mobile
NVIDIA-SMI Driver	470.182.03
CUDA Version	11.4
RAM	2×8 GiB DDR4 @ 2667 MHz
OS	Ubuntu 20.04.5 LTS

Table 5.2: Server hardware specs used for experiments.

Parameter	Value
CPU	Intel Xeon(R) Gold 5220R @ 2.20GHz
GPU	4× RTX 3070
NVIDIA-SMI Driver	530.30.02
CUDA Version	12.1.1
RAM	1535 GiB
OS	Ubuntu 20.04.6 LTS

Finally, Table 5.3 specifies which device has been used for each step of the experiment-
ation pipeline on each benchmark environment. First column, *Benchmark*, indicates the
environment, second column, *Debug*, indicates the hardware used to implement and per-
form preliminary experiments and, last column, *train & test*, reports the machine used to
collect the presented results. The machine used can vary between the custom rack (*server*),
and the consumer-oriented laptop (*XPS*).

Table 5.3: Details of simulations.

Benchmark	Debug	Train & Test
Cart-Pole	XPS	server
MT-PP	XPS	server
HFO	XPS	XPS
3R2S	XPS	server

The neural networks implemented are relatively simple, further details are provided in
Section 5.2, the computation cost mainly results from sampling the agent-environment trans-
itions and by back-propagating the losses on the multiple networks. As such, to replicate
the experiments presented within this thesis, GPU is not mandatory as the true bottleneck
are the CPU capabilities, although, having a configuration similar to the server mentioned
above helps to parallelise the computation and lower the overall time needed to collect the

results. For instance, to complete a training and testing phase in the Multi-Team Predator-Prey (MT-PP) environment takes around 7 hours on the XPS laptop, which is more than doubled on the server.

5.2 Deep Reinforcement Learning Algorithms

This section outlines the architecture design for the deep RL algorithms used to learn the agent's policy. For the *EF-OnTL* experiments in this thesis, there are 2 RL algorithms used: dueling Deep Q-Network (DQN), discussed in Section 5.2.1, and Parameterised Action Deep Deterministic Policy Gradient (PA-DDPG), discussed in Section 5.2.2. Additionally, for the MT-PP benchmark environment we defined an encoder to process the multichannel state. The encoder is presented as stand-alone as it has been used for both learning the agent's policy and in the uncertainty estimator model. The implementation of the encoder is detailed in Section 5.2.3.

5.2.1 Dueling Deep Q-Network

Dueling Deep Q-Network is the model chosen to replace Proximal Policy Optimisation (PPO) model, due to the on-policy limitation of PPO, and to enable the sharing of experience. This model has been used in tasks with a discrete action space, i.e., Cart-Pole, Multi-Team Predator-Prey (MT-PP) and Ride-Sharing Ride-Requests Simulator (3R2S). Dueling DQN, depicted in Figure 2.3, decouples the learning of a state-value function and the advantage of selecting a certain action. As a result, dueling DQN improves the estimation of the Q-values when compared to standard DQN.

Table 5.4 reports the setup used in Cart-Pole, MT-PP and 3R2S. Betas is a decaying parameter used within the Adam optimiser and the Policy Update Step defines the quantity of mini-batches used to update the online network before copying the weights of the online network into the target network.

Dueling DQN is designed for environments with discrete action spaces and is not suitable to handle continuous action space. Therefore, to enable the learning of a policy in

Table 5.4:   Dueling DQN parameters setup.

Parameter	Cart-Pole	MT-PP	3R2S
Input Layer	Linear(4, 128)	Linear(135, 256)	Linear(26, 256)
Hidden Layer(s)	Linear(128, 64)	n.a.	Linear(256, 128) Linear(128, 64)
Advantage Layer	Linear(64, 2)	Linear(256, 5)	Linear(64, 5)
Value Layer	Linear(64, 1)	Linear(256, 1)	Linear(64, 1)
Activation	ReLU	ReLU	ReLU
Optimiser	Adam	Adam	Adam
Learning Rate	1e-4	1e-5	1e-4
Betas	(.9, .999)	(.9, .999)	(.9, .999)
Gamma	.999	.999	.999
Mini-batch Size	32	32	32
Policy Update Step	1,000	10,000	2,500

environment with continuous control space, i.e., Half Field Offense (HFO), in this thesis we use an off-policy based method described in the following section.

5.2.2   Parameterised Action Deep Deterministic Policy Gradient

Parameterised Action Deep Deterministic Policy Gradient (PA-DDPG) [8] is an extension of Deep Deterministic Policy Gradient (DDPG) and is an off-policy based RL algorithm. PA-DDPG enables an agent to learn a policy in an environment with parametrised action space, i.e. Half Field Offense (HFO), where an agent selects a discrete action along with the parameters required to parametrise the chosen action.

The PA-DDPG architecture shown in Figure 2.5 consists of an actor with a 2-stream output to select both discrete action and parameters, and a critic network to estimate the Q-value for a state and an action.

The details of the PA-DDPG network used in the HFO experiments are reported in Table 5.5. Both the actor and the critic share the same configuration for the hidden layers. The input layer of the critic consists of the aggregation of three components: 95 neurons for the state, 4 neurons for the discrete action selection, and 7 neurons for the parameters of the actions. In total, the input layer processes 106 values.

Table 5.5: PA-DDPG parameters for HFO

Parameter	HFO
actor	
Input Layer	Linear(95, 1024)
Hidden Layers	Linear(1024, 512)
	Linear(512, 256)
	Linear(256, 256)
	Linear(256, 128)
	Linear(128, 128)
Output Discrete Action Layer	Linear(128, 4)
Output Parameters Action Layer	Linear(128, 7)
Activation	leaky relu
critic	
Input Layer	Linear(106, 1024)
Hidden Layers	Linear(1024, 512)
	Linear(512, 256)
	Linear(256, 256)
	Linear(256, 128)
	Linear(128, 128)
Output Layer	Linear(128, 1)
Activation	leaky relu
Optimiser	Adam
Learning Rate	1e-3
Betas	(0.9, 0.999)
Target Policy Update	soft
Policy Update τ	1e-3
Mini-batch Size	32

5.2.2.1 Multi-Agent Parameterised Action Deep Deterministic Policy Gradient

In the multi-agent implementation of PA-DDPG, the architecture is similar to the single-agent PA-DDPG with once crucial difference, the critic is centralised. The centralised critic takes as input the state-action pairs of each agent and estimates the state-value of the joint actions.

While the hidden and output layers of the critic remain unchanged from the parameters previously presented in Table 5.5 for the single-agent PA-DDPG, the input size is multiplied by the number of agents that share the centralised critic.

5.2.3   Multi-Team Predator-Prey Encoder

For Multi-Team Predator-Prey (MT-PP), the observation of an agent is defined as a 3-channel array encoding the type of object within a cell, the orientation and the team-membership. The orientation of an object and its team membership are meaningful only for predator and prey.

Table 5.6: Preprocess encoder for MT-PP.

Parameter	Value
$1^{st} layer$	Conv1d(3, 7, k=1)
$2^{nd} layer$	Conv1d(7, 15, k=1)
$3^{rd} layer$	Flatten(out_dim=-2)

In this thesis, the experiments provided are based on an encoder to pre-process the multi-channel observation and to flatten it into a single dimension array, Table 5.6 reports its configuration. The encoder uses a 1D convolution with a kernel size of 1 to capture local patterns and dependencies within the input. The choice of a kernel size of 1 is because there are no relationships between neighbouring cells in the grid.

5.3   Expert-Free Online Transfer Learning Implementation Details

EF-OnTL is extensively detailed in Chapter 4, where both the architecture and the workflow are thoroughly described to facilitate the transferring of experiences among agents. The *EF-OnTL* code is accessible at the following link: https://github.com/acastagn/EF-OnTL. In summary, beyond the core RL process, *EF-OnTL* incorporates three additional components:

1. a copy of State Action Reward Next-state Random Network Distillation (*sars-RND*), class *UE* in the *SARSrnd* file available in the repository, to estimate the agent's epistemic uncertainty. Each agent has associated a dedicated copy of *sars-RND* that needs to be updated solely on the experiences coming from the related agent;

2. a transfer buffer, class *buffer* in the *transferBuffer* file available in the repository, to store the uncertainty labelled experiences in form of a seven-element tuple

$(s^t, a^t, r^t, s^{t+1}, d^t, u^t);$

3. Transfer Core Engine, file *transferCoreEngine* in the repository, to enable the sharing of experience and the execution of the two processes needed to select the source and the content to be transferred.

EF-OnTL is designed to enable multiple agents to transfer experience in a distributed manner. However, for sake of simplicity, some operations can be executed in a centralised manner to reduce the implementation complexity, i.e., *Source Selection Process*. The goal of this task is to select a source of transfer which is globally selected for all the agents. As the selection of source is not influenced by any external factor, the execution of this task can be executed in a centralised manner by an external entity to reduce the communication across agents.

Regardless of the *Source Selection process* implementation, the following step, *Transfer Content Selection*, must be distributed on the agent side as this step is based on the target agent. To estimate the surprise values for the *Transfer Content Selection*, the learning process of an agent must enable the estimation of the Temporal-Difference (TD)-error for an agent-environment interaction. In addition, the learning process has to allow for taking a training step based on an external buffer passed as parameter.

6 Evaluation

This chapter presents the evaluation study performed on Expert-Free Online Transfer Learning (*EF-OnTL*). *EF-OnTL* enables knowledge transfer across multiple agents by sharing experience as a batch of agent-environment interactions. *EF-OnTL* enables a target agent to receive, filter and process a subset of the transferred experiences which is expected to expedite the convergence of its policy.

The remainder of this chapter is organised as follows. The evaluation objectives for *EF-OnTL* are introduced in Section 6.1, along with the baseline approaches and associated evaluation metrics. The benchmark environments are detailed in Section 6.2. Subsequently, the evaluation is organised into three sections and is presented in Section 6.3. Section 6.3.1 compares *EF-OnTL* with the baseline methods. Section 6.3.2 examines the impact of different transfer settings in *EF-OnTL*. Section 6.3.3 assesses the adaptation capabilities of *EF-OnTL* when transferring across tasks with heterogeneous dynamics. Lastly, Section 6.4 summarises the evaluations presented in the chapter.

6.1 Evaluation Objectives

This section, outlines the core objectives, denoted as *O1* to *O9* underlying the experiments conducted and presented within this thesis:

O1 − The first objective of evaluation is to assess whether *EF-OnTL* improves convergence time compared to a no-transfer baseline, in which agents do not share any information and learn independently. Achieving quicker convergence is the fundamental goal for any experience-based offline Transfer Learning (TL) framework aiming to outperform a transfer-free baseline. To assess speed convergence, in all benchmark environments

we also implement a no-transfer baseline where agents learn independently, more details are provided in Section 6.1.1, which will be used to compare *EF-OnTL*. The convergence time will be analysed through the comparison of the learning curves defined on the cumulated reward over an episode or a specified interval;

O2 − The second objective evaluates whether and to what extent *EF-OnTL* improves the agents' performance when compared to a no-transfer baseline, in which agents do not share any information and learn independently, therefore the baseline used in this objective is the same as in *O1*, but we compare the performance against a different metric. When imperfect agents are used as a source of transfer, the convergence speed alone may not be adequate to analyse the transfer impact. Furthermore, by transferring agent-environments interactions, the effect of transfer on a target agent may be delayed. This objective aims to determine whether *EF-OnTL*, through online experience sharing, enables agents to achieve overall better performance after convergence when compared to transfer-free agents. The performance improvements will be analysed by comparing the performance achieved by agents at a fixed episode, while following their policies.

O3 − The third objective evaluates whether *EF-OnTL* achieves comparable performance when compared to a TL scenario following the teacher-student paradigm. In the baselines used, the source of transfer influences directly the action-decision process of a target agent by providing an action as advice. The goal of this objective is to determine if *EF-OnTL* can be a valid alternative to the teacher-student framework, with the additional advantage of not having to have identified a fixed teacher a-priori. For this comparison, we implement two action-advice based baselines, presented in Section 2.2.6: one based on experienced agents, and the other based on imperfect agents. To compare *EF-OnTL* against the TL baselines, we report both training performance over episodes and performance achieved at a fixed episode.

O4 − The fourth objective analyses whether and to what degree *EF-OnTL* transfer effectiveness is influenced by Source Selection Criteria (SS) used to identify the source of transfer. When there are multiple agents that could act as a source of transfer there is a need for a process to identify the suitable source of knowledge. This objective

aims to study the impact of selecting the source of transfer based on different criteria, as presented in *EF-OnTL* design chapter in Section 4.5.1. The experiments will be carried out across multiple scenarios in two different environments: Cart-Pole and Multi-Team Predator-Prey (MT-PP). The setup of these experiments will vary the SS while keeping the other criteria fixed.

O5 − The fifth objective analyses whether and to what degree *EF-OnTL* transfer effectiveness is influenced by Transfer Content Selection Criteria (TCS) used to filter the knowledge on target side. While *EF-OnTL* system identifies a common source of transfer, the target agents, pulling batch of experiences from the source, will require different interactions specific to their learning status. To assess the impact of utilising various criteria for filtering incoming knowledge on a target agent side, this objective analyses the impact brought by TCS, as presented in *EF-OnTL* design chapter in Section 4.5.2. Similarly to *O4*, the experiments will be carried out in multiple scenarios across two different environments: Cart-Pole and MT-PP. The setup of these experiments will vary TCS while keeping the other transfer parameters fixed.

O6 − The sixth objective analyses whether and to what extent *EF-OnTL* transfer effectiveness is influenced by the number of interactions selected by a target agent within a single transfer step. While the number of interactions transferred within a single transfer step is not related to the transfer budget B used to filter the incoming knowledge on the target side, this objective studies if there is a correlation between the transfer outcome and the quantity of tuples sampled on a target agent side. The experiments used for this analysis will be carried out in multiple scenarios across two environments: Cart-Pole and MT-PP. The setup of these experiments will vary the budget B while keeping SS and TCS fixed.

O7 − The seventh objective studies the extent of applicability of *EF-OnTL*, based on complexity of the environment. Reinforcement Learning (RL) can be used to learn a policy in various environments of different complexity, i.e., dimensions of the combinatorial state-action space, delayed and sparse rewards, and complex underlying dynamics. The seventh objective aims to identify the capabilities and limitations of *EF-OnTL* in relation to the complexity of different environments. *EF-OnTL*

will be analysed into four different environments of increasing complexity: Cart-Pole, MT-PP, Half Field Offense (HFO), and Ride-Sharing Ride-Requests Simulator (3R2S). These environments are detailed in Section 6.2.

O8 − The eighth objective evaluates the adaptation capability of *EF-OnTL* to environments with heterogeneous dynamics. Note, however, that adaption to different dynamics was not and is not considered explicitly in the design of *EF-OnTL*, any further than the adaptation capabilities of underlying RL algorithm used, i.e., Deep Q-Network (DQN) and Parameterised Action Deep Deterministic Policy Gradient (PA-DDPG). However, in order to assess the boundaries of *EF-OnTL* applicability, and establish directions for future work, we perform evaluation in which we use *EF-OnTL* to transfer knowledge not just to the same environment dynamics as in previous experiments, but to those with different underlying dynamics. Specifically, we analyse various transfer scenarios within the 3R2S environment, enabling transfers across agents specialised in different demand trends.

O9 − The ninth objective evaluates performance comparing to centralised multi-agent learning. While *EF-OnTL* operates in a decentralised manner with reduced communication, making it distinct from multi-agent learning, we still intend to assess them purely in terms of performance. Such an evaluation would be useful, especially in scenarios where either approach might be suitable from an architectural point of view. This evaluation will be conducted through a comparison with Multi-Agent Reinforcement Learning (MARL) baselines, more detail in Section 6.1.1. The experiments will carried out in two different environments: MT-PP and HFO, using two different RL paradigms.

6.1.1 Baselines

To assess *EF-OnTL* impact, this thesis compares *EF-OnTL* against different baselines:

1. *no-transfer* − each agent is equipped with an independent copy of the learning model that is updated only using its own experience;

2. *Online Confidence-Moderated Advice Sharing (OCMAS)* [52] − this baseline is dis-

cussed in detail in Section 2.2.6, but here we repeat its main characteristics. In this approach, each agent owns a copy of the learning process, which is optimised based on the experiences collected by that specific agent. Additionally, agents follow the teacher-student framework to share knowledge and rely on Random Network Distillation (RND) to estimate their confidence level within a state.

At each time step, every agent shares the state that they have visited with the rest of the agents to estimate their uncertainties. When an agent is the most uncertain in its visited state, the transfer process begins. Other agents provide an action-advice to the most uncertain agent. The final action decision is made through majority voting based on the received advice. Transfer is constrained by a budget set for each agent. This budget limits the number of times that an agent follows the received advice. Despite this limitation, agents maintain continuous interaction, sharing their explored states and evaluating their associated uncertainties.

3. *Requesting Confidence-Moderated Policy (RCMP)* [42] − this baseline is discussed in detail in Section 2.2.6, but here we repeat its main characteristics. A target agent is guided by an external group of expert teachers, referred as jury, following the teacher-student framework. The advising strategy is based on comparing the uncertainty level of a target agent against a predefined threshold and the advice shared is in form of action. The communication is restricted by a budget and, in this implementation, the target agent selects the action to take as the most frequent action suggested by the jury.

4. *MARL* − the MARL approaches used in this thesis are the multi-agent adaptation of PA-DDPG [8] and QMIX [59], both described in Section 2.2.7. The selection of the MARL algorithm used is based on the learning paradigm used by *EF-OnTL*. The multi-agent PA-DDPG variant is used in HFO. Here, a centralised critic learns a global action-value function based on the joint actions taken by the agents. In the MT-PP environment QMIX is used. QMIX enables the agents to aggregate their independent Q-functions to influence the learning model of a certain agent based on other agents' actions.

Intuitively, based on its design, *EF-OnTL* is expected to improve over *no-transfer*

method and perform similarly as *OCMAS*. *RCMP* is expected to be an upper limit for *EF-OnTL* due to the availability of pre-trained agents.

6.1.2   Metrics

The evaluation metrics commonly used within the TL context are generally defined on the reward. These metrics assess the impact of transferred knowledge by measuring the difference in rewards between the TL approach and a baseline. In this thesis, we assess the following:

1. asymptotic improvement − it measures the performance gap that there might be on the learning curves of two agents;

2. performance at fixed time − it measures the performance achieved by an agent after a certain number of episodes. Performance are expressed through cumulated reward as well as environment-specific metrics;

3. transfer cost − it measures the number of interactions or action-advice transferred to a single agent;

Although these metrics are widely used to assess the impact of TL, they offer a comprehensive understanding of the training stage of a TL-enabled agent versus an agent with no external support. In *EF-OnTL*, the impact of transferring external knowledge as agent-environment experiences is expected to be delayed when compared to other form of advice, i.e., action-advice. Therefore, in this thesis, where we required, we also introduce additional metrics specific to the environment we are analysing. Finally, to ensure a robust evaluation and minimise random bias, the results are presented as an aggregation of multiple independent simulations collected under different initialisation seeds.

6.2   Evaluation Environments

To assess *EF-OnTL* performance against the baselines, this thesis evaluates the approaches across 4 different environments of increasing complexity: 1) Cart-Pole, 2) MT-PP, 3) HFO and 4) 3R2S with real-world data. This section motivates the environments used in this research and presents their implementation details needed to reproduce the experiments.

6.2.1 Cart-Pole

Cart-Pole is a simple and popular task widely adopted in the field. Due to its simplicity, Cart-Pole is a good option for validating the correctness of RL implementations and contributions.
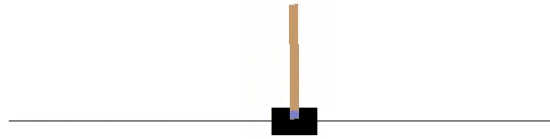
Figure 6.1: Cart-Pole environment [10].

The Cart-Pole problem consists of an unstable pole placed vertically on top of a moving cart [142]. At initialisation time, the pole is balanced but subjected to gravity forces. The goal consists of preventing the pole from falling by moving the cart along the x-axis while keeping it within certain boundaries. Figure 6.1 shows a capture of the Cart-Pole problem used for the experiments this thesis.

