Summer 2023

SC-MATRL: Semi-Centralized Multi-Agent Transfer Reinforcement Learning

Ayesha Siddika Nipu
Missouri State University, Nipu62@MissouriState.edu

SC-MATRL: SEMI-CENTRALIZED MULTI-AGENT TRANSFER REINFORCEMENT LEARNING

A Master's Thesis

Presented to

The Graduate College of

Missouri State University

In Partial Fulfillment

Of the Requirements for the Degree

Master of Science, Computer Science

By

Ayesha Siddika Nipu

August 2023

SC-MATRL: SEMI-CENTRALIZED MULTI-AGENT TRANSFER REINFORCEMENT LEARNING

Computer Science

Missouri State University, August 2023

Master of Science

Ayesha Siddika Nipu

ABSTRACT

Distributed decision-making in multi-agent systems (MAS) poses significant challenges for interactive behavior learning in both cooperative and competitive environments. While reinforcement learning (RL) has shown great success in single-agent domains like Checkers, Chess and Go, researchers are motivated to extend RL to MAS. However, as the number of agents increases, effectively dealing with each agent becomes increasingly complex. To mitigate the resulting complexity, a semi-centralized Multi-Agent Influence Dense Reinforcement Learning (MAIDRL) algorithm was previously developed, enhancing agent influence maps to facilitate effective multi-agent control in StarCraft Multi-Agent Challenge (SMAC) scenarios. While MAIDRL shows improved performance in homogeneous multi-agent scenarios, it struggles to make optimal decisions in complex heterogeneous systems. In this research, two major objectives are pursued: first, extending MAIDRL to improve performance in both homogeneous and heterogeneous scenarios, and second, unifying the representations in state space and action space for enabling transfer learning (TL) to leverage knowledge gained from one scenario for other unseen scenarios. To achieve the first objective, this study extends the DenseNet in MAIDRL architecture and introduces a semi-centralized Multi-Agent Dense-CNN Reinforcement Learning framework (MAIDCRL) by incorporating convolutional layers into the deep model. The results demonstrate that the CNN-enabled MAIDCRL significantly enhances learning performance and achieves a faster learning rate compared to the existing MAIDRL, particularly in more complex heterogeneous SMAC scenarios. Additionally, a novel framework is introduced to enable TL for Multi-Agent RL by unifying diverse state spaces into fixed-size inputs, allowing for a unified deep-learning policy applicable across different scenarios within MAS. Furthermore, Curriculum Transfer Learning is adopted, enabling progressive knowledge and skill acquisition through pre-designed homogeneous learning scenarios organized by difficulty levels. This approach facilitates inter- and intra-agent knowledge transfer, leading to high-performance multi-agent learning in more complex heterogeneous scenarios.

KEYWORDS: deep reinforcement learning, convolutional neural network, multi-agent system, transfer learning, curriculum learning, MAIDRL, MAIDCRL, SMAC, StarCraft II

SC-MATRL: SEMI-CENTRALIZED MULTI-AGENT TRANSFER REINFORCEMENT LEARNING

By

Ayesha Siddika Nipu

A Master's Thesis
Submitted to the Graduate College
Of Missouri State University
In Partial Fulfillment of the Requirements
For the Degree of Master of Science, Computer Science

August 2023

Approved:

Siming Liu, Ph.D., Thesis Committee Chair

Ajay K. Katangur, Ph.D., Committee Member

Mohammed Y. Belkhouche, Ph.D., Committee Member

Julie Masterson, Ph.D., Dean of the Graduate College

ACKNOWLEDGEMENTS

I would like to express my heartfelt gratitude to my esteemed Supervisor, Dr. Siming Liu for his invaluable guidance, support, and mentorship throughout the course of my thesis. His expertise, insightful feedback, and dedication have greatly contributed to the success of this work. I would also like to extend my sincere appreciation to the members of my thesis committee, Dr. Ajay K. Katangur and Dr. Mohammed Y. Belkhouche for their time, insights, and valuable input. Their constructive feedback and thoughtful suggestions have played a crucial role in shaping this thesis and improving its quality.

To my beloved parents, Badiul Alam and Shahanaz Begum, I owe an immeasurable debt of gratitude. Their sacrifices, unconditional love, and unwavering support throughout my academic journey have been constant sources of motivation for me. I would like to express my deepest appreciation to my sisters, Umma Salma and Jafrin Anika, for being pillars of mental peace during this challenging journey. Finally, I want to thank my loving husband, K M Sajjadul Islam, for his continuous support, understanding, and encouragement. His unwavering belief in my capabilities, along with his patience, has been my rock during the highs and lows of this thesis, enabling me to overcome challenges and remain focused on my goals.

I humbly dedicate this thesis to my deceased grandfather Sirajul Islam, who had a profound influence on my early career. His strong moral character and tireless work ethic have left an indelible mark on my life, shaping me into the person I am today. I am eternally grateful for the values he instilled in me. This accomplishment is a heartfelt tribute to his enduring presence in my life, and I carry his memory with me as I embark on new endeavors.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF EQUATIONS

LIST OF ABBREVIATIONS

Abbreviation	Definition
AC	Actor-Critic
A2C	Advantage Actor-Critic
AI	Artificial Intelligence
AIM	Agent Influence Map
ALE	Arcade Learning Environment
CL	Curriculum Learning
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CTL	Curriculum Transfer Learning
DCNN	Deep Convolutional Neural Network
DNN	Deep Neural Network
DP	Dynamic Programming
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
ELU	Exponential Linear Unit
GCLA	Global Critic - Local Actor
GNN	Graph Neural Network
GPU	Graphics Processing Unit
IM	Influence Map
MADDPG	Multi-Agent Deep Deterministic Policy Gradient
MAIDRL	Multi-Agent Influence Dense Reinforcement Learning
MAIDCRL	Multi-Agent Influence Dense-CNN Reinforcement Learning

MAIM	Multi-Agent Influence Map
MAIRL	Multi-Agent Influence Reinforcement Learning
MAPPO	Multi-Agent PPO
MARL	Multi-Agent Reinforcement Learning
MAS	Multi-Agent System
MATRL	Multi-Agent Transfer Reinforcement Learning
MDP	Markov Decision Process
ML	Machine Learning
MOBA	Multiplayer Online Battle Arena
NN	Neural Network
PPO	Proximal Policy Optimization
RL	Reinforcement Learning
RNN	Recurrent Neural Network
RTS	Real-Time Strategy
SARSA	State, Action, Reward, State, Action
SC2	StarCraft II
SC2LE	StarCraft II Learning Environment
SCDDPG	Semi-Centralized Deep Deterministic Policy Gradient
SL	Supervised Learning
SMAC	StarCraft Multi-Agent Challenge
STD	Standard Deviation
TCLA	Total Critic - Local Actor
TD	Temporal Difference
TF	Tensorflow
TG	Thought Game

TL	Transfer Learning
TRL	Transfer Reinforcement Learning
TRPO	Trust Region Policy Optimization
TTL	Transitive Transfer Learning
UL	Unsupervised Learning

INTRODUCTION

Artificial Intelligence (AI) has advanced significantly in many aspects of our lives in recent years. The rapid progress in AI has reached the human level or even outperformed human champions in a wide variety of tasks including autonomous driving, game playing, protein folding, and robotics [1], [2], [3]. However, most of these achievements of AI are limited to single-agent systems where interaction among agents is not considered. Since there are a large number of applications that involve cooperation and competition between multiple agents, I was interested in AI techniques that work not only on single-agent systems but also on multi-agent systems (MAS). Reinforcement Learning (RL) has emerged as a promising approach for handling MAS in recent years.