Observation consists of 4 values, cart position $[-4.8, +4.8]$, cart velocity $(-inf, +inf)$, pole angle $[-24°, +24°]$ and pole velocity $(-inf, +inf)$. Cart is moved left or right by an agent through 2 discrete actions and receives a positive reward of $+1$ for every step that the pole is balanced, therefore acquiring a reward of 1 in every time step, until the pole collapses and the episode terminates.

The Cart-Pole environment used within this research is based on the implementation by Open-AI Gym [10]. To enable multiple agents visiting a Cart-Pole problem simultaneously, the environments is initialised 5 times in parallel.

As the agents interact one to the other in *OCMAS* and *EF-OnTL*, agents are synchronised at the end of each episode to ensure that transfer happens between agents of similar performance.

6.2.2   Multi-Team Predator-Prey

This version of predator-prey is an advanced configuration built upon the environment introduced in Section 3.2.1. Such a task has been widely used to study multi-agent algorithms and contributions to the field of MARL. A screenshot of the game is available at Figure 6.2. There are a total of 8 predators, shown as filled triangles, and 4 prey, shown as pierced triangles. Predators and prey are evenly distributed in two different colour-based teams.
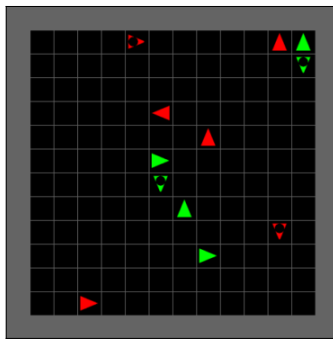
Figure 6.2: Screenshot of Multi-Team Predator-Prey (MT-PP).

The environment dynamics and goal are simple, predator chases prey while prey tries to survive the longest by evading the predator. Both predator and prey can be controlled by an intelligent agent and the number of agents involved can be fully customisable. Furthermore, the grid is fully customisable in size and layout as well as the agents sensors. The results shown in this thesis assume that predators are controlled by RL-powered agents while prey follow a random policy to escape the chasers.

MT-PP consists of two teams that compete to catch prey faster than the other team. At initialisation time, teams are balanced and spread over the grid, thus they have same amount of resources, predators and prey. An episode terminates when a team has no prey left to be captured.

The predator reward model is based on the action taken by the agent and the observed outcome. This is reported in Table 6.1. The actions that change the position or orientation of the predator have an associated reward of -0.01. For the *catch* action there are multiple possible outcomes: when a prey is caught a reward of ± 1 is given to the agent. This reward

is positive when both predator and prey belong to the same team and negative otherwise; On the other hand, when the catch fails, a reward of -0.5 is given to the agent. Finally, a reward of -0.25 is given to an agent for the *hold* action.

Table 6.1: Multi-Team Predator-Prey (MT-PP) reward model.

Parameter	Value
catch same team prey	$+1$
catch opponent team prey	-1
failed catch	-.5
hold	-.25
other actions	-.01

In the configuration used within this research, the grid is composed by 12×12 accessible cells shaped as a obstacle-free square delimited by a wall. Each agent perceives a 3×3 grid centred over next consecutive cell. Thus, the observation is composed of a 3-dimensional 3×3 matrix. First channel describes object type, varying between: void, wall, predator and prey. Second channel identifies the team membership: None, in case of an empty cell or wall, red and green. Third channel indicates the orientation of the perceived object. The orientation of an object is determined by a global orientation system, independent of the observing agent's orientation. It can be categorised as up, down, left, right, or none in the case of an empty or wall cell.

In the experiments reported within this thesis, predators are given precedence to select and take actions on prey. Consequently, in situations where a predator selects *catch* action when facing a prey, the prey becomes unable to escape.

Moreover, the priorities among predators are randomly assigned during initialisation, regardless of team membership, and remain fixed throughout each episode. Consequently, if the actions taken by two predators are conflicting, e.g., by occupying the same empty cell that are both facing from different sides, only the predator that moves first will succeed in the subsequent step, while the other will remain in its original position.

This specific MT-PP setup is extremely interesting as different kind of multi-agent situations and interactions arise. As per Ferber at [54], it is possible to observe competitive interactions, between the two teams as their goals are mutually incompatible, and collaborative interaction within a team, as all the agents that belong to a team aim to accomplish

a common goal.

As evaluation metrics, on top of the reward, this thesis reports the observed probability of winning a match and capturing a prey assigned to the same and opposite team.

6.2.3   Half Field Offense

HFO is a multi-agent environment based on a simplified game of soccer where agents need to collaborate to score a goal. Collaborative interaction further complicates the task that already has sparse reward function, continuous state and parametrised control space. Thus, having a single poor performing agent within the team will likely result in capping the team performance. Task is episodic and an episodes terminates when one of the following events occurs: 1) a goal is scored; 2) ball is out of bounds; 3) a defender or the goalkeeper gets possession of the ball; 4) episode reaches the time limit.
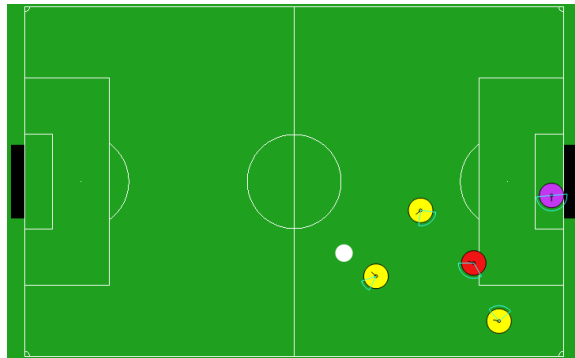
Figure 6.3: Screenshot of HFO game.

The setup used in this work is a 3 vs 2 situation consisting of 3 offense players and 2 for the defense. The defense consist of a goalie and a single defender. Figure 6.3 shows a screenshot of the HFO setup used in this thesis. At the beginning of an episode, all players are initialised in the rightmost half of the pitch. Offense players are positioned in proximity to the midfield line, the defender is placed just outside the penalty box, and the goalie is located between the sticks. Initially, ball control is given to the offense team, while the other team tries to intercept the ball. In this work, offense players are powered by RL agents, while the others are played by HELIOS baseline [73].

Offense players are controlled by RL agents through the mid-level control space [1]. As a

---

[1]There are three levels of control with different complexity available for an agent: high, offering only

result, there are four available actions:

1. $kick\_to(x, y, speed)$ − the player kicks the ball to the specified coordinates and with the provided speed;

2. $move\_to(x, y)$ − the player goes from the current position to the specified point using the maximum running speed;

3. $dribble\_to(x, y)$ − the player dribbles the ball to the specified coordinates. If the player controls the ball it tries to avoid the opponent and tackle otherwise;

4. $intercept$ − based on the ball velocity, the player moves to intercept the ball.

(x, y) coordinates used to specify a target point are normalised within $[-1, +1]$ interval while speed ranges in $[0, 3]$ interval. When selecting an action, an agent must pick the discrete action alongside any relevant continuous parameters necessary to parametrise the selected action.

HFO state provides information on the controlled player, the ball, the team mates and the opponent team players. In the setup used in this thesis, the observation for each agent consists of 95 features. For readability, HFO state is detailed in Appendix at Sec. A.

The performance of players is assessed by tracking the number of goals they score within a set range of episodes. When a goal is scored, the environment returns a positive reward $(+1)$ to all players on the scoring team, regardless of any individual contribution to the goal.

The high quantity of observed features and the continuous control space make the HFO environment an interesting benchmark for assessing TL frameworks, in particular *EF-OnTL*. The Neural Network (NN) underlying the learning process may gain significant advantages from the transferred experiences in this context.

6.2.4 Multi-brain Ride-Sharing Ride-Requests Simulator

The multi-brain version of 3R2S is built upon the environment introduced in Section 3.2.2.

While Manhattan is geographically concentrated in a limited area and could potentially be managed by a single policy replicated across all vehicles, in our environment setup, we

---

discrete actions; low, where actions are defined on continuous parameters, and mid, offering intermediate control complexity between the two.

divided the zone into different sub-areas to investigate the impact of transferring across multiple policies, each tailored to an unique non-overlapping zone. Each policy is trained on a distinct subset of data, making it interesting to observe the transfer learning effects induced by *EF-OnTL*.

The multi-brain implementation of 3R2S consists of four brains, each trained on an independent and non-overlapping sub-area of Manhattan. The division of these zones is established by clustering the origins and destinations of ride-requests available during the morning peak hours from $7 - 10$am. Cluster centroids are identified using K-means [143], with the Manhattan distance is employed as the distance measure for estimating cluster membership.
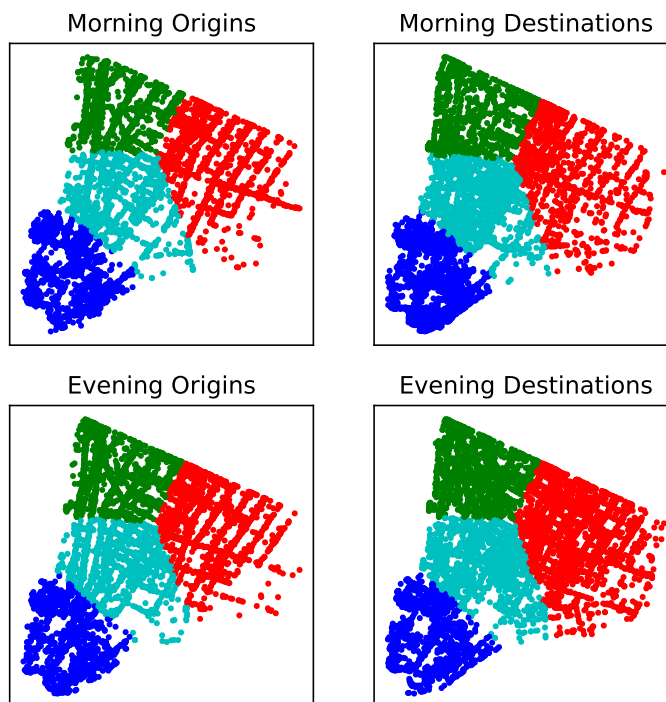
Figure 6.4: Ride-requests membership over the 4 clusters identified by K-means [143].

The clustering of requests origin and destination shown in Figure 6.4 demonstrates the distribution of membership across the four clusters during both morning ($7 - 10$am) and evening ($6 - 9$pm) demand sets. Although this clustering approach is designed to achieve a balanced division of zones and mitigate demand imbalances, each brain will have different

opportunities to collect training data due to the different area covered by each brain.

When an agent needs to take an action, it employs the policy from the respective brain determined by hard cluster membership based on the current vehicle's position. When enabling online TL across agents, despite the state-domain space visited by each agent will be unique to that agent, each brain will receive part of knowledge from the other zones.

6.3   *EF-OnTL* Evaluation

This section presents *EF-OnTL* evaluation experiments across the four environments to validate whether the promising results on experience sharing in an offline context, shown in Chapter 3, are maintained when transferring online across learning agents.

Firstly, Section 6.3.1 compares *EF-OnTL* against the identified baselines to address the evaluation objectives *O1*, *O2*, *O3*, *O7* and *O9*. The baselines used are *no-transfer*, *RCMP*, *OCMAS*, and Cooperative Centralised Learning (CCL). These evaluation objectives appraise the positive transfer impact of *EF-OnTL* and compare the improvements against the TL baselines and MARL CCL to assess whether *EF-OnTL* could be a more advantageous choice for enhancing agents' performance in a multi-agent context. To provide a robust evaluation for these studies three scenarios of different complexity are utilised: Cart-Pole, MT-PP, and HFO.

Secondly, Section 6.3.2 presents the *EF-OnTL* studies to address the evaluation objectives *O4*, *O5* and *O6*, presented in Section 6.1. These objectives aim to study the impact of various transfer parameters in *EF-OnTL* such as the Source Selection Criteria (SS), introduced in Section 4.5.1, the TCS, introduced in Section 4.5.2, and the transfer budget. The benchmark scenarios used for these studies are the Cart-Pole and MT-PP benchmark environments, presented respectively in Section 6.2.1 and Section 6.2.2.

Thirdly, Section 6.3.3 reports the experiments in the 3R2S environment with real-world demand pattern. These experiments evaluate the evaluation objective *O8* to appraise the generalisation ability of *EF-OnTL* powered agents across different scenarios.

6.3.1 *EF-OnTL* and Baselines Evaluation

This section answers **RQ1**, identifying if and to what extent *EF-OnTL* improves the system performance. To provide a better understanding and to position *EF-OnTL* performance, this section uses the evaluating objectives *O1* and *O2*, by comparing the performance achieved by *EF-OnTL* against a *no-transfer* baseline, *O3*, by comparing *EF-OnTL* against action-advice based teacher-student baselines, *OCMAS* and *RCMP*, and *O9*, by comparing the *EF-OnTL* performance against the multi-agent baselines, QMIX and Multi-Agent Deep Deterministic Policy Gradient [61] (MADDPG). Furthermore, this section analyses the generalisation capability of *EF-OnTL* by investigating its TL effectiveness across multiple environments of varying complexities, aiming to evaluate the objective *O7*.

EF-OnTL and the baselines are tested in four different benchmark environments: Cart-Pole, discussed in Section 6.2.1, MT-PP, discussed in Section 6.2.2, HFO, discussed in Section 6.2.3, and 3R2S, discussed in Section 6.2.4.

Table 6.2: Summary table of the RL approaches implemented in each benchmark scenario.

Environment	*EF-OnTL*	*no-transfer*	*OCMAS*	*RCMP*	*QMIX*	*MADDPG*
Cart-Pole	✓	✓	✓	✓	✗	✗
MT-PP	✓	✓	✓	✓	✓	✗
HFO	✓	✓	✗	✓	✗	✓
3R2S	✓	✓	✓	✓	✗	✗

Table 6.2 reports the methods employed in each benchmark environment. Due to specific limitations of certain environments, some of the baselines were not applicable. We further discuss this matter later in the subsequent sections. Table 6.3 provides a summary of the transfer settings used to evaluate *EF-OnTL*. For the 3R2S, epochs and episodes are separated by a period.

In the subsequent sections, we analyse the impact of *EF-OnTL* versus the baselines in each benchmark environment.

6.3.1.1 Cart-Pole

In the Cart-Pole benchmark environment, as introduced in Section 6.2.1, a total of 5 agents are concurrently trained. Every agents start a new episode at a synchronised timestep,

Table 6.3: *EF-OnTL* parameters on benchmark environments.

Parameter	Cart-Pole	MT-PP	HFO	3R2S
N	5	4	3	4
TF	200	300	400	5
TB Capacity	10, 000	100, 000	25, 000	10, 000
SS	$\overline{U}$	$\overline{U}$	$\overline{U}$	$\overline{U}$
TCS	*high* Δ-*conf*	*high* Δ-*conf*	*high* Δ-*conf*	*high* Δ-*conf*
Budget B	5, 000	500	100	100
Ep. Start Transfer	600	2, 500	2, 400	1.5
Max Timestep	400	200	500	n.a.
Max Episode	1, 800	8, 000	20, 000	9.9

in order to facilitate the inter-agent transfer. The sharing of experience is enabled from episode 600 and a batch of experiences is shared every 200 episodes thereafter. The transfer of knowledge is postponed until the 600*th* training episode as, in the initial stage, agents lack sufficient knowledge to estimate their epistemic confidence. Furthermore, transferring knowledge in the first episodes would add little value to the target policies.

Evaluation in the Cart-Pole benchmark environment analyses the cumulative reward achieved by agents across training episodes. Figure 6.5 shows episode count on the x-axis and the cumulative episode reward on the y-axis. In this benchmark environment, the multi-agent baseline is omitted as each agent interacts exclusively with its copy of the environment and therefore, there is no context of joint actions.

In Figure 6.5, *EF-OnTL* demonstrates improved performance over the *no-transfer* baselines starting from episode 1, 400. Prior to that episode, the learning curves of *EF-OnTL* agents and the *no-transfer* approach overlap, achieving, on average, similar performance level. The *RCMP* achieves very high levels of reward during the initial episodes, from 0 to 100. The increased reward is given by the lack of confidence on the target agents that lead to a receiving continuous guidance thorough advice. However, following the initial spike, the learning curves exhibits a declining trend until episode 600. Thereafter, the increased confidence of the target agents hinders advising, resulting in stationary performance with limited improvements. Lastly, *OCMAS* shows a noticeable performance gap compared to *no-transfer* between episodes 1, 100 and 1, 400, ultimately matching the performance by the conclusion of the training episodes.
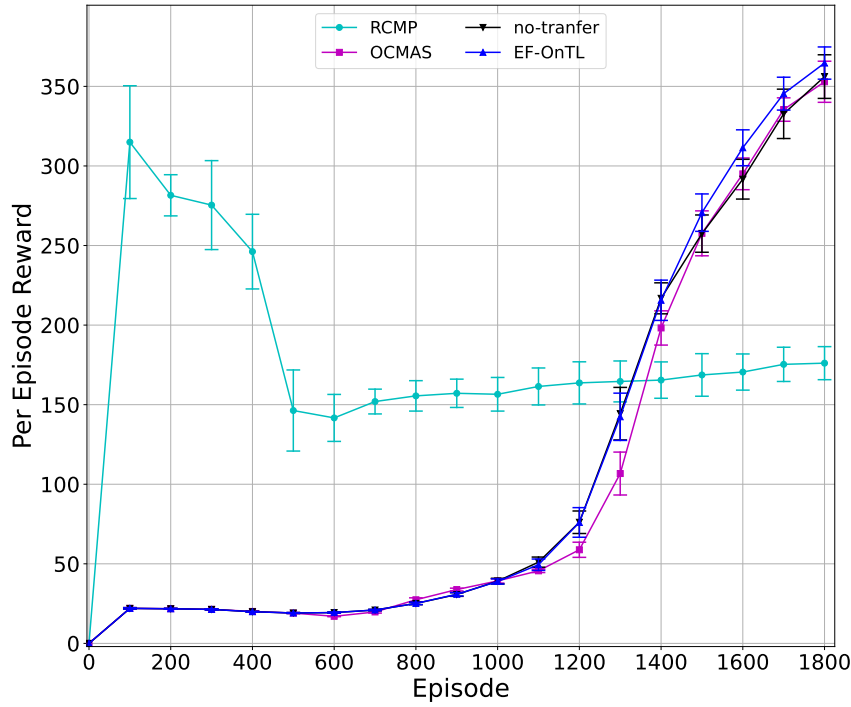
Figure 6.5: *EF-OnTL* (transfer based on Higher Delta Confidence (*high Δ-conf*), budget: $5,000$ and $SS : \overline{U}$) compared against *RCMP*, *OCMAS* and *no-transfer* in Cart-Pole.

6.3.1.2   Multi-Team Predator-Prey

The second environment used to benchmark *EF-OnTL* against the baselines is MT-PP, detailed in Section 6.2.2. In MT-PP four agents, each controlling a predator within the same team, are concurrently trained in a multi-agent scenario. Similarly as in Cart-Pole, the sharing of experience is delayed until the $2,500$th episode and transfer happens every 300 episodes thereafter until episode $7,500$.

Since agents are concurrently trained to achieve their team goals, in this environment we also compare *EF-OnTL* against QMIX, a MARL baseline.

To evaluate the performance of the implemented methods in MT-PP we first report the learning curves in Figure 6.6. The x-axis denotes the episode counter from 0 to $8,000$ and the y-axis denotes the average reward scored by agents. The last 500 episodes are used to test the methods as the learnt policy is sampled in a deterministic way.

Both *OCMAS* and *RCMP* show considerable positive transfer within the first initial episodes where transfer is enabled. Specifically, *OCMAS* demonstrates a noticeable performance improvement between episodes $2,500$ and $2,600$ upon enabling transfer. Similarly, *RCMP*,
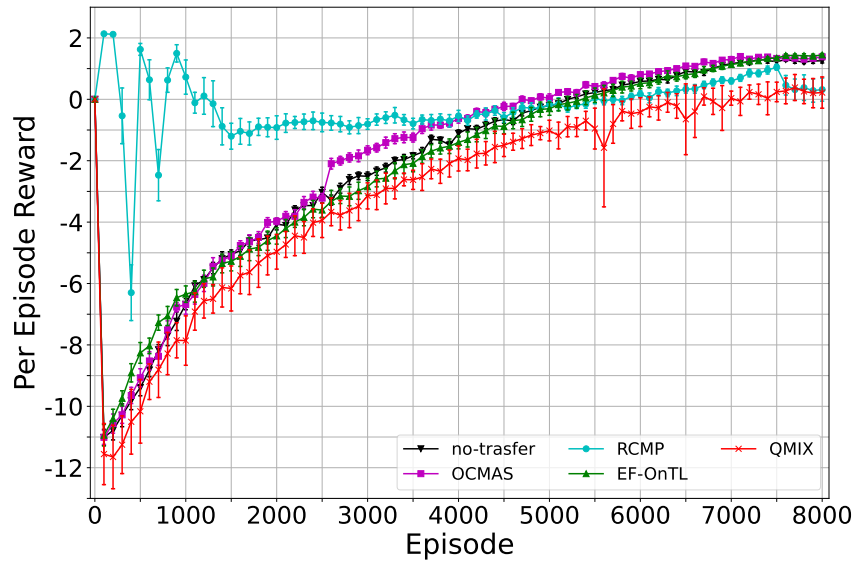
Figure 6.6: Learning curves of *EF-OnTL* (*high Δ-conf*, budget: 500 and $SS : \overline{U}$) compared against *no-transfer*, *OCMAS*, *RCMP* and *QMIX* in MT-PP.

thanks to the expert teacher, achieves positive rewards in the first 200 episodes, while others techniques experience severe negative rewards due to random exploration. Although, *RCMP* trend resembles what observed in Cart-Pole, where, after a remarkable jump-start, the learning curves assumes a flat trend. In contrast, *EF-OnTL* does not display an immediate improvement due to the enabling of transfer; instead, the effect of transfer is distributed over time and eventually it demonstrates a trend similar to that of *OCMAS*.

Among all the techniques evaluated, QMIX records the lowest level of reward. Such lower performance is a consequence of the training setup, where the two teams that are benchmarked against each other are trained in a competitive scenario.

In the QMIX multi-agent learning framework, agents experience a delay in the learning phase due to the complexities involved in learning the joint state-action value function. Therefore, the opponent teams, composed by independent agents, who do not rely on joint state-action value function, learn to adapt their strategies more quickly. As a result, the opportunity for the QMIX team to develop an effective policy is lower.

While QMIX certainly encourages collaboration among agents, such joint learning can be a drawback in competitive scenarios. In contrast, *EF-OnTL* decouples the learning and transfer phases. In the evaluation carried out on MT-PP, *EF-OnTL* leads to better results. This is because the external knowledge, with the exception of cases involving negative

transfer, does not delay an agent's learning process.

The MT-PP environment provides additional metrics that enhance the comprehension of the ongoing dynamics. In addition to the reward, this environment offers evaluation metrics such as the average number of successful prey caught in an episode (Avg. Catch) and the win probability (Win [%]). The performance against these metrics are shown in Figure 6.7, providing a comprehensive perspective over 500 test episodes spanning from episode $7,500$ to $8,000$. The figure presents box plots for each assessed technique and is divided into three sections. The leftmost section shows the average reward achieved by agent in an episode, the middle section shows the average number of prey caught, and the rightmost section shows the probability of winning an episode.
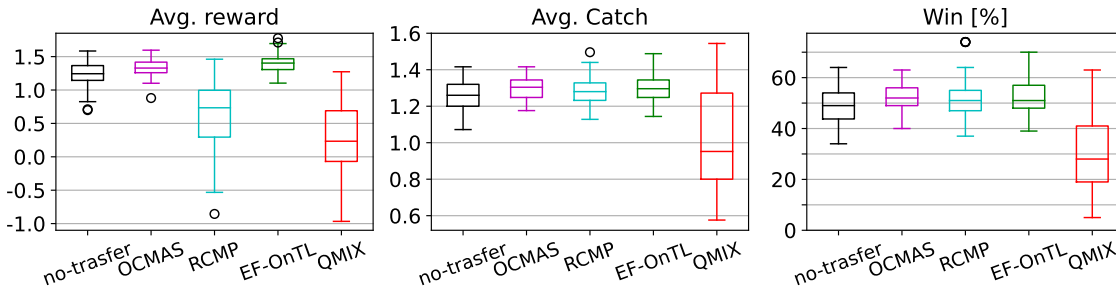
Figure 6.7: *EF-OnTL* and baselines evaluation metrics on 500 test episodes in MT-PP

Both *OCMAS* and *EF-OnTL* show an improvement over *no-transfer* method across all the three metrics. On the contrary, *RCMP* agents score a lower average reward in the test episodes while having a winning probability and an average catch aligned with the other two TL techniques. The negative reward is given by the different strategy learnt by *RCMP* agents. In fact, while *EF-OnTL* and *OCMAS* agents learnt to pursue prey across the grid, *RCMP* agents prioritised the catch action which leads to higher cost when not successful, and sometimes results in incorrectly catching other team's prey. Finally, these results confirm that QMIX agents are penalised, in this benchmark environment, by the additional step to evaluate the joint actions.

6.3.1.3   Half Field Offense

The third environment used to benchmark *EF-OnTL* against the baselines is HFO, detailed in Section 6.2.3. In HFO the offense players, powered by RL agents, are those whose per-

formance we are evaluating. The defense player and the goalkeeper are played by *HELIOS* baseline [73].

In HFO, *EF-OnTL* target agents are enabled to receive 100 selected interactions from episode $2,400$ to episode $20,000$. Transfer occurs every 400 episodes and the interactions on target side are filtered by *high* Δ-*conf* while the source of transfer is selected by Average Uncertainty ($\overline{U}$).

All the RL agents are based on the PA-DDPG model presented in Section 5.2.2. *EF-OnTL* is compared against a *no-transfer* baseline, where agents act independently, MADDPG, and the expert action-advice based baseline *RCMP*. The TL method *OCMAS* is not implemented due to the client-server architecture of HFO, which delays the training of the agents due to continuous communication to establish whether to provide an advice or not. *OCMAS*, fully discussed in Section 2.2.6, requires each agent to forward the visited state to all the other agents. As a consequence, the maximum number of intra-agent communications, calculated as the product of the number of agents (3), the number of episodes ($20,000$), and the maximum timestep (500), totals to $30,000,000$.

To evaluate the performance of the offense team, which is controlled by RL agents, this thesis assesses the goal probability recorded within a specific number of episodes. The probability is determined by counting the number of goals scored in a fixed interval. Whenever the offense team scores a goal, the episode terminates. During the learning phase, the goal probability is estimated using the most recent 100 episodes. On the other hand, the goal probability during evaluation is estimated over a span of 500 episodes, during which players take deterministic actions and no transfer is allowed. The deterministic evaluation is conducted every 500 episodes, where agents leverage their latest knowledge.

Figure 6.8 reports the results. The x-axis denotes the training step and the y-axis denotes the goal probability, in the range of $[0, 1]$. During evaluation, the training step of the x-axis indicates the version of the policy used by each agent. The results are organised into two graphs, the upper row reports the goal probability during training, while the lower row shows the goal probability on the test set.

Similarly to what we observed previously for the other benchmarks, *EF-OnTL* successfully improves the performance of the offense team compared to the *no-transfer* method.
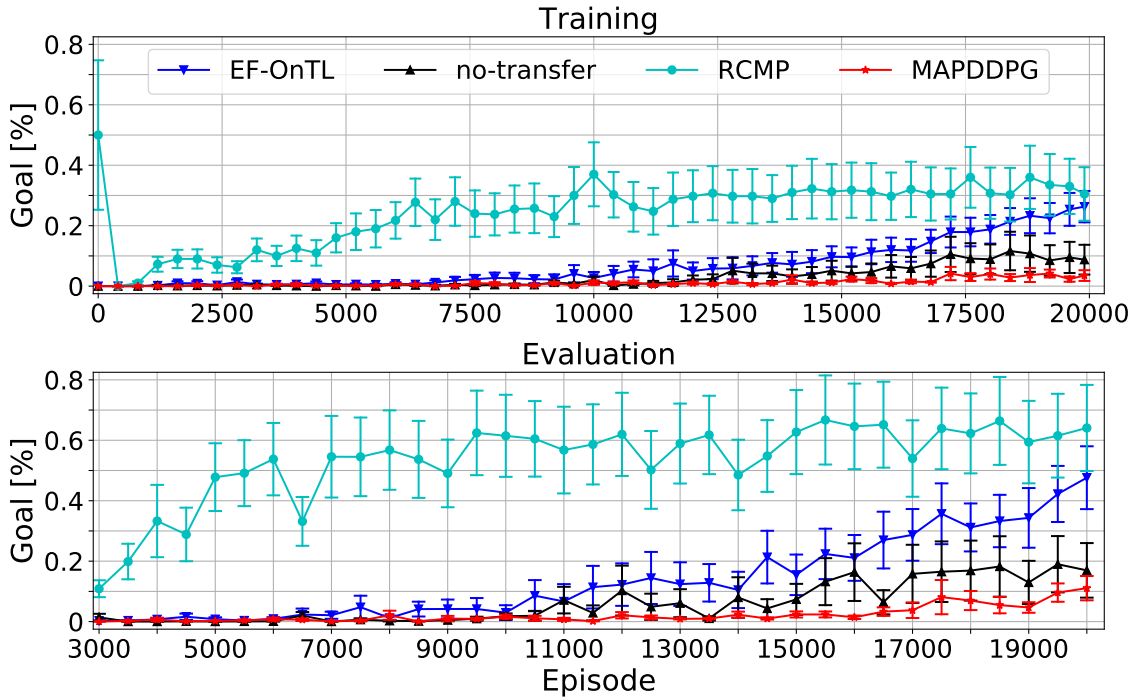
Figure 6.8: *EF-OnTL*($high\ \Delta\text{-}conf$, budget: 100 and $SS : \overline{U}$) compared against *no-transfer*, *RCMP* and multi-agent PA-DDPG in HFO.

Furthermore, the positive effect of *EF-OnTL* becomes noticeable during the second-half of the training phase. Sharing a restrained number of selected experiences across agents driven by *EF-OnTL*, it enables the offense team to achieve up to 2.5 times higher scores compared to *no-transfer* agents. In particular, from episode $14,000$, the goal probability in training and testing for *EF-OnTL* diverges from *no-transfer* method and MADDPG, which requires nearly the double amount of episodes to achieve comparable performance.

In HFO the exploitation of expert knowledge plays a crucial role and *RCMP* achieves outstanding results since an early stage of training. In training, during the first 500 episodes, the target agents receive substantial advice being fully driven by the experts. The advice has a positive impact even on evaluation as agents powered by *RCMP* have a goal probability of 50% from episode $7,000$ onward. Certainly, the exploitation of external expertise is unmatchable by *EF-OnTL* method with the transfer setting used within this study.

Figure 6.9 reports the average budget used by *RCMP* powered agents, with a threshold set to 10^{-4}, along their median uncertainties registered throughout the training phase.

The ensemble architecture used to approximate uncertainty for PA-DDPG is on the critic
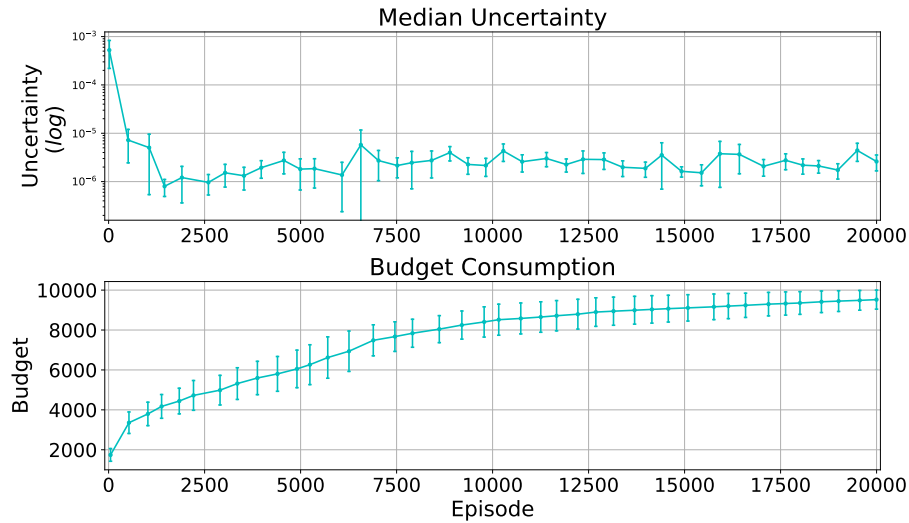
Figure 6.9: *RCMP* analysis in HFO with uncertainty trend and budget consumption.

network which outputs a single value and as such, it is impossible to predict a range and hence to normalise the estimated uncertainty as done previously in *Cart-Pole* and MT-PP. In fact, the *RCMP* uncertainty curve decreases sharply within the first hundreds episodes and then ranges within a narrow interval making it not trivial to find the right threshold given the tight median uncertainty interval. In fact, around 60% of times, the budget is used up to $10,000$, by the end of the simulation.

To conclude, even though *RCMP* demonstrates remarkable performance from the early stages of training, by the end of the $20,000$ episode, the performance gap between *RCMP* and *EF-OnTL* is drastically reduced. The growing trend of *EF-OnTL* suggests that eventually it could potentially catch up to the performance of *RCMP*. However, it is evident that when an expert is available, *RCMP* remains the preferable choice. On the other hand, *EF-OnTL* remains a viable alternative in scenarios where an expert is not available up front.

6.3.1.4 Ride-Sharing Ride-Requests Simulator

The fourth and last environment used to benchmark *EF-OnTL* against the baselines is 3R2S with the multiple brains configuration, detailed in Section 6.2.4. In this version of the environment, the four RL-agents control a non-overlapping zone of the Manhattan area. Consequently, each vehicle uses the policy provided by the brain of the zone where it is located to select the appropriate action.

In the 3R2S environment, as detailed in Section 3.2.2, the training consists of 10 epochs, with each epoch composed by 10 episodes. In *EF-OnTL*, the sharing occurs in total 17 times. In each transfer step, target agents filter up to 100 selected interactions. The transfer is delayed until the beginning of the second epoch and it is enabled every 5 episodes thereafter.

Although within the environment there are multiple agents that interact with the same copy of the environment and learns concurrently, no MARL scenario has been evaluated in this setup as there is no direct collaboration between the agents. However, we further examine different settings of the environment in Section 6.3.3.

To assess the performance of the 200-vehicle fleet, we analyse several commonly used measures adopted by related work [18, 86, 144–146]. These include the distribution of waiting times for passengers before they board a vehicle (*waiting time*), the total number of successfully served requests (*satisfied requests*), and specific vehicle-related metrics. These vehicle metrics report the distribution of passengers across the fleet (*passengers distribution*), the cumulative distance travelled by the vehicles in kilometres (*mileage*), and the proportion of distance travelled by vehicles while unoccupied (*% distance travelled empty*) or engaged in ride-sharing (*% distance travelled in RS*).

The performance of *EF-OnTL* is compared against both *no-transfer* and the TL baselines, *OCMAS* and *RCMP*. The outcomes across the evaluated metrics are visualised using box-plots, as presented in Figure 6.10.

Enabling TL in the multi-brain 3R2S environment results in a slight decrease in the total number of satisfied requests by the fleet compared to the *no-transfer* approach, although, the difference in the number of satisfied requests is negligible. The *EF-OnTL* approach appears to be more consistent across different simulations in the total number of served request. While *EF-OnTL* exhibits comparable waiting times and passenger distribution to the *no-transfer* approach, the mileage recorded by vehicles is slightly reduced. This difference is likely attributable to the lower number of satisfied requests, which could suggest a reduced inclination towards ride-sharing. This affirmation is further supported by the increased distance travelled by the vehicles while being empty.

Based on the observed results from the evaluated TL-enabled techniques, it can be concluded that TL induces to negative transfer in this benchmark environment. However,
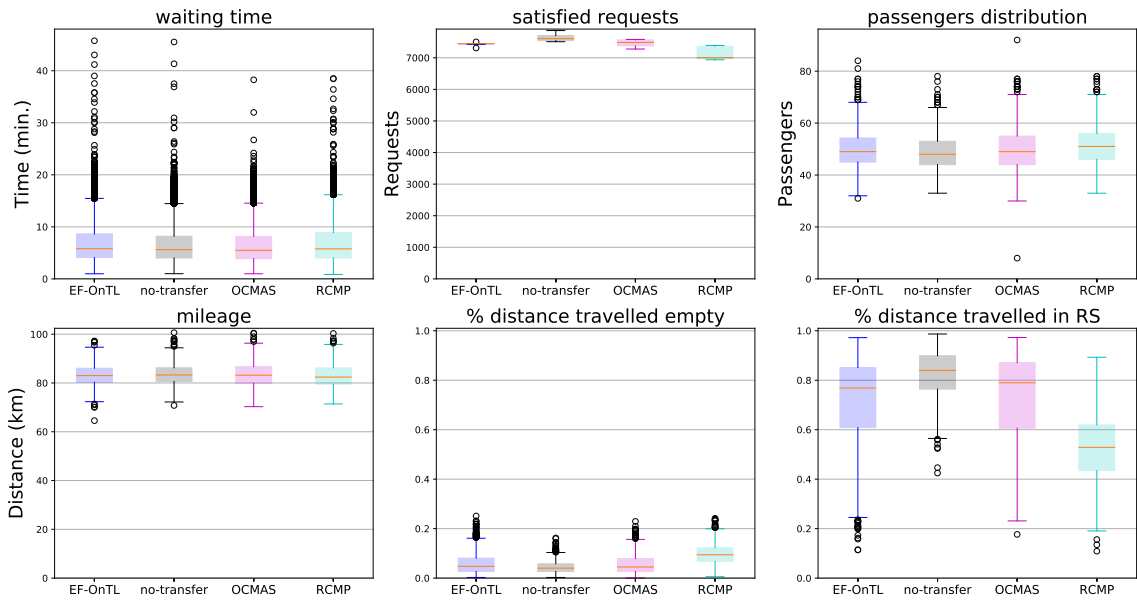
Figure 6.10: *EF-OnTL* compared versus *no-transfer*, *OCMAS* and *RCMP* in the multi-brains 3R2S benchmark.

EF-OnTL has a lower negative impact due to the transfer of experiences compared to the other TL baselines. Interestingly, both *EF-OnTL* and *OCMAS* exhibit a similar performance profile in this study. In contrast, despite relying on expert agents trained within the same zone, *RCMP* exhibited the most pronounced negative transfer effect. However, it is important to note that the difference in performance between these approaches is relatively moderate, and this evaluation did not consider factors like convergence time or the time required to learn an effective policy.

6.3.1.5 Conclusion

This section presented an evaluation of *EF-OnTL* against the identified baselines in four different environments: Cart-Pole, Multi-Team Predator-Prey (MT-PP), Half Field Offense (HFO) and, the multi-brain implementation of 3R2S.

Based on the evaluation objectives *O1* and *O2*, the presented evaluation shows the efficacy of *EF-OnTL* in contrast to *no-transfer*, where agents do not share any information. Specifically, in the Cart-Pole and HFO environments, *EF-OnTL* displays a quicker convergence time, with the improvement being milder in Cart-Pole and more pronounced in HFO. Additionally, *EF-OnTL* agents exhibit superior performance in HFO and MT-PP. While

the disparity in HFO is considerable, the enhancement is less pronounced in MT-PP.

With regard to the evaluation objective *O3*, *EF-OnTL* has shown comparable transfer effect to those of *OCMAS* across the Cart-Pole, MT-PP, and multi-brains 3R2S benchmark environments. However, *OCMAS* requires ongoing communication between agents to assess their confidence in visited states, such process is essential for determining whether an advice is necessary for a particular agent. This requirement makes *OCMAS* inapplicable to the HFO environment.

Notably, in the Cart-Pole, MT-PP, and multi-brains 3R2S scenarios, *EF-OnTL* eventually surpasses the expert-based TL baseline, *RCMP*. In the case of HFO, *EF-OnTL* exhibits an upward trajectory, suggesting the potential to eventually match the performance of *RCMP*.

Among the implemented TL techniques, *RCMP* demonstrates a significant jump-start in performance due to the use of expert teachers. However, as time progresses, the influence of the expert teacher tends to restrict the performance potential of the target agents. This phenomenon is particularly evident in the Cart-Pole and MT-PP environments, where the rewards curve shows an initial surge but then plateau over time. In the Cart-Pole, MT-PP and multi-brains 3R2S environments, *EF-OnTL* has eventually outperformed the expert-based TL baseline, *RCMP*. In the case of HFO, *EF-OnTL* has demonstrated an uprising trend, suggesting the potential to eventually match the performance of *RCMP*.