Recently, Deep Reinforcement Learning (DRL) has been considered some of the most effective AI techniques to solve problems, i.e., AlphaGo and AlphaStar [4], [3]. Extending DRL to enable interaction and communication among agents is critical to building artificially intelligent systems in multi-agent environments. One of the main challenges of Multi-Agent Reinforcement Learning (MARL) is that the canonical RL algorithms including Q-Learning and policy gradient algorithms do not generalize well to MAS due to the exponential growth of the number of states as the number of agents increases. The reason behind this issue is the necessity of mapping state-action values for each of the agents in the environment. The second challenge in MARL is that the stationary Markovian property from an individual agent's perspective no longer exists in MAS due to the dynamic activities of other agents. These non-stationary states lead to significant stability issues to MARL in the learning process. In addition, MAS themselves introduce extra cooperative and competitive learning tasks 2 to achieve team objectives for

individual agent decision-making. Deep Q-Network (DQN), an integration of Deep Neural Network (DNN) with Q-learning, resolved the issue of dimensionality by approximating the action for each state representation [5], [6].

There are numerous simulation environments for both single and muti-agent scenarios that augmented the implementation of DRL techniques. Atari games are used for learning in Arcade Learning Environment (ALE). Minecraft is used by Johnson et al. [7], likewise Vinyals et al. [8] used StarCraft II Learning Environment (SC2LE), and Samvelyan et al. [9] considered StarCraft Multi-Agent Challenge (SMAC). These versions of StarCraft are deployed in the StarCraft II (SC2) environment. Figure 1 illustrates a multi-agent scenario in the SMAC environment. Researchers are able to run hundreds or even thousands of tests concurrently when using video games, especially ones with research-oriented tools and environments like those indicated above. This becomes helpful due to the learning process followed by the RL methodology.

Figure 1. Illustration of a MAS in Built-in SMAC Scenario

A popular approach is to provide MARL with complete information for training agents, commonly named centralized learning. In centralized learning, each of the agents shares both local and global states with the train agents. Considering the availability of such information, it is avoided in multi-agent environment systems. Researchers then focused on using decentralized learning as it's not dependent on global information. However, this approach sometimes failed to match the expected outcome as it only focuses on local observations. To overcome this challenge, Foerster et al. and Lowe et al. [10], [11] proposed a hybrid approach named semi-centralized learning which undertakes hybrid centralized learning with decentralized execution of the model. In this technique, instead of focusing on the complete global state information, only the imperfect generalized scenarios are considered. By creating a two-level actor-critic network, Semi-Centralized Deep Deterministic Policy Gradient (SCDDPG) model provides both local and global information to the environment [12]. Wang et.al used global information with macro-level control in StarCraft Commander [13]. Though researchers were able to unravel the challenge, it is still questionable how to represent the global observations to make fine-grained decisions.

In 2021, Agent Influence Maps (AIM) are introduced and aggregated into a global Multi-Agent Influence Map (MAIM), which is then used in addition to local agent observations for fine-grained decision-making [9]. The influence of this semi-centralized representation defined as Multi-Agent Influence Reinforcement Learning (MAIRL) improved the performance of the system significantly. I then combined MAIRL and the DenseNet model architecture, which defined Multi-Agent Influence Dense Reinforcement Learning (MAIDRL), and evaluated MAIDRL's performance in SMAC scenarios in a real-time strategy (RTS) game, SC2 [9], [14]. By extracting a descriptive representation from the complete global information and combining it

with the DenseNet architecture, MAIDRL demonstrated a significant improvement in centralized, decentralized, and hybridized methods. In this study, I extended MAIDRL with the use of a Convolutional Neural Network (CNN) and introduced Multi-Agent Dense-CNN Reinforcement Learning (MAIDCRL) for solving MAS problems. This reformulation of extracting spatial features from MAIM by utilizing multiple CNN layers further improved the learning performance of MAIDCRL in a variety of SMAC scenarios. To evaluate how the influence map (IM) affects the multi-agent learning performance of MAIDCRL, I, therefore, performed a rigorous analysis of the agent's behavior and found several fascinating behavioral attributes determined by the agent in the testing scenarios.