The low performance of *RCMP* has to be attributed to the threshold used for receiving advice on a target agent. Interestingly, *RCMP* demonstrated exceptional performance in the HFO environments, where the uncertainty estimator model is defined on the critic. In contrast, when the ensemble used to approximate uncertainty is based on the network used for action selection, the uncertainty seems to be limited to certain fixed values, especially for deterministic actions. An in-depth analysis on the threshold sensitivity of *RCMP* is reported in the Appendix in Section C.

In the MT-PP and HFO environments, a MARL algorithm has been implemented for the evaluation objective *O9* and compare its performance against *EF-OnTL*. Specifically, the joint state-action value is learnt through QMIX in the MT-PP environment and through MADDPG in the HFO environment. Generally, MARL has shown lower perform-

ance across the team due to the learning of joint action values, which introduces a delay. As a result, in the HFO environment, MADDPG requires approximately twice the number of episodes to achieve a similar performance as *EF-OnTL*.

To conclude, regarding objective *O7*, *EF-OnTL* has proven successful in three of the four evaluated environments, achieving a positive transfer outcome. We have observed a trend in the improvement given by the complexity of the task undertaken. However, in the 3R2S environment, both *EF-OnTL* and the other TL baselines have resulted in negative transfer.

6.3.2 *EF-OnTL* - Transfer Criteria Evaluation

The studies presented in this section address the research questions **RQ2** and **RQ3**, introduced earlier in Section 1.2. Therefore, this evaluation aims to empirically demonstrate how different selection criteria may impact the performance on *EF-OnTL* algorithm.

In the following study a total of 18 *EF-OnTL* settings are compared by varying different transfer parameters within the algorithm:

- *Source Selection Criteria (SS)* − to answer **RQ2**, which relates to finding a valid strategy for selecting a suitable agent to be used as a source of transfer, this section shows experiments to evaluate the evaluation objective *O4*. SS is presented in Section 4.5.1 and can vary between *Best Performance (BP)* and $\overline{U}$. BP selects the agent that achieved better performance, measured by higher average reward, within the latest episodes, while $\overline{U}$ selects the agent with lower average uncertainty within its transfer buffer.

- *Transfer Content Selection Criteria (TCS)* − to answer **RQ3**, which addresses the establishing criteria that should be used by an agent to filter the incoming knowledge to improve its performance, this section shows experiments to evaluate *O5*. TCS is presented in Section 4.5.2 and can vary between transfer Randomly based on threshold determined by Delta Confidence (*rnd Δ-conf*), transfer based on Higher Loss and Confidence combined (*loss & conf*) and *high Δ-conf*. *rnd Δ-conf* and *high Δ-conf* use only the epistemic uncertainty as filtering criteria while *loss & conf* takes into account the surprising effect of a tuple on the target side.

- *Transfer Size* − to evaluate *O6*, the evaluation focuses on determining whether and to what extent the quantity of interactions transferred through *EF-OnTL* impacts the performance of the system.

The combinations of the aforementioned parameters results into 18 different *EF-OnTL* configurations. These 18 *EF-OnTL* scenarios are tested across two benchmark environments, Cart-Pole and MT-PP. For each setting of *EF-OnTL*, a total of 20 independent runs are collected to obtain a robust evaluation. In the subsequent sections, we first analyse the impact of the transfer parameters in the Cart-Pole environment and then in the MT-PP environment.

6.3.2.1 Cart-Pole

The evaluation in the Cart-Pole environment analyses the cumulative reward achieved by agents across training episodes. Figure 6.11 shows the episode count on the x-axis and the cumulative episode reward on the y-axis. To facilitate the evaluation of *O4*, the curves are organised into two different graphs based on the SS employed within *EF-OnTL* to select the source of transfer. The graph in the top row illustrates *EF-OnTL* scenarios where the source of transfer is selected using $\overline{U}$.Conversely, the graph in the lower row shows scenarios using BP for selecting the source of transfer.

To provide a comprehensive overview and to facilitate the evaluation of *O4*, *O5*, and *O6*, the results from the Cart-Pole benchmark environment, reported in Figure 6.11, are presented also in a further comparative manner. This includes analysing the curves according to the TCS used to filter the incoming experiences on a target side, as depicted in Figure 6.12, and analysing the curves according to the employed transfer budget B, as shown in Figure 6.13.

Across all 18 *EF-OnTL* scenarios, the observed results demonstrate agents achieving comparable performance irrespective of the transfer settings. Although the learning curves exhibit substantial similarity, slight differences emerge from episode $1,200$ onward, by which point agents have already learnt to balance the pole for a few consecutive steps.

Firstly, in Figure 6.12a, the impact of selecting incoming knowledge by *rnd Δ-conf* demonstrates a consistent influence on the target policy regardless of the chosen SS and B.
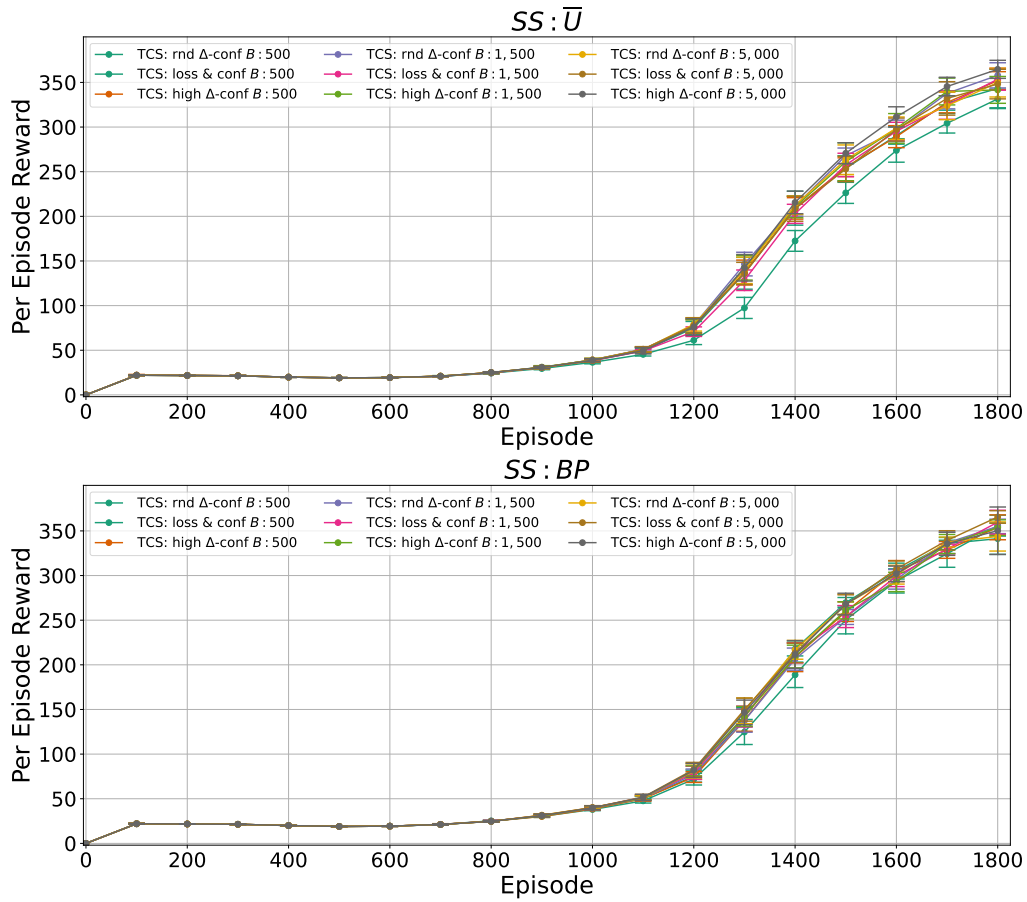
Figure 6.11: *EF-OnTL* learning curves in Cart-Pole with different parameters. In the top chart the source of transfer is selected by $\overline{U}$, the agent with lower average uncertainty within the transfer buffer while, in the bottom one the source is selected by BP, the performance achieved by the agent within the latest episodes.

In fact, *rnd Δ-conf* is the only TCS method that does not benefit from an increased budget of $5,000$. Conversely, *EF-OnTL* agents exhibit superior performance with *high Δ-conf*, as depicted in Figure 6.12b, and *loss & conf*, as seen in Figure 6.12c, when B set to $5,000$.

Despite the results suggesting that filtering for $5,000$ agent-environment interactions on the target side in each transfer step leads to an overall performance improvement, an intriguing finding emerges when the lowest budget of 500 is utilised. Specifically, the *high Δ-conf* approach demonstrates lower volatility in returns when considering both sources of transfer techniques, BP and $\overline{U}$, as depicted in Figure 6.13a. This quality is highlighted in contrast to *loss & conf*, even though the leading scenario within the most recent 500 episodes fluctuates between the two methods.

In conclusion, none of the 18 scenarios prevails significantly over the others. Among

(a) *rnd Δ-conf*

(b) *high Δ-conf*

(c) *loss & conf*
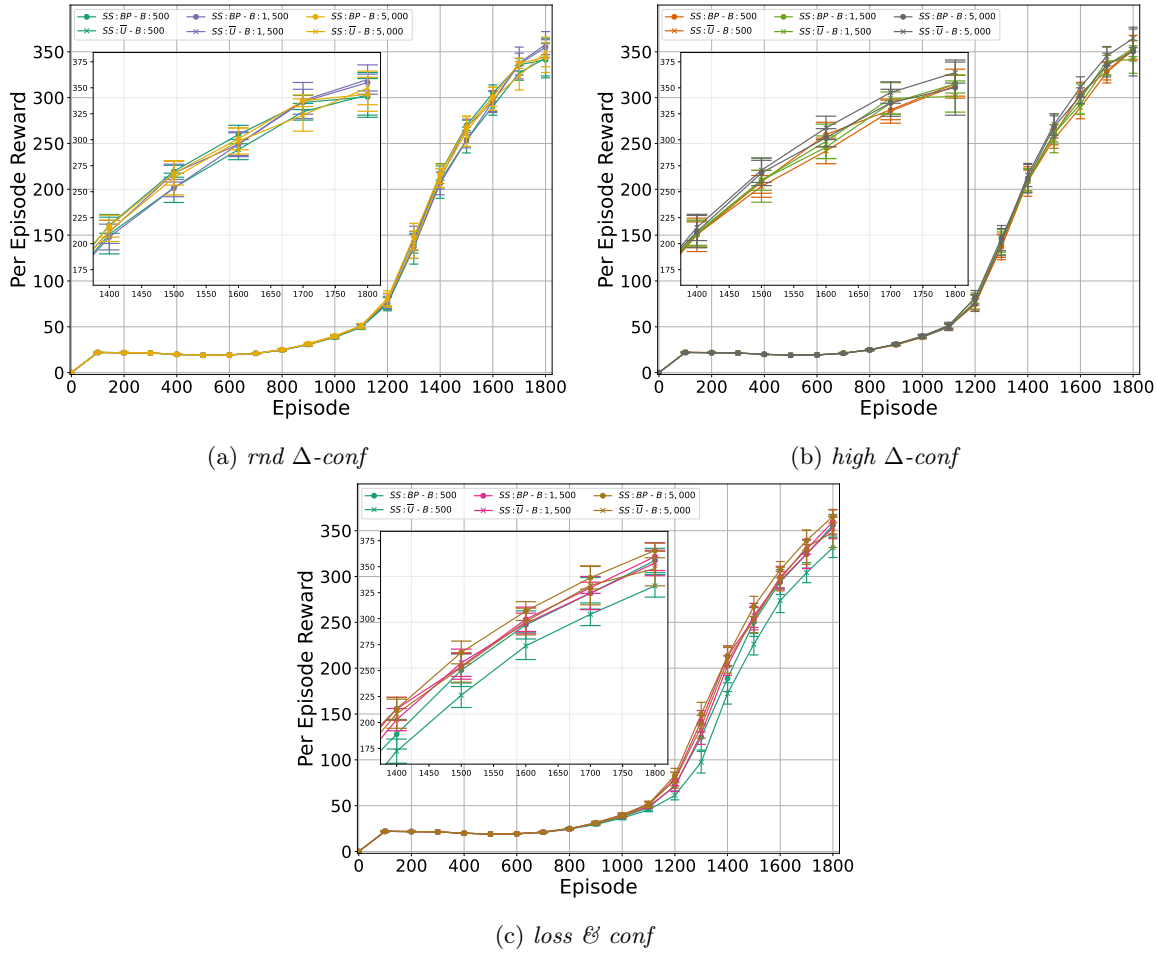
Figure 6.12: *EF-OnTL* learning curves organised by TCS used by a target agent to filter incoming knowledge in Cart-Pole.

these scenarios, *rnd Δ-conf* generally exhibit the mildest impact on the overall system performance. While *high Δ-conf* and *loss & conf* demonstrate comparable performance using different source selection techniques, the *EF-OnTL* configuration that emerges as the preferable choice is the SS: $\overline{U}$, B: $5,000$, and selection of tuples via *high Δ-conf*. This particular configuration consistently yields higher rewards from episode $1,400$ onward, as illustrated in Figure 6.13c.

6.3.2.2 Multi-Team Predator-Prey (MT-PP)

The evaluation in the MT-PP environment analyses both the cumulative reward achieved by agents across the training episodes as well as additional metrics specific to the environment. The overall reward trend observed in MT-PP is similar to what observed in the Cart-Pole

(a) $B = 500$

(b) $B = 1,500$

(c) $B = 5000$
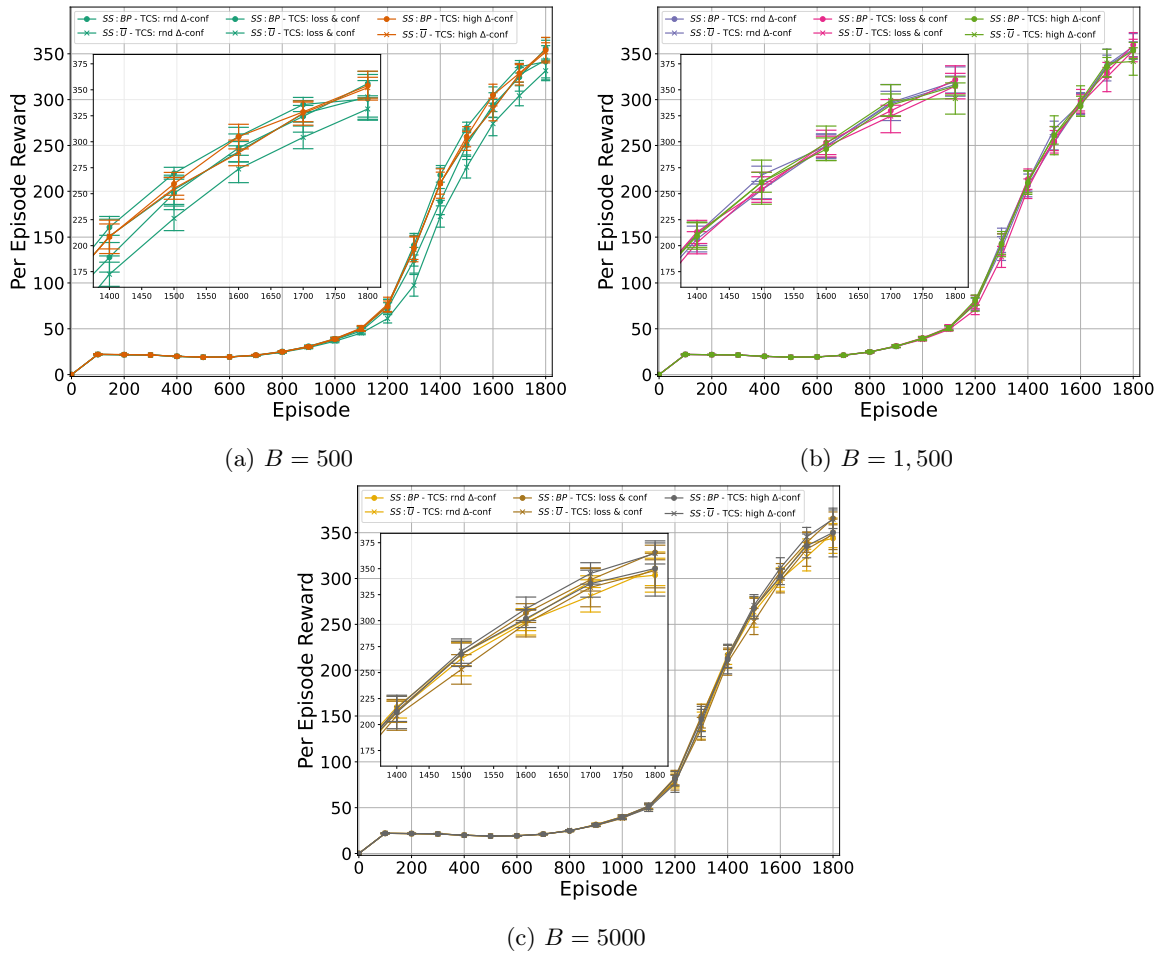
Figure 6.13: *EF-OnTL* learning curves organised by the transfer budget B in Cart-Pole.

environment, where despite the different transfer settings all configurations followed a similar trajectory. Therefore, to maintain the flow of the section, we present the results once, based on the SS used for selecting the source of transfer. Additional graphs are provided in the Appendix at Section B. Figure 6.14 displays the learning curves, the x-axis denotes the episode count and the y-axis denotes the cumulated reward. These learning curves overlap extensively, making it challenging to discern significant differences.

On top of reward, MT-PP offers additional metrics that help the comprehension of the learnt behaviour. These metrics, number of successful prey caught and the win probability, are shown in Figure 6.15, which presents a comprehensive view of 500 test episodes, $7,500$ to $8,000$. The graphs are divided into two distinct blocks. The upper block showcases *EF-OnTL* configurations based on $\overline{U}$ as the source selection criteria, while the lower block depicts *EF-OnTL* configurations utilising BP for selecting the source of transfer. Each block
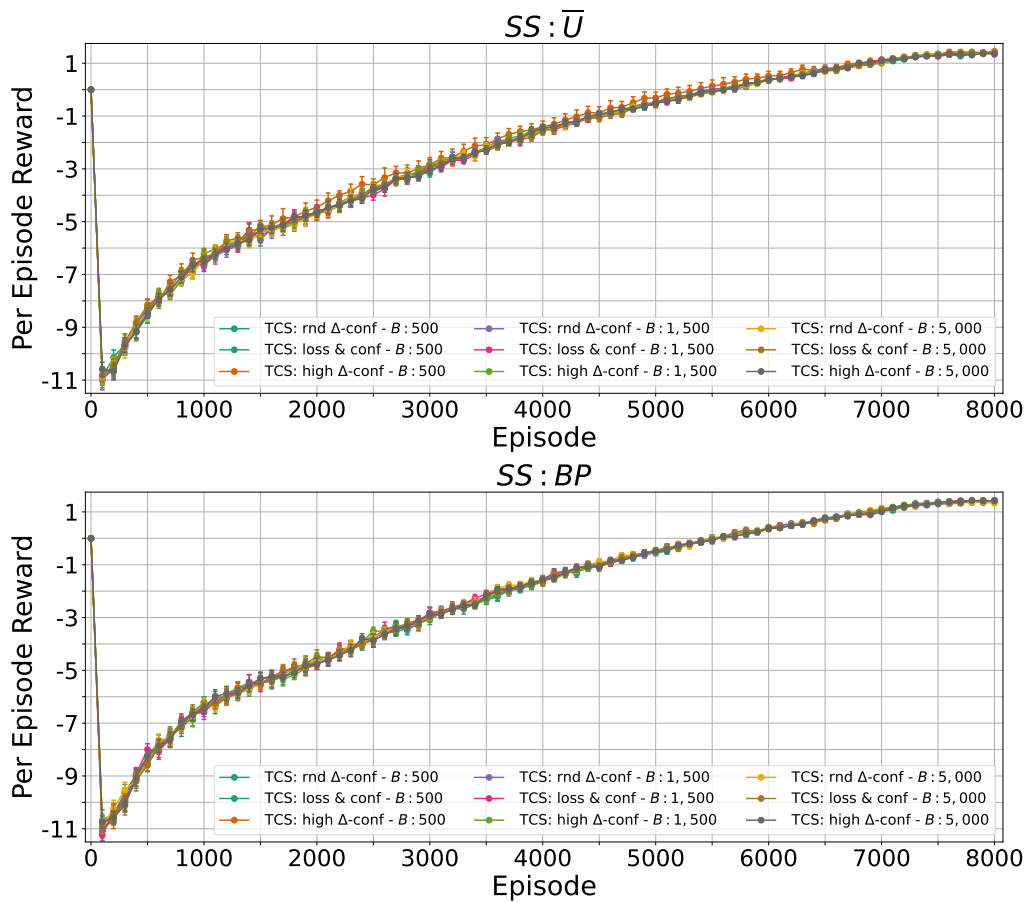
Figure 6.14: *EF-OnTL* learning curves in MT-PP with different parameters. In the top chart the source of transfer is selected by $\overline{U}$, while, in the bottom chart the source is selected by BP.

contains rows representing different evaluation metrics. The top row displays the average agent reward, the middle row represents the number of prey caught by the team, and the bottom row reports the win probability for the team. The configurations utilising a transfer budget B of 500 are reported in the leftmost column, those using B of $1,500$ are placed in the middle column and finally, those with a budget of $5,000$ are placed in the rightmost column.

Overall the criteria used to select the tuple clearly impacts the performance in a similar trend. *high Δ-conf* allows for the better performance while *loss & conf* is slightly below and finally *rnd Δ-conf* is generally the lowest. The only scenarios in which such trends are not confirmed is with a budget B of $5,000$ and $\overline{U}$ as SS. With such settings, *loss & conf* reports the highest performance across all the 3 metrics.

Contrary to what we observed in Cart-Pole, where processing more experiences resulted
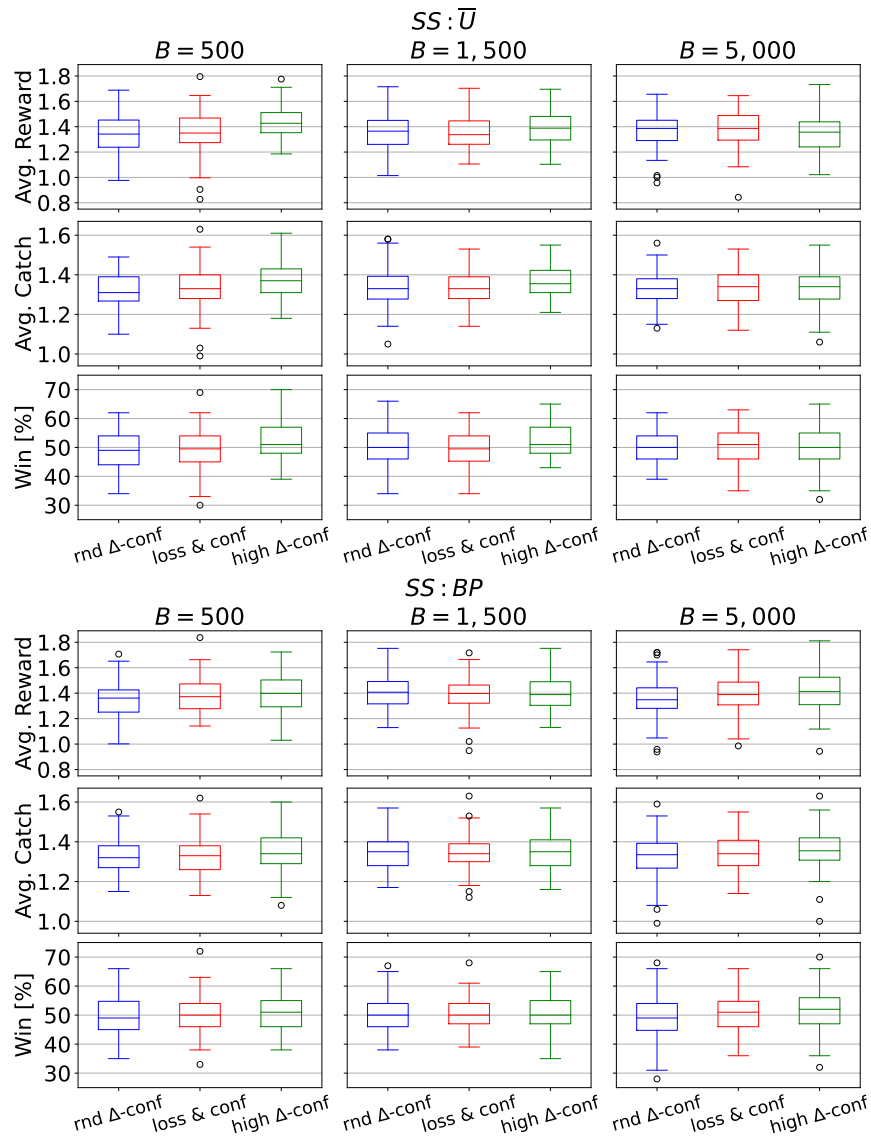
Figure 6.15: Evaluation metrics on the final 500 episodes in the MT-PP environment across the 18 *EF-OnTL* configurations.

in improved performance for the target agents, in MT-PP transferring $5,000$ tuples within each transfer step leads to an overall decrease in performance. For instance, when using TCS as *high Δ-conf* and selecting the source of transfer using $\overline{U}$, the impact of transfer diminishes as the budget B increases.

We believe that an increased budget could potentially lead to a deterioration in performance, similar to what we observed in the preliminary results with an elevated transfer budget. Essentially, the performance decline could be given by the dynamics of the environment, which force a learning agent to visit states with minimal or no information, regardless of the action taken. The probability of encountering these situation is reduced as the agent

learns to explore the surroundings more effectively. Consequently, selecting the source of transfer based on $\overline{U}$ is prone to selecting an agent whose transfer buffer mainly consists of interactions from these low information states. In this context, transferring a substantial volume of interactions could result into a deterioration of the system performance.

To conclude, in the MT-PP benchmark environment, enabling dynamic selection of the source agent using $\overline{U}$ consistently results in improved performance for the transfer enabled team. Moreover, choosing tuples based on the difference between source and target uncertainties, denoted as *high Δ-conf*, consistently leads to superior outcomes.

6.3.2.3   Conclusion of *EF-OnTL* - Transfer Criteria Evaluation

This section has presented a targeted study on *EF-OnTL*, aiming to evaluate the objectives *O4*, *O5*, and *O6*. These objectives have been defined to guide the assessment of different transfer configurations within two benchmark environments: Cart-Pole and Multi-Team Predator-Prey (MT-PP). In total, 18 distinct configurations of *EF-OnTL* have been tested.

O4 addresses **RQ2** and it evaluates whether and to what extent *EF-OnTL* is influenced by the Source Selection Criteria (SS). Similarly, *O5* addresses **RQ3** and it evaluates whether and to what degree Transfer Content Selection Criteria (TCS) influences *EF-OnTL* transfer effectiveness. Finally, *O6* study the relation between the transfer budget B and the transfer outcome.

These evaluation study resulted into an interesting observation. From the experiments, selecting the source of transfer by $\overline{U}$ has led to slightly better performance in the MT-PP environment. That cannot be confirmed by the Cart-Pole scenario, but that might be because of the reduced action space combined with non-sparse reward function.

Contrary to the SS, the experiments have highlighted the need of establishing a criteria to filter the incoming knowledge on a target side, and therefore have highlighted the importance of **RQ3**. In fact, sampling by *rnd Δ-conf* has registered overall the lowest performance. On the contrary, *loss & conf* and *high Δ-conf* has shown comparable performance. However, in the Cart-Pole environment, the *high Δ-conf* approach has demonstrated a lower volatility in return when compared to *loss & conf*.

Lastly, the transfer budget B has shown a contradictory trend depending on the evaluated environment. An increased budget has led to an improvement in the Cart-Pole scenario, whereas it has yielded the opposite outcome in the MT-PP environment, where a lower budget has resulted in higher performance returns. The positive transfer effect observed in Cart-Pole with an increased transfer budget may be attributed to the binary action decision and the non-sparse reward function.

6.3.3 *EF-OnTL* Adaptation to Heterogeneous Dynamics

This section evaluates objective *O8*, assessing the adaptability of *EF-OnTL* in environments with heterogeneous dynamics. To evaluate this objective, three different scenarios were designed to assess the impact of *EF-OnTL* in the 3R2S environment. In two of these scenarios, the agents are exposed to different dynamics.

Table 6.4 presents the three scenarios used in this evaluation. The two scenarios with heterogeneous dynamics are *Single-Agent Mixed Train Data* and *Multi-Agent Same Train Data*. *Multi-Agent Same Train Data*, where the heterogeneity arises by the non-overlapping zones, and this has already been discussed in Section 6.3.1.4. The heterogeneity is due by the non-overlapping zones. On the other hand, in the *Single-Agent Mixed Train Data* scenario, two agents are trained during the morning peak hours while the others on the evening peak hours. Even though agents in both scenarios operate within unique ride-request demand settings, *EF-OnTL* enables the processing of experiences from agents subjected to different dynamics and therefore learning a different policy.

Table 6.4: 3R2S: *EF-OnTL* setup across the three evaluated scenarios.

Scenario ID	Train Demand Set	Test Demand Set	Number of Agents	Multi-Agent
Single-Agent Same Train Data Sec. 6.3.3	Morning	Evening	4	✗
Single-Agent Mixed Train Data Sec. 6.3.3	Morning Evening	Evening	4	✗
Multi-Agent Same Train Data Sec. 6.3.1.4	Morning	Evening	4	✓

In addition to these two scenarios, *Single-Agent Same Train Data* is included to assess whether *EF-OnTL* can achieve positive transfer when transferring across agents in tasks

with common dynamics, similarly to what was observed in the Cart-Pole scenario in Section 6.3.1.1.

The *Single-Agent Same Train Data* and *Single-Agent Mixed Train Data* scenarios presented in this section involve four agents interacting with independent copies of the environment. These two scenarios differ in terms of the requests used by the agents to learn their policies. In *Single-Agent Same Train Data*, all agents are trained using data from the morning peak hours ($7 - 10$am), while in *Single-Agent Mixed Train Data*, two agents learn from data collected during the morning peak hours, and the other two agents learn from data collected during the evening ($6 - 9$pm). In both scenarios the *EF-OnTL* settings matches the one used for the multi-agent scenario and are specified in Table 6.3.

In this evaluation, we included *Single-Agent Mixed Train Data* to investigate the effect of transfer across agents addressing different demand sets. To provide a comprehensive analysis and assess whether transferring between agents learning on different demand sets leads to similar results as the preliminary studies presented in Chapter 3, we also considered the scenario *Single-Agent Same Train Data*, in which agents are enabled to transfer while learning to serve ride-requests from the same demand set.

While the ride-requests used for training the policies in the two scenarios are sourced from different datasets, the test requests are all drawn from the same dataset on the evening peak hours.

In both scenarios, *EF-OnTL* is compared against *no-transfer* method. The results of *Single-Agent Same Train Data* are reported in Figure 6.16, where agents are trained on the morning peak hours and tested on the evening. On the other hand, the results of *Single-Agent Mixed Train Data* are reported in Figure 6.17, where the agents tested are those that are trained and tested on the evening peak hours while enabling the sharing with other agents which are trained on the morning hours.

In the *Single-Agent Same Train Data* scenario, the agents exhibit a different behaviour although both *EF-OnTL* and *no-transfer* share the very same reward model. It appears that *no-transfer* agents have likely learnt to maximise the number of requests served through ride-sharing. On the other hand, *EF-OnTL* agents prioritise requests with more than one passenger, even if it means avoiding ride-sharing. As a result, the distribution of passengers
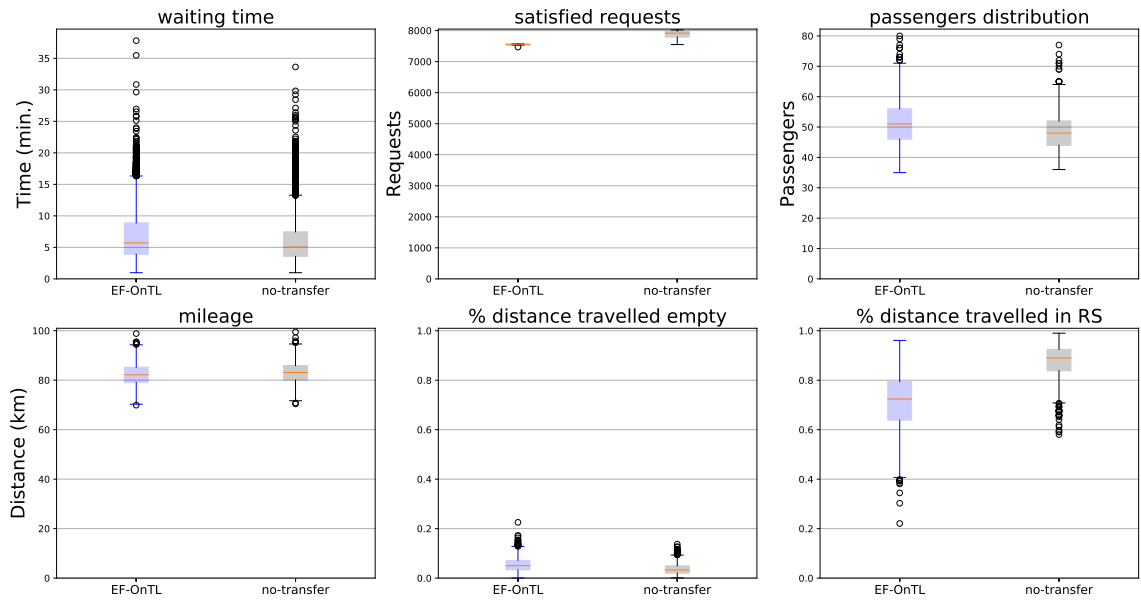
Figure 6.16: *Single-Agent Same Train Data* - train on $7-10$am test on $6-9$pm.

per vehicle is higher for *EF-OnTL* than for the *no-transfer* method, while the distance travelled in ride-sharing and the number of satisfied requests are slightly lower. In conclusion, the difference in the number of served requests is negligible, and the choice of the method preferred depends on the desired behaviour.
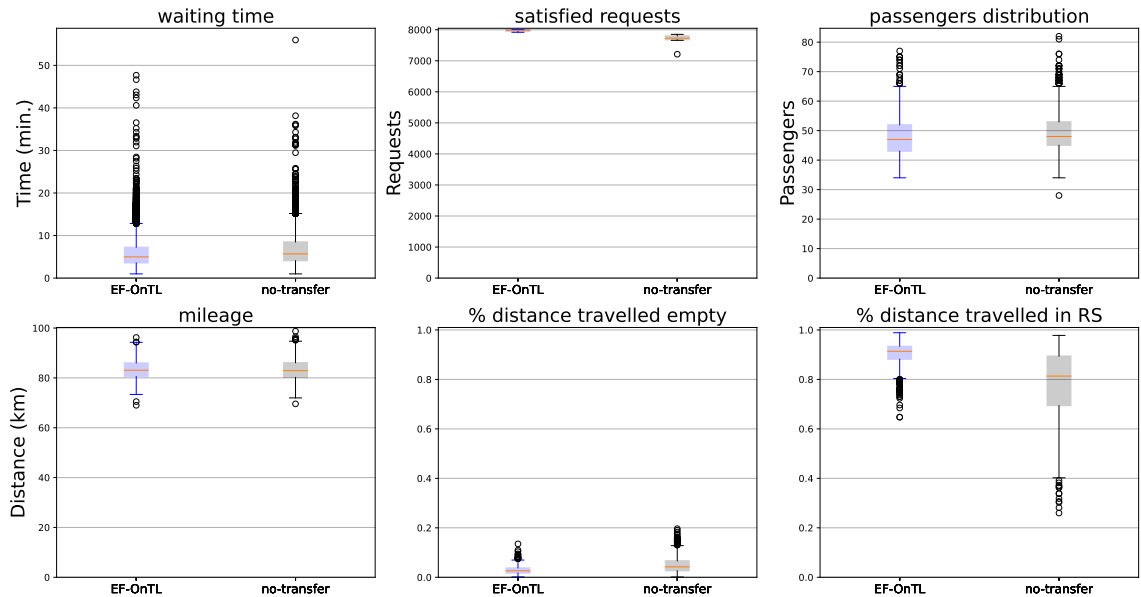
Figure 6.17: *Single-Agent Mixed Train Data* - train on both $7-10$am and $6-9$pm and test on $6-9$pm.

In the *Single-Agent Mixed Train Data* scenario, where transfer occurs between agents

trained on different demand sets originating from the same geographical area, it appears that *EF-OnTL* has enhanced the generalisation ability of the agents. This improvement has led to an increase in the number of requests served compared to the *no-transfer* method. Although the additional requests have resulted in extra distance travelled in ride-sharing, the total distance travelled has not increased compared to the *no-transfer* scenario.

In relation to evaluation objective *O8*, we observed that *EF-OnTL* enabled a fleet of 200 vehicles to achieve an improvement in the total number of requests served when transferring knowledge between agents serving during both the morning and evening peak hours. This improvement in total served requests aligns with what we previously observed in the preliminary results chapter, as presented in Section 3.2.2, where the increase in terms of satisfied requests was approximately 3%. Furthermore, when examining the roles of the transfer sources in the *Single-Agent Mixed Train Data* scenario, we noted that these roles remained relatively stable over time, with one agent being the primary source of transfer in most instances.

The *Single-Agent Mixed Train Data* has demonstrated that enabling the transferring across agents that are learning from different demand set leads a fleet to serve a higher number of requests.

While *EF-OnTL* was not originally designed for enabling this type of transfer, future research should focus on adapting and facilitating transfer between agents specialised in different dynamics. State Action Reward Next-state Random Network Distillation (*sars-RND*) has proven unsuitable for selecting the transfer source in these scenarios, as one agent consistently serves as the source of transfer most of the time. In contrast, *EF-OnTL* should promote varying roles among agents to enhance overall performance.

6.4   Summary

This chapter has presented the evaluation objectives, the baselines and the benchmark environments used to assess *EF-OnTL*.

We compared *EF-OnTL* against multiple baselines. Firstly, *no-transfer*, to measure the improvement against common RL agents that do not share any information. Secondly,

two TL baselines, *OCMAS* and *RCMP*, to compare *EF-OnTL* versus two action-based teacher-student methods. Thirdly, we compared centralised MARL algorithms, QMIX and MADDPG, and *EF-OnTL* in terms of performance.

EF-OnTL and the baselines are evaluated across four different benchmark environments of increasingly complexity: Cart-Pole, MT-PP, HFO and a real-world simulated environment 3R2S. Table 6.2 reports the baselines implemented in each environment.

In the 3R2S environment we designed three different scenarios to observe *EF-OnTL* effect while allowing the transfer across tasks subjected to different underlying dynamics.

To address the research question **RQ1**, we observed that *EF-OnTL* can effectively improve the agents performance involved in a multi-agent system. However, in the *Single-Agent Same Train Data* scenario of the 3R2S environment, *EF-OnTL* has resulted into decreased performance, as shown in Figure 6.10.

The decline in performance, however, may be attributed to the different dynamics of the studied scenario as each agent visits ride-requests originated within a restricted and unique geographical sub-area of the Manhattan borough. Nevertheless, *EF-OnTL* led to an increased number of requests served when enabling the transfer across agents trained on different demand sets originating from the same area as shown in Figure 6.17.

When compared against *OCMAS* and *RCMP*, *EF-OnTL* has proven to be a viable alternative without disrupting the exploration process of a target agent. However, in complex environments, if an optimal expert is available to provide on-demand advice, it should be the preferred choice. Nevertheless, if the teacher is not optimal, *EF-OnTL* should be favoured, as the target agent may be constrained by the suboptimal expertise of the teacher. On the other hand, the MARL baselines, QMIX in MT-PP and MADDPG in HFO, have shown an higher convergence time, possibly because of the MARL extra step to learn the joint state-action values function.

To address the research questions **RQ2** and **RQ3**, this thesis has evaluated multiple transfer settings in the Cart-Pole and MT-PP benchmark environment. The evaluation focused on the transfer impact given by SS, TCS and transfer budget B.

We observed overall that $\overline{U}$, for selecting the source of transfer, and *high Δ-conf*, for

selecting the transfer content, led to better overall performance in both environments. However, while it was not possible to establish a single criterion for **RQ3** in Cart-Pole, the experiments have demonstrated that randomly selecting tuples to be integrated into the target learning process had no significant impact on the final performance.