Although DRL and MARL have made tremendous successes in many fields, the enormous amount of training samples and extensive learning duration significantly limit their applicability to complex problems, especially in large scale multi-agent settings. To alleviate sample complexity and accelerate the learning process of autonomous agents in tackling complex learning tasks, transfer learning (TL) has attracted much attention from researchers. TL proposes to reuse knowledge from previous tasks or external sources, such as demonstrations from humans or advice from other learning agents, to speed up the learning process. Unprincipled reuse of knowledge, however, can lead to negative transfer, making learning more difficult. To develop flexible and robust methods for autonomously reusing knowledge, TL for single-agent RL has evolved significantly to be usable in complex applications. Nevertheless, multi-agent TL approaches still require further development to find real-world applications and achieve autonomous learning.

Significant efforts have been made in developing dedicated neural networks (NN) and comprehensive training techniques to enable TL in multi-agent settings [15]. Efficiently learning

policies from scratch in multi-agent settings is difficult and time-consuming, particularly for tasks with exploration challenges. In such settings, how to unify multi-modal data encoding and decoding to enable agents' knowledge transferring and curriculum learning is essential to increase MARL performance. One attempt to enable TL for Multi-Agent Deep Deterministic Policy Gradient (MADDPG) has been made by Zhang et al. in [16] for training agents in multi-UAV combat. In that study, the input dimension of the network structure is proportional to the number of agents, representing the agent's observation of the entire environment. However, this approach can only be applied for transfer training when the number of agents is consistent. The question of enabling autonomous agents that can learn faster by reusing knowledge from various sources in MAS remains open.

To build a generalized approach in a model-free RL environment where the agents transfer the learning from a simpler scenario to a complex one, the state representations need to be unified. For StarCraft micromanagement, the traditional RL technique takes decisions that are made sequentially. To obtain the terrain's information, Shantia et al. used a fixed number of inputs depending on the number of units in each simulation [17]. The lack of cooperation and teamwork deteriorates efficiency as the models need to be trained for individual scenarios. It takes a lot of time to train a complex multi-agent scenario in the decision-making process. Though the prior model was able to take decisions collaboratively from the current action space, it cannot transfer the knowledge from one scenario to another. One probable approach should be modifying the representation of local observations in a unified way so that the knowledge could be transferred to the source without modifying the original state representation. This will bring stability to the model from the very early episodes of each iteration. This approach benefits not only in the domain of gaming, but also in other platforms such as document classification [18],

sentiment analysis [19], and other problems where machine learning-based approaches are utilized to conclude a result.

In order to address the aforementioned issues and overcome the existing challenges, I propose a novel spatial and feature encoding framework for unifying the state inputs of individual agents to the neural network along with a generalized output representation regardless of different multi-agent scenarios. I utilize a spatial abstraction technique IM, to unify various local observations into a fixed dimension in combination with abstracted global information using MAIM [14] for agents to achieve scenario-independent capability. This spatial and feature representation is aggregated to an agent's previous and current states and trained with fixed-size neural network policies preserving domain knowledge across multiple scenarios in MAS. I then conducted a rigorous analysis to evaluate the performance of my proposed TL model in different SMAC scenarios. This approach shows promising results regarding robustness and scalability among agents for intra- and inter-agent knowledge transfer.

The remainder of this thesis is partitioned as follows: the Background section introduces various concepts regarding machine learning, with emphasis on CNN and TL, while the Related Work section explores the contribution of CNN models in the broad area of AI and MARL spectrum, in addition to the current progress made in the area of transfer learning. The SIMULATION PLATFORM section provides describes the requirements of environment configuration. The Methodology section elaborates the current architecture along with the proposed model for the experiments. The Results and Discussion section subsequently presents the results from my experimentation, as well as the observed behaviors that were learned by the various considered methodologies, and the Conclusion And Future Work section draws the conclusions and suggests future works.

BACKGROUND

Machine learning (ML) is a branch of AI that involves using computational techniques to improve predictions of future information by analyzing historical data [18]. ML methods can generally be classified into three main categories, and many advanced ML techniques incorporate elements from multiple categories. These categories are supervised learning, unsupervised learning, and reinforcement learning [20]. The following subsections provide an overview of each of these categories.