From the experiments, we could see that each transfer parameter had a different effect depending on the environment encountered by the agents. In Cart-Pole, which has a binary action selection process and a dense reward model, the budget B demonstrated the most significant impact on resulting performance. We observed a greater improvement with a larger budget, B. On the other hand, in MT-PP, which features a sparse reward model and agents frequently revisiting states with no useful information, we observed better overall performance with a smaller budget.

In the end, we noticed that selecting both the source of transfer and the experiences to be integrated into the target learning process based on uncertainty estimation worked better overall. The dynamic identification of the experiences is based on the comparison between the source and target uncertainties to prevent sending redundant information that have already adequately being explored by the target agent. While using $\overline{U}$ and *high Δ-conf* gave more consistent results, how well *EF-OnTL* performs ultimately depends on all the transfer parameters and the specific addressed task.

7   Conclusion

This thesis has introduced Expert-Free Online Transfer Learning (*EF-OnTL*), a novel online experience sharing framework designed to facilitate Transfer Learning (TL) in multi-agent systems where no fixed expert is available. In this chapter, we provide a brief summary of the achievements of this thesis in Section 7.1 and discuss its limitations in Section 7.2. Finally, in Section 7.3, we conclude this dissertation by proposing future research directions.

7.1   Thesis Contribution

This thesis has focused on the challenging problem of enabling dynamic transfer learning in online scenarios, particularly in situations where traditional fixed expert models are absent to advise Reinforcement Learning (RL) agents. The main goal of this thesis has been to study and design an online transfer learning framework based on experience sharing. The main contribution of this thesis is *EF-OnTL*, an expert-free transfer learning framework designed for online experience sharing among agents in a multi-agent environment. *EF-OnTL* enables dynamic teacher selection, allowing the system to choose the most suitable agent to be used as source of transfer at every transfer step, and dynamic transfer content selection, enabling target agents to select the most valuable experience to enhance their policies.

To enable these dynamic selection processes, this thesis also presents State Action Reward Next-state Random Network Distillation (*sars-RND*), a method designed to enhance Random Network Distillation (RND) as an uncertainty estimator in online contexts.

In the design of the TL framework, this thesis has identified and addressed a set of research questions.

Firstly, **RQ1** - can, and if so to what extent, online transfer learning through sharing

of experience across homogeneous agents with no fixed expert contribute to improving the system performance? Secondly, **RQ2** - what criteria can agents use to identify the suitable agent to be used a source of transfer? Thirdly, **RQ3** - what criteria should an agent use to filter incoming knowledge?

This thesis began with a feasibility study to analyse the impact of experience sharing in a reduced set of challenges given by an offline transfer learning context. Consequently, acknowledging the early results, this thesis introduced Expert-Free Online Transfer Learning (*EF-OnTL*). *EF-OnTL* stands as a transfer learning framework tailored for multi-agent systems that effectively eliminates the need for a pre-defined expert teacher by employing a dynamic selection process. This process chooses a temporary expert at each transfer step based on the real-time performance metrics of the agents, such as average cumulated rewards in recent episodes and average uncertainty in recent experiences.

The chosen temporary expert serves as the source of transfer, sharing a portion of its collected experience with the other agents, which are designated as target agents. These target agents filter and sample a batch of experiences from the transferred buffer to be integrated into their own learning processes. To prioritise certain experiences, a target agent relies on metrics like uncertainty and expected surprise. Uncertainty is compared against the source agent's uncertainty, while the expected surprise is estimated as the loss of the underlying deep learning model.

The state-of-the-art approaches enabling transfer across homogeneous agents are mostly based on fixed expert agents, where already trained agents provide advice to novel learning agents. Furthermore, the criteria used to decide whether to provide advice or not are often based on a single-side, either the source or target of transfer.

The teacher-student framework can be adapted to enable the transfer across learning agents, as demonstrated by Ilhan et al. [52]. However, agents need continuous interaction to establish whether an advice is needed or not.

EF-OnTL overcomes the need for a fixed expert agent and the need for continuous interaction by transferring a batch of experiences aimed at improving the learning model of a target agent. Furthermore, the experiences transferred are sampled on criteria defined for both source and target agents.

In conclusion, the answers to our posed research questions are as follows:

- **Can, and if so to what extent, online transfer learning through sharing of experience across homogeneous agents with no fixed expert contribute to improving the system performance?** – Compared to a traditional RL scenario where agents do not share any information, *EF-OnTL* has successfully shown the capability to improve the agents' performance when applied to multiple homogeneous agents or multi-agent systems. While expert-based action-advice TL approaches can potentially limit the performance of target agents, *EF-OnTL* prevents such negative impacts on a target agent's performance. This is achieved by eliminating the requirement for an expert agent and enabling dynamic teacher selection at each transfer step.

  The extent of improvement shown by *EF-OnTL* is closely associated to the complexity of the task being undertaken by the agents, a trend that has been similarly observed among other TL baselines. In the Half Field Offense (HFO) benchmark environment, *EF-OnTL* enables the offensive team to double the number of goals scored when compared to independent RL agents. Additionally, *EF-OnTL* achieves comparable performance to expert-based TL systems. However, it is worth noting that *EF-OnTL* typically requires nearly double the amount of time to converge to a similar level of performance.

- **What criteria can agents use to identify the suitable agent to be used as source of transfer?** – in this thesis we have investigated two different criteria to select the source of transfer for *EF-OnTL*, Average Uncertainty ($\overline{U}$) and Best Performance (BP), that agents can use to identify a suitable agent to be used as source of transfer within a transfer step. $\overline{U}$ selects as source of transfer the agent with lowest average uncertainty across the experiences stored in its transfer buffer. On the other hand, BP selects as source of transfer the agent with highest cumulated reward within the most recent episodes interval.

  Based on the experiments presented in this work, the most effective criterion used by agents in *EF-OnTL* to dynamically select the source of transfer is $\overline{U}$. Nevertheless, we have observed that the criteria used for selecting the source of transfer may have

minimal impact on the final performance in certain tasks, such as Cart-Pole. Further investigation is necessary to determine whether this limited impact is attributed to the simplicity of the Cart-Pole task.

- **What criteria should an agent use to filter incoming knowledge?** − in this thesis we have presented three different criteria that agents could use to sample relevant experiences to be integrated into their learning processes, transfer Randomly based on threshold determined by Delta Confidence (*rnd Δ-conf*), transfer based on Higher Delta Confidence (*high Δ-conf*) and transfer based on Higher Loss and Confidence combined (*loss & conf*). These criteria are based on Δ-conf, which expresses the difference between the target and source uncertainty, and expected surprise, defined solely over target agent and approximated through temporal difference error. *rnd Δ-conf* enables an agent to randomly select the experiences from tuples with a Δ-conf higher than the median value computed across all tuples. *high Δ-conf* enables an agent to sort in decreasing order the incoming experiences by the Δ-conf and to select the top entries to be integrated within its learning model. *loss & conf* enables an agent to balance expected surprise and uncertainty. To balance the different scales, uncertainty and surprise values are normalised within a common interval and weighted equally. Finally, agent selects the tuples with the highest values to be integrated within its learning model.

  Selecting the experience to be integrated by *rnd Δ-conf* has demonstrated the lowest overall impact on the system performance. On the other hand, *loss & conf* and *high Δ-conf* have both shown a better performance improvement. However, selecting the experiences by *high Δ-conf*, and thereby comparing the uncertainties of sender and receiver agent, has consistently led to a performance improvement.

Given the pivotal role of uncertainty, the final performance of the presented framework depends on the capacity of the estimator model used to approximate the agent's epistemic uncertainty. In this thesis, we have presented *sars-RND*, an extension of RND, designed to overcome the limitations of RND in estimating the epistemic uncertainty of a learning agent. To summarise, RND estimates uncertainty based solely on the visited state. We believe that the state alone is insufficient for estimating the uncertainty in a learning agent

since it does not provide information about the action taken. *sars-RND* extends RND by including into the estimation the action, the next state and the reward observed. Based on the comparison presented in Section 3.4.1, *sars-RND* successfully identifies different actions while recognising previously visited states and allowing for an uncertainty trend similar to that of RND.

Finally, while sampling the experiences by *high Δ-conf* certainly provided more consistent results, the final performance given by *EF-OnTL* is dependent of the combination of the transfer settings parameters, which need to be carefully evaluated based on the addressed task.

In the current state of the art, experience sharing is mostly used in offline transfer learning as a phase prior to the exploration. However, this thesis has demonstrated that combining independently collected experience with external sources may be beneficial to the learning agents.

When an agent with deep expertise, either human or artificial, is available to guide the exploration of learning agents, action-advice transfer methods may be preferred as these ensure a higher return over a reduced time frame. However, having an expert dedicated to this mentoring task is expensive. Thus, combining both expert-based and online experience sharing may allow to keep the same level of performance while lowering the expert-related costs.

However, given the impracticality of having an oracle agent, the expert agent, specifically human, may limit its choices to a subset of familiar actions. This constraint narrows the target agent's learning opportunities. In contrast, sharing raw experience is less likely to invalidate the learning phase of an agent. In fact, in the event of negative transfer, the target agent may take slightly longer to converge, but its performance will not be compromised.

7.2   Limitations

While the results of this thesis are promising, there are several limitations to consider. The training time for policies is increased when compared to agents that do not transfer any information. This increase in training time is not caused by the dynamic selection of

source and experiences during each transfer step but rather by the optimisation step of the uncertainty estimator model.

The *sars-RND* model is updated after each visited tuple, which involves back-propagation of gradients. While a single back-propagation step is relatively fast, over millions of interactions, it introduces a noticeable delay. To address this, back-propagation could be delayed and aggregated over a small batch of samples, but this would result in a delay in updating uncertainties.

The impact of experience sharing in *EF-OnTL* requires substantial time to be observed when compared to action as advice. When sharing action-advice it is possible to assess whether the transfer is leading a target agent towards positive transfer or not. This assessment requires the monitoring of the behaviour exhibited by the target agent. On the other hand, in *EF-OnTL*, with the sharing of experiences, it is hard to discern whether a certain behaviour is due to the transferred knowledge or self-exploration of an agent. From a human perspective, the sharing of experience is a black box where it is nearly impossible, or at least more complicated, to appreciate the value of the advice when compared to one-shot action.

In addition, our results have shown that *EF-OnTL* can occasionally lead to negative transfer with certain transfer settings. When implementing *EF-OnTL* from scratch, it can be challenging to determine whether the transfer will result in a positive or negative outcome until the simulation is completed. In contrast, when transferring from an experienced agent through the teacher-student framework, which relies on action-advice, it allows for faster estimation of whether the transfer will lead to a positive or negative outcome.

7.3   Future Work

When considering transfer learning among homogeneous agents, experience sharing is often not the primary choice. Instead, many TL frameworks, following the teacher-student paradigm, opt to enable agents to share actions as advice. In this context, *EF-OnTL* has demonstrated that enabling dynamic sharing of selected batches of experiences is a viable alternative to the traditional action-advice based teacher-student framework.

The effectiveness of the teacher-student framework is often dependent on the availability

of an expert agent to provide guidance, and it often necessitates extensive experimentation to fine-tune the algorithm parameters.

In contrast, *EF-OnTL* does not rely on a fixed expert agent to be effective. It has been designed with a forward-looking perspective, envisioning a future where transfer is self-regulated, aiming to eliminate the need for intricate parameter adjusting.

With this direction in mind and considering the observed results, we believe that having fixed criteria for selecting the source of transfer and determining the interactions to be integrated into the learning process may not be optimal. As a future work direction, we propose enabling dynamic criteria selection to choose both the source of transfer and the interactions to be integrated into a target agent's learning process. We believe that using multiple criteria based on the system's status could further enhance the system performance.

An additional research direction concerns the investigation of dynamic adjustment of the transfer budget B and its impact on the final performance, when compared to a fixed budget. Furthermore, our experiments did not consider the frequency of transfer, so a necessary step for the future development of *EF-OnTL* is to assess the effect of increased transfer frequency with a smaller budget against lower transfer frequency with an increased budget.

In addition to *EF-OnTL*, this thesis introduced *sars-RND* to extend the RND model used for estimating epistemic uncertainty. In the future, we plan to refine *sars-RND* by reducing its input to state-action pairs and assess the impact that this adjustment might have on both the estimation of epistemic uncertainty and the transfer effect.

Finally, the findings of this thesis could be integrated into other RL methodologies that necessitate the prioritisation of specific samples. An intriguing application would be the adaptation of *EF-OnTL* to shape a dataset for offline RL tasks. Subsequently, it would be insightful to train an agent prioritising certain tuples, deemed to be the most beneficial for learning. Comparing the performance of this approach against other sampling methods from the dataset may result in enhancing the learning outcomes.

Appendix

A Half Field Offense (HFO) State

For the experiments carried out in this research, perceived states consists of 95 values. From the manual available online [147]:

- 0 Self_Pos_Valid [Valid] Indicates if self position is valid.

- 1 Self_Vel_Valid [Valid] Indicates if the agent's velocity is valid.

- 2-3 Self_Vel_Ang [Angle] Angle of agent's velocity.

- 4 Self_Vel_Mag [Other] Magnitude of the agent's velocity.

- 5-6 Self_Ang [Angle] Agent's Global Body Angle.

- 7 Stamina [Other] Agent's Stamina: Low stamina slows movement.

- 8 Frozen [Boolean] Indicates if the agent is Frozen. Frozen status can happen when tackling or being tackled by another player.

- 9 Colliding_with_ball [Boolean] Indicates the agent is colliding with the ball.

- 10 Colliding_with_player [Boolean] Indicates the agent is colliding with another player.

- 11 Colliding_with_post [Boolean] Indicates the agent is colliding with a goal post.

- 12 Kickable [Boolean] Indicates the agent is able to kick the ball.

- 13-15 Goal Center [Landmark] Center point between the goal posts.

- 16-18 Goal Post Top [Landmark] Top goal post.

- 19-21 Goal Post Bot [Landmark] Bottom goal post.

- 22-24 Penalty Box Center [Landmark] Center of the penalty box line.

- 25-27 Penalty Box Top [Landmark] Top corner of the penalty box.

- 28-30 Penalty Box Bot [Landmark] Bottom corner of the penalty box.

- 31-33 Center Field [Landmark] The left middle point of the RoboCup field (note that this is not the center of the HFO play area).

- 34-36 Corner Top Left [Landmark] Top left corner HFO Playfield.

- 37-39 Corner Top Right [Landmark] Top right corner HFO Playfield.

- 40-42 Corner Bot Right [Landmark] Bottom right corner HFO Playfield.

- 43-45 Corner Bot Left [Landmark] Bottom left corner HFO Playfield.

- 46 OOB Left Dist [Proximity] Proximity to the nearest point of the left side of the HFO playable area. E.g. distance remaining before the agent goes out of bounds in left field.

- 47 OOB Right Dist [Proximity] Proximity to the right field line.

- 48 OOB Top Dist [Proximity] Proximity to the top field line.

- 49 OOB Bot Dist [Proximity] Proximity to the bottom field line.

- 50 Ball Pos Valid [Valid] Indicates the ball position estimate is valid.

- 51-52 Ball Angle [Angle] Agent's angle to the ball.

- 53 Ball Dist [Proximity] Proximity to the ball.

- 54 Ball Vel Valid [Valid] Indicates the ball velocity estimate is valid.

- 55 Ball Vel Mag [Other] Magnitude of the ball's velocity.

- 56-57 Ball Vel Ang [Angle] Global angle of ball velocity.

- $8T$ Teammate Features [Player] One teammate feature set (8 features) for each teammate active in HFO, sorted by proximity to the agent.

- $8O$ Opponent Features [Player] One opponent feature set (8 features) for each opponent active in HFO, sorted by proximity to the player.

- $1T$ Teammate Uniform Nums [Unum] One uniform number for each teammate active in HFO, sorted by proximity to the agent.

- $1O$ Opponent Uniform Nums [Unum] One uniform number for each opponent active in HFO, sorted by proximity to the player.

- $+1$ Last_Action_Success_Possible [Boolean] Whether there is any chance the last action taken was successful, either in accomplishing the usual intent of the action or (primarily for the offense) in some other way such as getting out of a goal-collision state.

Where xT means that there are x value for each teammate and xO x values for each opponent.

B   *EF-OnTL* - Transfer Criteria Evaluation in Multi-Team Predator-Prey (MT-PP)

This section presents the expanded results obtained by Expert-Free Online Transfer Learning (*EF-OnTL*) while varying the transfer settings in the MT-PP. The evaluation matches the results shown in Figure 6.14. However, here we show the learning curves compared by the Transfer Content Selection Criteria (TCS), shown in Figure 7.1, used to filter the experiences on a target agent and by the budget B used within a transfer step, shown in Figure 7.2.

(a) *rnd Δ-conf*

(b) *high Δ-conf*

(c) *loss & conf*
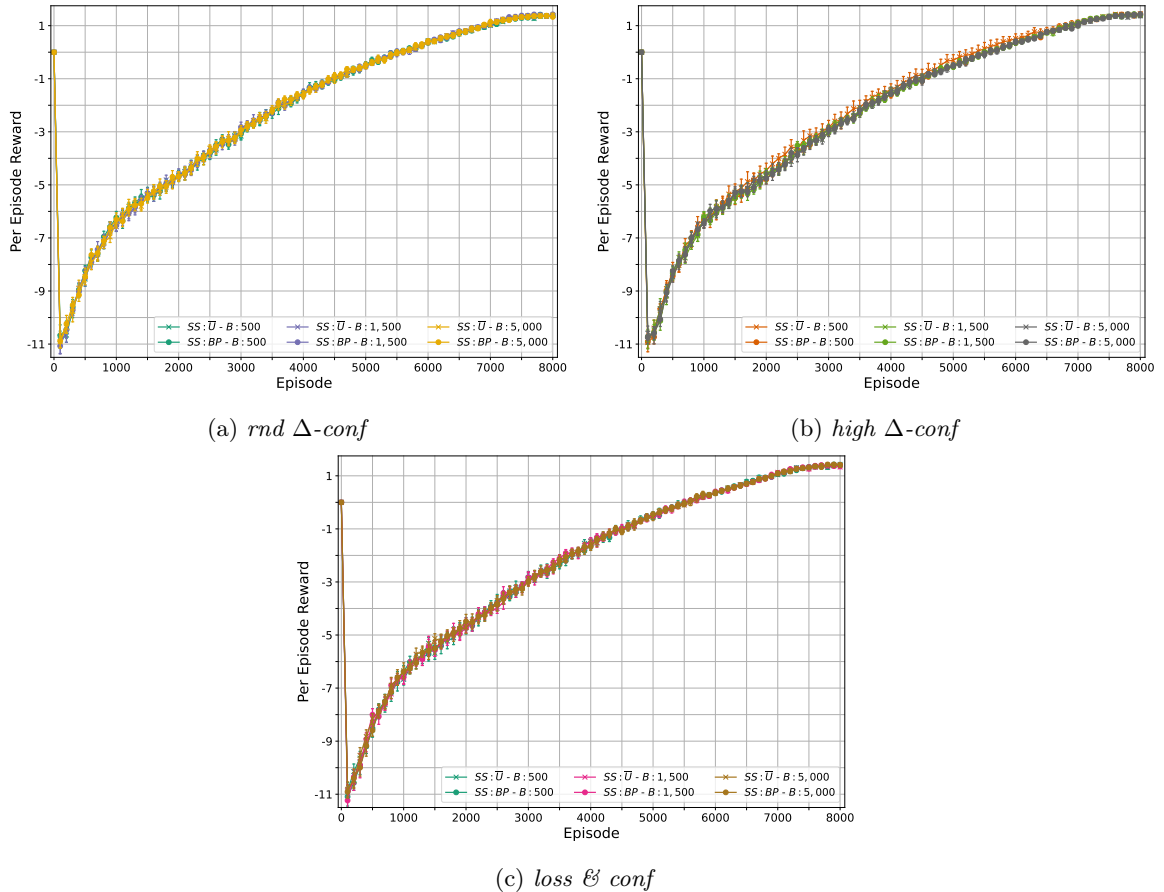
Figure 7.1: *EF-OnTL* learning curves organised by TCS used by a target agent to filter incoming knowledge.

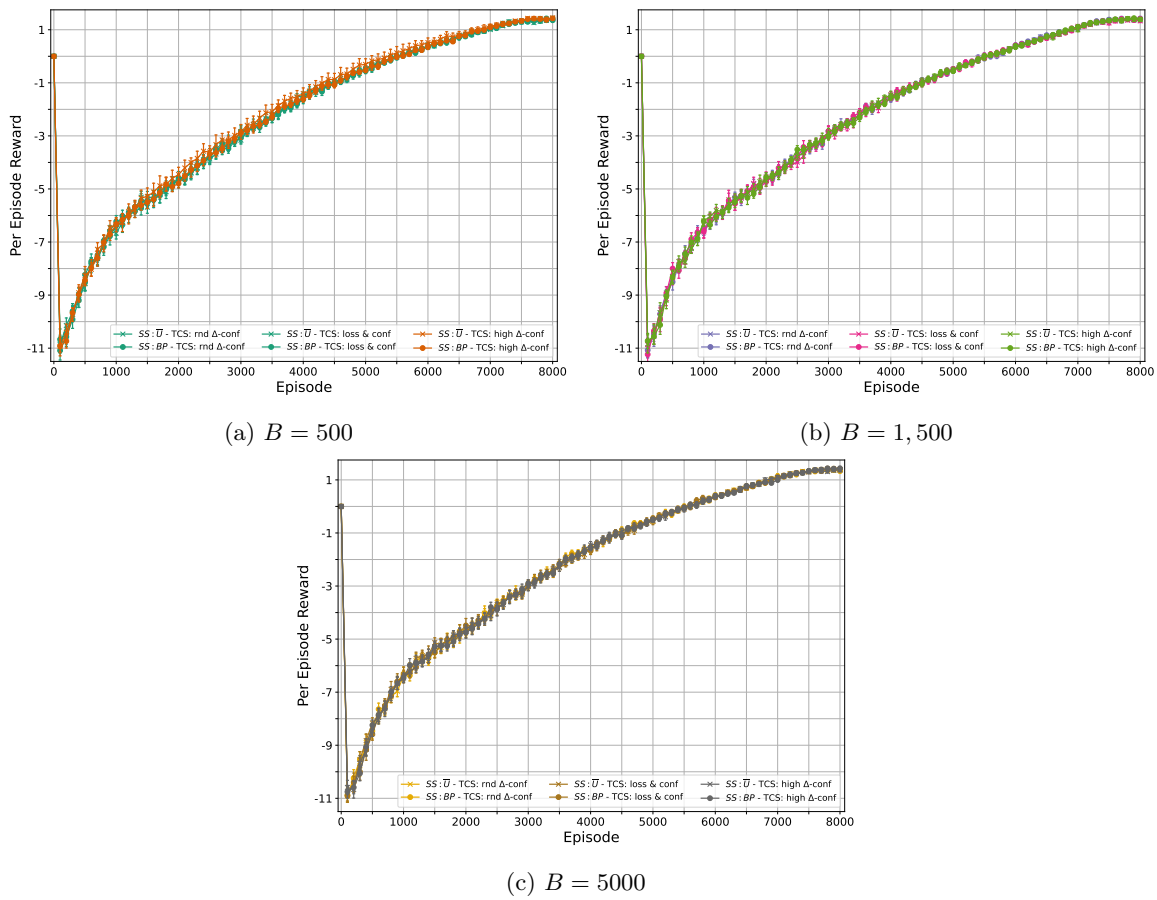(a) $B = 500$

(b) $B = 1,500$

(c) $B = 5000$

Figure 7.2: *EF-OnTL* learning curves organised by the transfer budget B in MT-PP.

C   Requesting Confidence-Moderated Policy - Analysis

This section investigates the low performance registered by the Requesting Confidence-Moderated Policy (*RCMP*) method. In particular, it explores the motivations behind the negative transfer in Cart-Pole and includes an analysis of threshold sensitivity in the MT-PP environment.

C.1   Cart-Pole

In the Cart-Pole benchmark environment, as shown in the evaluation graph in Figure 6.5, *RCMP* demonstrates an initial jump-start in performance from the beginning up to episode 200. Subsequently, there is a brief decline in performance, followed by a gradual increase in the reward starting from episode 500 onward.

The notable jump-start is a result of the continuous provision of advice from the expert teachers. However, the subsequent drop in performance can be attributed to a lack of advice caused by an inadequate allocation of the budget, as shown in Figure 7.3. This issue arises due to the threshold used to trigger the advice process.

Figure 7.3: Budget utilisation of *OCMAS* and *RCMP* in the Cart-Pole environment. This graph complements Figure 6.5.

The normalisation process applied to maintain a consistent scale for uncertainty, along with the binary action decision, results in the ensemble used as an uncertainty estimator showing low granularity in uncertainty estimation. Uncertainty falls into three possibilities based on the agreement among the five heads: 0 when all heads agree, 0.2 when a single head disagrees, and 0.3 when two heads disagree with the others.

C.2   Multi-Team Predator-Prey

In the MT-PP benchmark environment, we observed a more balanced budget utilisation throughout the episodes, as shown in Figure 7.4. Due to the discrete action decision process involving multiple actions, the uncertainty estimated by the ensemble allowed for a sufficient range of values to ensure a fair budget allocation.
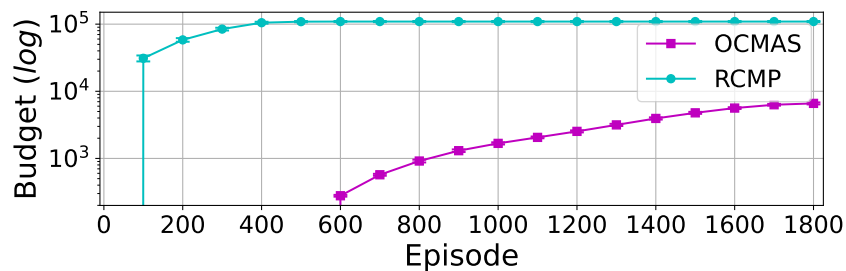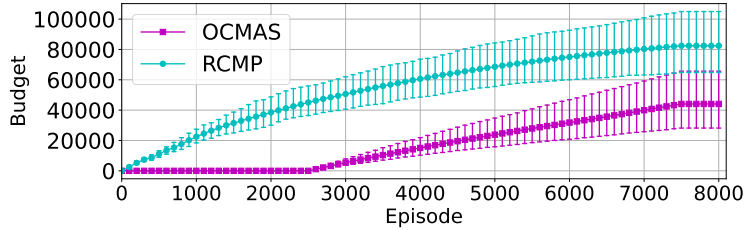

Figure 7.4: Budget utilisation of *OCMAS* and *RCMP* in the MT-PP environment. This graph complements Figure 6.6.

In the MT-PP environment, we have studied the performance of *RCMP* in relation to the selected threshold that initialise the advising process. For comparison, we present the learning curves, the MT-PP metrics, including the average number of preys caught from the opposite team, and the budget utilisation.

The results are reported in Figure 7.5, where we evaluated five different threshold levels: 0.01, 0.02, 0.03, and 0.05. The results presented in the thesis, and that are shown in Figure 6.6, are based on a threshold of 0.02.

A lower threshold leads the agent to be guided for longer, resulting in a higher cumulative reward during the training. However, the final performance in the tracked metrics are significantly lower. For instance, with a threshold of 0.01 we observed an overall higher cumulative reward until episode 7,000, as shown in Figure 7.5a. Eventually, this lower threshold led to a deterioration of performance in the tracked metrics, as shown in Figure 7.5b. It is possible that the target agents have learned to prioritise the catch action, which leads to catching the wrong prey and negatively impacted their overall performance.

Increasing the threshold to 0.05, results in lower budget utilisation, as shown in Figure 7.5c. Consequently, the learning curve during the initial episodes, up to episode 4,000, is lower compared to a more balanced threshold like 0.02. Nevertheless, the final tracked metrics remain comparable between the two thresholds. The main difference is that a
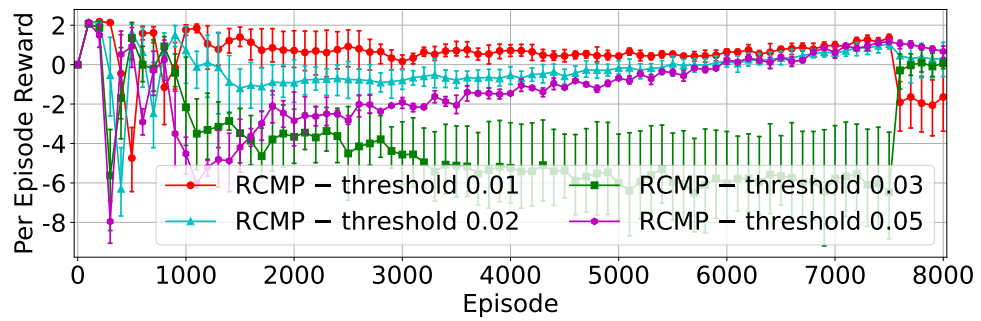
(a) Learning curves.
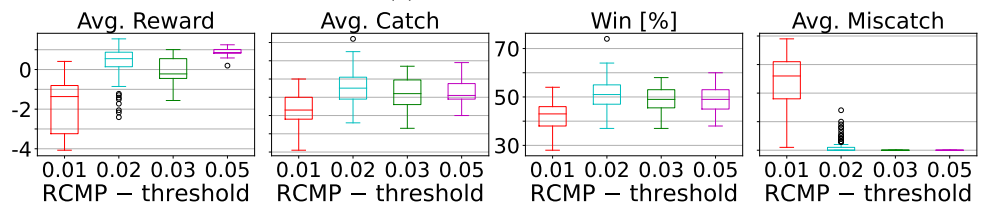
(b) Evaluation metrics.
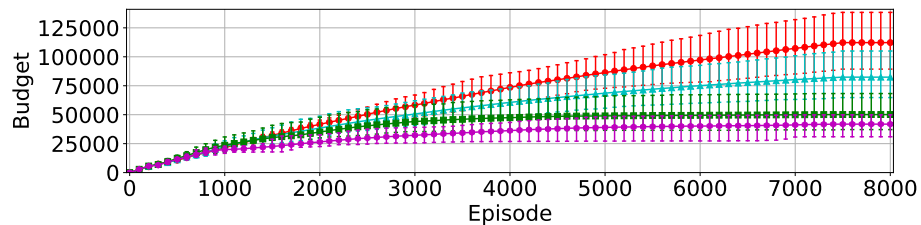
(c) Budget consumption.

Figure 7.5: *RCMP* performance in the MT-PP benchmark environment with different thresholds.

higher reduces the prioritisation of the catch action and therefore prevents the capturing of adversarial prey.

From the analysis presented in this section, it is evident that *RCMP* necessitates significant experimentation to investigate the correlation between transfer outcomes and the threshold used by target agents.

Acronyms

3R2S   Ride-Sharing Ride-Requests Simulator

A2A   Agent-to-Agent

BP   Best Performance

CCL   Cooperative Centralised Learning

CTDE   Centralised Training with Decentralised Execution

DDPG   Deep Deterministic Policy Gradient

DQN   Deep Q-Network

DRL   Deep Reinforcement Learning

EF-OnTL   Expert-Free Online Transfer Learning

HFO   Half Field Offense

high Δ-conf   transfer based on Higher Delta Confidence

H   Horizon

IL   Independent Learners

loss & conf   transfer based on Higher Loss and Confidence combined

MADDPG   Multi-Agent Deep Deterministic Policy Gradient [61]

MARL   Multi-Agent Reinforcement Learning

MAS   Multi-Agent System

ML   Machine Learning

MDP   Markov Decision Process

MSE   Mean Squared Error

MT-PP Multi-Team Predator-Prey

NN Neural Network

OCMAS Online Confidence-Moderated Advice Sharing

POMDP Partially-Observable Markov Decision Process

PA-DDPG Parameterised Action Deep Deterministic Policy Gradient

PPO Proximal Policy Optimisation

RCMP Requesting Confidence-Moderated Policy

RL Reinforcement Learning

RND Random Network Distillation

rnd Δ-conf transfer Randomly based on threshold determined by Delta Confidence

ST-PP Single-Team Predator-Prey

sars-RND State Action Reward Next-state Random Network Distillation

SS Source Selection Criteria

SUMO Simulation of Urban MObility [74]

VDN Value Decomposition Network

TD Temporal-Difference

TCS Transfer Content Selection Criteria

TL Transfer Learning

T2T Task-to-Task

$\overline{U}$ Average Uncertainty

Bibliography

[1] A. Castagna, M. Guériau, G. Vizzari, and I. Dusparic, "Demand-responsive zone generation for real-time vehicle rebalancing in ride-sharing fleets," in *11th International Workshop on Agents in Traffic and Transportation (ATT 2020) at ECAI 2020*, 2020.

[2] ——, "Demand-responsive rebalancing zone generation for reinforcement learning-based on-demand mobility," *AI Communications*, vol. 34, no. 1, pp. 73–88, 2021.

[3] A. Castagna and I. Dusparic, "Multi-agent transfer learning in reinforcement learning-based ride-sharing systems," in *International Conference on Agents and Artificial Intelligence (ICAART)*, 2022.

[4] ——, "Expert-free online transfer learning in multi-agent reinforcement learning," in *(Accepted to) European Conference on Artificial Intelligence ECAI 2023*, 2023.

[5] C. Hasan, A. Agapitos, D. Lynch, A. Castagna, G. Cruciata, H. Wang, and A. Milenovic, "Continual model-based reinforcement learning for data efficient wireless network optimisation," in *(Accepted to) European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases*, 06 2023.

[6] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction.* MIT press, 2018.

[7] J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov, "Proximal policy optimization algorithms," *arXiv preprint arXiv:1707.06347*, 2017.

[8] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," *arXiv preprint arXiv:1511.04143*, 2015.

[9] NYC Taxi and Limousine Commission, "Tlc trip record data," 2020. [Online]. Available: https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page

[10] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," 2016.

[11] W. Zhao, J. P. Queralta, and T. Westerlund, "Sim-to-real transfer in deep reinforcement learning for robotics: a survey," in *2020 IEEE symposium series on computational intelligence (SSCI).* IEEE, 2020, pp. 737–744.

[12] J. Kober, J. A. Bagnell, and J. Peters, "Reinforcement learning in robotics: A survey," *The International Journal of Robotics Research*, vol. 32, no. 11, pp. 1238–1274, 2013.

[13] A. S. Polydoros and L. Nalpantidis, "Survey of model-based reinforcement learning: Applications on robotics," *Journal of Intelligent & Robotic Systems*, vol. 86, no. 2, pp. 153–173, 2017.

[14] A. Charpentier, R. Elie, and C. Remlinger, "Reinforcement learning in economics and finance," *Computational Economics*, pp. 1–38, 2021.

[15] B. M. Hambly, R. Xu, and H. Yang, "Recent advances in reinforcement learning in finance," *Recent Advances in Reinforcement Learning in Finance (November 24, 2021)*, 2021.

[16] M. A. Wiering *et al.*, "Multi-agent reinforcement learning for traffic light control," in *Machine Learning: Proceedings of the Seventeenth International Conference (ICML'2000)*, 2000, pp. 1151–1158.

[17] H. Wei, G. Zheng, V. Gayah, and Z. Li, "Recent advances in reinforcement learning for traffic signal control: A survey of models and evaluation," *ACM SIGKDD Explorations Newsletter*, vol. 22, no. 2, pp. 12–18, 2021.

[18] M. Guériau, F. Cugurullo, R. A. Acheampong, and I. Dusparic, "Shared autonomous mobility on demand: A learning-based approach and its performance in the presence of traffic congestion," *IEEE Intelligent Transportation Systems Magazine*, vol. 12, no. 4, pp. 208–218, 2020.

[19] Z. Xiong, Y. Zhang, D. Niyato, R. Deng, P. Wang, and L.-C. Wang, "Deep reinforcement learning for mobile 5g and beyond: Fundamentals, applications, and challenges," *IEEE Vehicular Technology Magazine*, vol. 14, no. 2, pp. 44–52, 2019.

[20] V. Sciancalepore, X. Costa-Perez, and A. Banchs, "Rl-nsb: Reinforcement learning-based 5g network slice broker," *IEEE/ACM Transactions on Networking*, vol. 27, no. 4, pp. 1543–1557, 2019.

[21] S. K. Zhou, H. N. Le, K. Luu, H. V. Nguyen, and N. Ayache, "Deep reinforcement learning in medical imaging: A literature review," *Medical image analysis*, vol. 73, p. 102193, 2021.

[22] A. Coronato, M. Naeem, G. De Pietro, and G. Paragliola, "Reinforcement learning for intelligent healthcare applications: A survey," *Artificial Intelligence in Medicine*, vol. 109, p. 101964, 2020.

[23] A. Jonsson, "Deep reinforcement learning in medicine," *Kidney diseases*, vol. 5, no. 1, pp. 18–22, 2019.

[24] A. R. Sharma and P. Kaushik, "Literature survey of statistical, deep and reinforcement learning in natural language processing," in *2017 International Conference on Computing, Communication and Automation (ICCCA)*. IEEE, 2017, pp. 350–354.

[25] V. Uc-Cetina, N. Navarro-Guerrero, A. Martin-Gonzalez, C. Weber, and S. Wermter, "Survey on reinforcement learning for language processing," *Artificial Intelligence Review*, pp. 1–33, 2022.

[26] A. A. Rusu, N. C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, and R. Hadsell, "Progressive neural networks," *arXiv preprint arXiv:1606.04671*, 2016.

[27] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y. W. Teh, R. Pascanu, and R. Hadsell, "Progress & compress: A scalable framework for continual learning," in *International conference on machine learning*. PMLR, 2018, pp. 4528–4537.

[28] Z. Wang, Z. Qin, X. Tang, J. Ye, and H. Zhu, "Deep reinforcement learning with knowledge transfer for online rides order dispatching," in *2018 IEEE International Conference on Data Mining (ICDM).* IEEE, 2018, pp. 617–626.

[29] C. D'Eramo, D. Tateo, A. Bonarini, M. Restelli, J. Peters *et al.*, "Sharing knowledge in multi-task deep reinforcement learning," in *8th International Conference on Learning Representations,{ICLR} 2020, Addis Ababa, Ethiopia, April 26-30, 2020.* OpenReview. net, 2020, pp. 1–11.

[30] C. Devin, A. Gupta, T. Darrell, P. Abbeel, and S. Levine, "Learning modular neural network policies for multi-task and multi-robot transfer," in *2017 IEEE international conference on robotics and automation (ICRA).* IEEE, 2017, pp. 2169–2176.

[31] T. D. Kulkarni, A. Saeedi, S. Gautam, and S. J. Gershman, "Deep successor reinforcement learning," *arXiv preprint arXiv:1606.02396*, 2016.

[32] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A. A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska *et al.*, "Overcoming catastrophic forgetting in neural networks," *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.

[33] R. Glatt, F. L. Da Silva, and A. H. R. Costa, "Towards knowledge transfer in deep reinforcement learning," in *2016 5th Brazilian Conference on Intelligent Systems (BRACIS).* IEEE, 2016, pp. 91–96.

[34] H. Yin and S. Pan, "Knowledge transfer for deep reinforcement learning with hierarchical experience replay," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 31, no. 1, 2017.

[35] A. A. Rusu, S. G. Colmenarejo, C. Gulcehre, G. Desjardins, J. Kirkpatrick, R. Pascanu, V. Mnih, K. Kavukcuoglu, and R. Hadsell, "Policy distillation," *arXiv preprint arXiv:1511.06295*, 2015.

[36] R. Traoré, H. Caselles-Dupré, T. Lesort, T. Sun, G. Cai, N. Díaz-Rodríguez, and D. Filliat, "Discorl: Continual reinforcement learning via policy distillation," *arXiv preprint arXiv:1907.05855*, 2019.

[37] A. Barreto, W. Dabney, R. Munos, J. J. Hunt, T. Schaul, H. P. van Hasselt, and D. Silver, "Successor features for transfer in reinforcement learning," *Advances in neural information processing systems*, vol. 30, 2017.

[38] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, I. Osband *et al.*, "Deep q-learning from demonstrations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[39] A. Lazaric, M. Restelli, and A. Bonarini, "Transfer of samples in batch reinforcement learning," in *Proceedings of the 25th international conference on Machine learning*, 2008, pp. 544–551.

[40] H. R. Walke, J. H. Yang, A. Yu, A. Kumar, J. Orbik, A. Singh, and S. Levine, "Don't start from scratch: Leveraging prior data to automate robotic reinforcement learning," in *Conference on Robot Learning.* PMLR, 2023, pp. 1652–1662.