Supervised Learning

Supervised Learning (SL) is a highly effective approach that involves finding a function f, which maps inputs X to corresponding target outputs Y, with the aim of minimizing cost or risk [21]. The specific cost or risk involved depends on the problem at hand and influences the quality of the resulting feature mapping. The power of SL increases as the amount of labeled data grows, as both the data and the learned function become more representative of the entire range of possible input-output combinations. This enhanced representation leads to improved generalization, even for previously unseen pairs. In addition, SL can complement RL by providing a starting point for RL agents through behavioral cloning or imitation learning [22], enabling faster and more efficient insights. While SL is widely used in classification, regression, and ranking problems, it has a significant limitation: reliance on labeled data. Although SL is the preferred method when labeled data is available, there are numerous situations where such data is not readily accessible. Figure 2 illustrates the concept of SL where the input takes the annotation along with the data and predicts the outcome based on the given information.

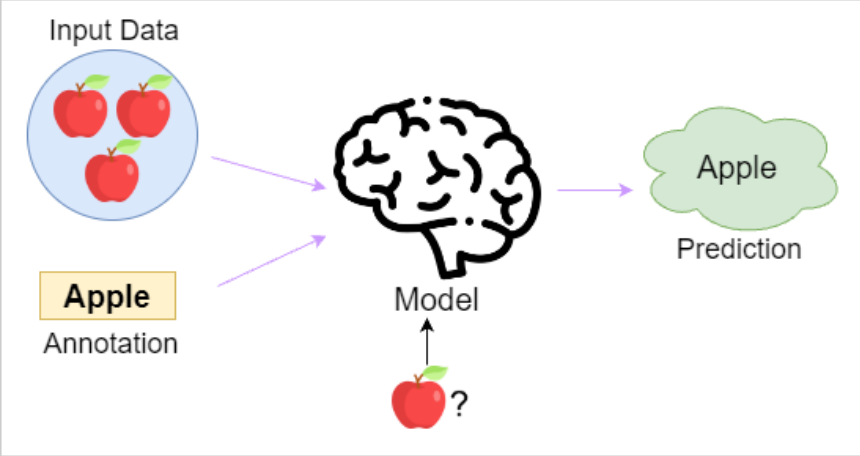

Figure 2. Illustration of Supervised Learning

Unsupervised Learning

In situations where labeled output data is unavailable for a given set of inputs, Unsupervised Learning (UL) is commonly employed as an alternative to supervised learning. UL involves using an unlabeled set of feature data to gain insights about the entire feature space and is primarily used in clustering, which is the unsupervised counterpart of categorization [20]. Clustering aims to identify similarities among subsets of inputs, grouping inputs within a cluster based on logical similarity while maintaining separation from inputs in other clusters. However, due to the inherent uncertainty in determining clusters compared to true categories, assessing the effectiveness of UL methods is challenging. Nonetheless, certain metrics like the Silhouette score have been devised to provide some indication of the quality of identified clusters [23]. Similar to SL, UL still relies on the availability of feature data, but without the need for corresponding labels. Consequently, UL is applicable only when such feature data is accessible. Figure 3 shows how UL clusters each of the group of fruits based on the similarities in their characteristics.