[41] M. E. Taylor, N. Carboni, A. Fachantidis, I. Vlahavas, and L. Torrey, "Reinforcement learning agents providing advice in complex video games," *Connection Science*, vol. 26, no. 1, pp. 45–63, 2014.

[42] F. L. Da Silva, P. Hernandez-Leal, B. Kartal, and M. E. Taylor, "Uncertainty-aware action advising for deep reinforcement learning agents," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 5792–5799.

[43] C. Zhu, Y. Cai, H.-f. Leung, and S. Hu, "Learning by reusing previous advice in teacher-student paradigm," in *Proceedings of the 19th International Conference on Autonomous Agents and MultiAgent Systems*, 2020, pp. 1674–1682.

[44] M. Norouzi, M. Abdoos, and A. L. Bazzan, "Experience classification for transfer learning in traffic signal control," *The Journal of Supercomputing*, vol. 77, pp. 780–795, 2021.

[45] L. Torrey and M. Taylor, "Teaching on a budget: Agents advising agents in reinforcement learning," in *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, 2013, pp. 1053–1060.

[46] C. Zhu, H.-F. Leung, S. Hu, and Y. Cai, "A q-values sharing framework for multi-agent reinforcement learning under budget constraint," *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, vol. 15, no. 2, pp. 1–28, 2021.

[47] Y. Liang and B. Li, "Parallel knowledge transfer in multi-agent reinforcement learning," *arXiv preprint arXiv:2003.13085*, 2020.

[48] F. L. Da Silva, R. Glatt, and A. H. R. Costa, "Simultaneously learning and advising in multiagent reinforcement learning," in *Proceedings of the 16th conference on autonomous agents and multiagent systems*, 2017, pp. 1100–1108.

[49] A. Taylor, I. Dusparic, M. Guériau, and S. Clarke, "Parallel transfer learning in multi-agent systems: What, when and how to transfer?" in *2019 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2019, pp. 1–8.

[50] W. Liu, L. Dong, J. Liu, and C. Sun, "Knowledge transfer in multi-agent reinforcement learning with incremental number of agents," *Journal of Systems Engineering and Electronics*, vol. 33, no. 2, pp. 447–460, 2022.

[51] S. G. Subramanian, M. E. Taylor, K. Larson, and M. Crowley, "Multi-agent advisor q-learning," *Journal of Artificial Intelligence Research*, vol. 74, pp. 1–74, 2022.

[52] E. Ilhan, J. Gow, and D. Perez-Liebana, "Teaching on a budget in multi-agent deep reinforcement learning," in *2019 IEEE Conference on Games (CoG)*. IEEE, 2019, pp. 1–8.

[53] M. Gerstgrasser, T. Danino, and S. Keren, "Selectively sharing experiences improves multi-agent reinforcement learning," 2022.

[54] J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*, 1st ed. USA: Addison-Wesley Longman Publishing Co., Inc., 1999.

[55] T. Chu, J. Wang, L. Codecà, and Z. Li, "Multi-agent deep reinforcement learning for large-scale traffic signal control," *IEEE Transactions on Intelligent Transportation Systems*, vol. 21, no. 3, pp. 1086–1095, 2020.

[56] A. Das, T. Gervet, J. Romoff, D. Batra, D. Parikh, M. Rabbat, and J. Pineau, "Tarmac: Targeted multi-agent communication," in *International Conference on Machine Learning*.   PMLR, 2019, pp. 1538–1546.

[57] J. Jiang and Z. Lu, "Learning attentional communication for multi-agent cooperation," *Advances in neural information processing systems*, vol. 31, 2018.

[58] S. Kar, J. M. Moura, and H. V. Poor, "QD-learning: A collaborative distributed strategy for multi-agent reinforcement learning through consensus + innovations," *IEEE Transactions on Signal Processing*, vol. 61, no. 7, pp. 1848–1862, 2013.

[59] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *The Journal of Machine Learning Research*, vol. 21, no. 1, pp. 7234–7284, 2020.

[60] P. Sunehag, G. Lever, A. Gruslys, W. M. Czarnecki, V. Zambaldi, M. Jaderberg, M. Lanctot, N. Sonnerat, J. Z. Leibo, K. Tuyls *et al.*, "Value-decomposition networks for cooperative multi-agent learning," *arXiv preprint arXiv:1706.05296*, 2017.

[61] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[62] M. Matta, G. C. Cardarilli, L. Di Nunzio, R. Fazzolari, D. Giardino, M. Re, F. Silvestri, and S. Spanò, "Q-rts: a real-time swarm intelligence based on multi-agent q-learning," *Electronics Letters*, vol. 55, no. 10, pp. 589–591, 2019.

[63] M. Tan, "Multi-agent reinforcement learning: Independent vs. cooperative agents," in *Proceedings of the tenth international conference on machine learning*, 1993, pp. 330–337.

[64] S. Wang, J. Duan, D. Shi, C. Xu, H. Li, R. Diao, and Z. Wang, "A data-driven multi-agent autonomous voltage control framework using deep reinforcement learning," *IEEE Transactions on Power Systems*, vol. 35, no. 6, pp. 4644–4654, 2020.

[65] M. Moradi, "A centralized reinforcement learning method for multi-agent job scheduling in grid," in *2016 6th International Conference on Computer and Knowledge Engineering (ICCKE)*.   IEEE, 2016, pp. 171–176.

[66] G. Chen, "A new framework for multi-agent reinforcement learning–centralized training and exploration with decentralized execution via policy distillation," *arXiv preprint arXiv:1910.09152*, 2019.

[67] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[68] A. Khan, C. Zhang, D. D. Lee, V. Kumar, and A. Ribeiro, "Scalable centralized deep multi-agent reinforcement learning via policy gradients," *arXiv preprint arXiv:1805.08776*, 2018.

[69] J. K. Gupta, M. Egorov, and M. Kochenderfer, "Cooperative multi-agent control using deep reinforcement learning," in *Autonomous Agents and Multiagent Systems: AAMAS 2017 Workshops, Best Papers, São Paulo, Brazil, May 8-12, 2017, Revised Selected Papers 16*.   Springer, 2017, pp. 66–83.

[70] Y. Burda, H. Edwards, A. Storkey, and O. Klimov, "Exploration by random network distillation," *arXiv preprint arXiv:1810.12894*, 2018.

[71] A. White, J. Modayil, and R. S. Sutton, "Surprise and curiosity for big data robotics," in *Workshops at the Twenty-Eighth AAAI Conference on Artificial Intelligence*, 2014.

[72] M. Chevalier-Boisvert, B. Dai, M. Towers, R. de Lazcano, L. Willems, S. Lahlou, S. Pal, P. S. Castro, and J. Terry, "Minigrid & miniworld: Modular & customizable reinforcement learning environments for goal-oriented tasks," *CoRR*, vol. abs/2306.13831, 2023.

[73] M. Hausknecht, P. Mupparaju, S. Subramanian, S. Kalyanakrishnan, and P. Stone, "Half field offense: An environment for multiagent learning and ad hoc teamwork," in *AAMAS Adaptive Learning Agents (ALA) Workshop*, vol. 3. sn, 2016.

[74] P. A. Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner, "Microscopic traffic simulation using sumo," in *The 21st IEEE International Conference on Intelligent Transportation Systems*. IEEE, 2018. [Online]. Available: https://elib.dlr.de/124092/

[75] A. Castagna and I. Dusparic, "Multi-agent transfer learning in reinforcement learning-based ride-sharing systems," *arXiv preprint arXiv:2112.00424*, 2021.

[76] M. L. Puterman, "Markov decision processes," *Handbooks in operations research and management science*, vol. 2, pp. 331–434, 1990.

[77] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artificial intelligence*, vol. 101, no. 1-2, pp. 99–134, 1998.

[78] V. Konda and J. Tsitsiklis, "Actor-critic algorithms," *Advances in neural information processing systems*, vol. 12, 1999.

[79] C. J. Watkins and P. Dayan, "Q-learning," *Machine learning*, vol. 8, pp. 279–292, 1992.

[80] S. Schaal and C. G. Atkeson, "Robot juggling: implementation of memory-based learning," *IEEE Control Systems Magazine*, vol. 14, no. 1, pp. 57–71, 1994.

[81] S. Mahadevan and J. Connell, "Automatic programming of behavior-based robots using reinforcement learning," *Artificial intelligence*, vol. 55, no. 2-3, pp. 311–365, 1992.

[82] C. Gaskett *et al.*, "Q-learning for robot control," 2002.

[83] M. J. Matarić, "Reinforcement learning in the multi-robot domain," *Robot colonies*, pp. 73–83, 1997.

[84] P. Thiam, V. Kessler, and F. Schwenker, "A reinforcement learning algorithm to train a tetris playing agent," in *Artificial Neural Networks in Pattern Recognition: 6th IAPR TC 3 International Workshop, ANNPR 2014, Montreal, QC, Canada, October 6-8, 2014. Proceedings 6*. Springer, 2014, pp. 165–170.

[85] I. Erev and A. E. Roth, "Predicting how people play games: Reinforcement learning in experimental games with unique, mixed strategy equilibria," *American economic review*, pp. 848–881, 1998.

[86] M. Gueriau and I. Dusparic, "Samod: Shared autonomous mobility-on-demand using decentralized reinforcement learning," in *The 21st IEEE International Conference on Intelligent Transportation Systems*, 2018.

[87] G. Brockman, V. Cheung, L. Pettersson, J. Schneider, J. Schulman, J. Tang, and W. Zaremba, "Openai gym," *arXiv preprint arXiv:1606.01540*, 2016.

[88] 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.

[89] G. Dulac-Arnold, D. Mankowitz, and T. Hester, "Challenges of real-world reinforcement learning," *arXiv preprint arXiv:1904.12901*, 2019.

[90] W. S. McCulloch and W. Pitts, "A logical calculus of the ideas immanent in nervous activity," *The bulletin of mathematical biophysics*, vol. 5, pp. 115–133, 1943.

[91] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning representations by back-propagating errors," *nature*, vol. 323, no. 6088, pp. 533–536, 1986.

[92] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad, "State-of-the-art in artificial neural network applications: A survey," *Heliyon*, vol. 4, no. 11, p. e00938, 2018.

[93] X. Glorot, A. Bordes, and Y. Bengio, "Deep sparse rectifier neural networks," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. JMLR Workshop and Conference Proceedings, 2011, pp. 315–323.

[94] G. E. Hinton, S. Osindero, and Y.-W. Teh, "A fast learning algorithm for deep belief nets," *Neural computation*, vol. 18, no. 7, pp. 1527–1554, 2006.

[95] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[96] J. S. Bridle, "Probabilistic interpretation of feedforward classification network outputs, with relationships to statistical pattern recognition," in *Neurocomputing: Algorithms, architectures and applications*. Springer, 1990, pp. 227–236.

[97] M. D. Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[98] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[99] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited on*, vol. 14, no. 8, p. 2, 2012.

[100] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *nature*, vol. 521, no. 7553, pp. 436–444, 2015.

[101] V. Mnih, K. Kavukcuoglu, D. Silver, A. Graves, I. Antonoglou, D. Wierstra, and M. Riedmiller, "Playing atari with deep reinforcement learning," *arXiv preprint arXiv:1312.5602*, 2013.

[102] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski *et al.*, "Human-level control through deep reinforcement learning," *nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[103] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[104] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952*, 2015.

[105] Z. Wang, T. Schaul, M. Hessel, H. Hasselt, M. Lanctot, and N. Freitas, "Dueling network architectures for deep reinforcement learning," in *International conference on machine learning*. PMLR, 2016, pp. 1995–2003.

[106] 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," *arXiv preprint arXiv:1509.02971*, 2015.

[107] A. Lazaric, "Transfer in reinforcement learning: a framework and a survey," *Reinforcement Learning: State-of-the-Art*, pp. 143–173, 2012.

[108] R. Silva, M. Vasco, F. S. Melo, A. Paiva, and M. Veloso, "Playing games in the dark: An approach for cross-modality transfer in reinforcement learning," *arXiv preprint arXiv:1911.12851*, 2019.

[109] M. E. Taylor, N. K. Jong, and P. Stone, "Transferring instances for model-based reinforcement learning," in *Machine Learning and Knowledge Discovery in Databases: European Conference, ECML PKDD 2008, Antwerp, Belgium, September 15-19, 2008, Proceedings, Part II 19*. Springer, 2008, pp. 488–505.

[110] L. Torrey, T. Walker, J. Shavlik, and R. Maclin, "Using advice to transfer knowledge acquired in one reinforcement learning task to another," in *Machine Learning: ECML 2005: 16th European Conference on Machine Learning, Porto, Portugal, October 3-7, 2005. Proceedings 16*. Springer, 2005, pp. 412–424.

[111] M. E. Taylor, H. B. Suay, and S. Chernova, "Integrating reinforcement learning with human demonstrations of varying ability," in *The 10th International Conference on Autonomous Agents and Multiagent Systems-Volume 2*, 2011, pp. 617–624.

[112] M. E. Taylor and A. Borealis, "Improving reinforcement learning with human input." in *IJCAI*, vol. 328, 2018, pp. 5724–5728.

[113] T. Brys, A. Harutyunyan, H. B. Suay, S. Chernova, M. E. Taylor, and A. Nowé, "Reinforcement learning from demonstration through shaping," in *Twenty-fourth international joint conference on artificial intelligence*, 2015.

[114] Z. Wang and M. E. Taylor, "Interactive reinforcement learning with dynamic reuse of prior knowledge from human/agent's demonstration," *arXiv preprint arXiv:1805.04493*, 2018.

[115] ——, "Improving reinforcement learning with confidence-based demonstrations." in *IJCAI*, 2017, pp. 3027–3033.

[116] S.-A. Chen, V. Tangkaratt, H.-T. Lin, and M. Sugiyama, "Active deep q-learning with demonstration," *Machine Learning*, vol. 109, pp. 1699–1725, 2020.

[117] A. Nair, B. McGrew, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Overcoming exploration in reinforcement learning with demonstrations," in *2018 IEEE international conference on robotics and automation (ICRA)*. IEEE, 2018, pp. 6292–6299.

[118] M. Vecerik, T. Hester, J. Scholz, F. Wang, O. Pietquin, B. Piot, N. Heess, T. Rothörl, T. Lampe, and M. Riedmiller, "Leveraging demonstrations for deep reinforcement learning on robotics problems with sparse rewards," *arXiv preprint arXiv:1707.08817*, 2017.

[119] A. Rajeswaran, V. Kumar, A. Gupta, G. Vezzani, J. Schulman, E. Todorov, and S. Levine, "Learning complex dexterous manipulation with deep reinforcement learning and demonstrations," *arXiv preprint arXiv:1709.10087*, 2017.

[120] G. V. Cruz Jr, Y. Du, and M. E. Taylor, "Pre-training neural networks with human demonstrations for deep reinforcement learning," *arXiv preprint arXiv:1709.04083*, 2017.

[121] V. Gabriel, Y. Du, and M. E. Taylor, "Pre-training with non-expert human demonstration for deep reinforcement learning," *The Knowledge Engineering Review*, vol. 34, 2019.

[122] Y. Gao, H. Xu, J. Lin, F. Yu, S. Levine, and T. Darrell, "Reinforcement learning from imperfect demonstrations," *arXiv preprint arXiv:1802.05313*, 2018.

[123] J. Donahue, Y. Jia, O. Vinyals, J. Hoffman, N. Zhang, E. Tzeng, and T. Darrell, "Decaf: A deep convolutional activation feature for generic visual recognition," in *International conference on machine learning*. PMLR, 2014, pp. 647–655.

[124] Y. Bengio, "Deep learning of representations for unsupervised and transfer learning," in *Proceedings of ICML workshop on unsupervised and transfer learning*. JMLR Workshop and Conference Proceedings, 2012, pp. 17–36.

[125] N. Ghassemi, A. Shoeibi, and M. Rouhani, "Deep neural network with generative adversarial networks pre-training for brain tumor classification based on mr images," *Biomedical Signal Processing and Control*, vol. 57, p. 101678, 2020.

[126] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," *Advances in neural information processing systems*, vol. 26, 2013.

[127] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.

[128] Y. Seo, K. Lee, S. L. James, and P. Abbeel, "Reinforcement learning with action-free pre-training from videos," in *Proceedings of the 39th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvari, G. Niu, and S. Sabato, Eds., vol. 162. PMLR, 17–23 Jul 2022, pp. 19 561–19 579. [Online]. Available: https://proceedings.mlr.press/v162/seo22a.html

[129] A. Stooke, K. Lee, P. Abbeel, and M. Laskin, "Decoupling representation learning from reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2021, pp. 9870–9879.

[130] C. W. Anderson, M. Lee, and D. L. Elliott, "Faster reinforcement learning after pre-training deep networks to predict state dynamics," in *2015 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2015, pp. 1–7.

[131] M. Schwarzer, N. Rajkumar, M. Noukhovitch, A. Anand, L. Charlin, R. D. Hjelm, P. Bachman, and A. C. Courville, "Pretraining representations for data-efficient reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 686–12 699, 2021.

[132] A. Nair, A. Gupta, M. Dalal, and S. Levine, "Awac: Accelerating online reinforcement learning with offline datasets," *arXiv preprint arXiv:2006.09359*, 2020.

[133] Y. Lu, K. Hausman, Y. Chebotar, M. Yan, E. Jang, A. Herzog, T. Xiao, A. Irpan, M. Khansari, D. Kalashnikov *et al.*, "Aw-opt: Learning robotic skills with imitation andreinforcement at scale," in *Conference on Robot Learning*. PMLR, 2022, pp. 1078–1088.

[134] S. Lee, Y. Seo, K. Lee, P. Abbeel, and J. Shin, "Offline-to-online reinforcement learning via balanced replay and pessimistic q-ensemble," in *Conference on Robot Learning*. PMLR, 2022, pp. 1702–1712.

[135] I. Osband, C. Blundell, A. Pritzel, and B. Van Roy, "Deep exploration via boot-strapped dqn," *Advances in neural information processing systems*, vol. 29, 2016.

[136] M. Bellemare, S. Srinivasan, G. Ostrovski, T. Schaul, D. Saxton, and R. Munos, "Unifying count-based exploration and intrinsic motivation," *Advances in neural information processing systems*, vol. 29, 2016.

[137] H. Tang, R. Houthooft, D. Foote, A. Stooke, O. Xi Chen, Y. Duan, J. Schulman, F. De-Turck, and P. Abbeel, "# exploration: A study of count-based exploration for deep reinforcement learning," *Advances in neural information processing systems*, vol. 30, 2017.

[138] M. Fortunato, M. G. Azar, B. Piot, J. Menick, I. Osband, A. Graves, V. Mnih, R. Munos, D. Hassabis, O. Pietquin *et al.*, "Noisy networks for exploration," *arXiv preprint arXiv:1706.10295*, 2017.

[139] 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*, ser. Proceedings of Machine Learning Research, D. Precup and Y. W. Teh, Eds., vol. 70. PMLR, 06–11 Aug 2017, pp. 2778–2787. [Online]. Available: https://proceedings.mlr.press/v70/pathak17a.html

[140] A. A. Berryman, "The orgins and evolution of predator-prey theory," *Ecology*, vol. 73, no. 5, pp. 1530–1535, 1992.

[141] M. Chevalier-Boisvert, L. Willems, and S. Pal, "Minimalistic gridworld environment for openai gym," https://github.com/maximecb/gym-minigrid, 2018.

[142] R. S. Sutton, A. G. Barto *et al.*, "Introduction to reinforcement learning," 1998.

[143] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[144] A. O. Al-Abbasi, A. Ghosh, and V. Aggarwal, "Deeppool: Distributed model-free algorithm for ride-sharing using deep reinforcement learning," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4714–4727, 2019.

[145] J. Alonso-Mora, S. Samaranayake, A. Wallar, E. Frazzoli, and D. Rus, "On-demand high-capacity ride-sharing via dynamic trip-vehicle assignment," *Proceedings of the National Academy of Sciences*, vol. 114, no. 3, pp. 462–467, 2017. [Online]. Available: https://www.pnas.org/content/114/3/462

[146] Y. Liu and S. Samaranayake, "Proactive rebalancing and speed-up techniques for on-demand high capacity vehicle pooling," *CoRR*, 2019. [Online]. Available: http://arxiv.org/abs/1902.03374

[147] DurgeshSamant, "Larg/hfo: Robocup 2d half field offense," https://github.com/LARG/HFO/blob/master/doc/manual.pdf, Accessed: 22 May 2023.