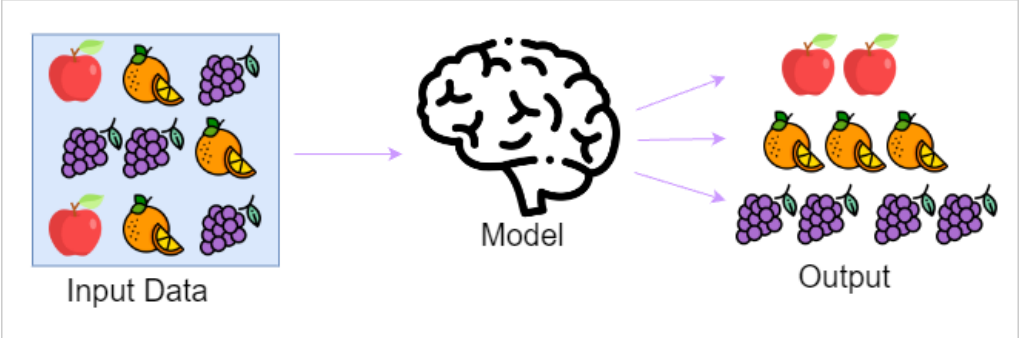

Figure 3. Illustration of Unsupervised Learning

Reinforcement Learning

RL is a distinct learning approach compared to SL and UL because it does not rely on data availability. Instead, RL is a goal-oriented methodology that learns through interacting with an environment [24]. This shifts the focus from data availability to the ability to interact with the environment and generate experiences for insights into future interactions. Unlike other machine learning methods, RL learns through evaluating the outcomes of interactions rather than aiming for a "correct" action. Figure 4 shows the learning process of reinforcement learning through trial an error. When the dog, functioning as an agent in the environment, moves, it receives a positive reward, thereby enabling it to learn the optimal action. This learning process resembles how living creatures learn, such as a child learning to walk by exploring their environment and experimenting with available tools. While other factors may influence the learning process, such as observing intelligent behavior, the fundamental concept of RL remains intact. There are three different approaches in the field of reinforcement learning: Value-Based, Policy-Based, and Model-Based. The Value-Based approach aims to maximize a value function by anticipating long-term returns of current states. In Policy-Based methods, strategies are developed to achieve maximum rewards in the future based on actions taken in each state, with two subtypes:

deterministic and stochastic. Model-Based methods involve creating a virtual model for the agent to facilitate learning in specific environments.

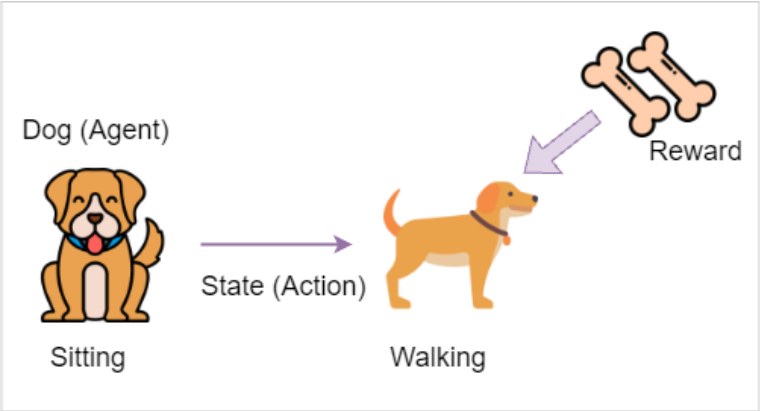

Figure 4. Illustration of Reinforcement Learning

The Fundamentals of Reinforcement Learning

In this study, I focused on RL, presenting a detailed exploration of the various components and methodologies employed in traditional RL. Figure 5 provides a comprehensive overview of the agent-environment interactions in reinforcement learning. It depicts how the agent interacts with the environment, showcasing the dynamic relationship between the two entities. The figure highlights the flow of information and actions exchanged between the agent and the environment during the learning process.

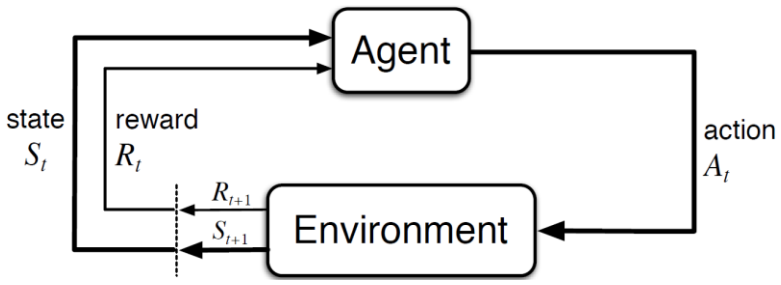

Figure 5. Illustration of The Agent-Environment Interaction in RL [22]

To illustrate these concepts, I refer to a contextual example, considering a child as the agent navigating their surrounding environment. The state variable corresponds to the observed condition of the environment at a specific time step, while the child's action is defined by their interaction with the environment based on the given state. The reward signal in this context is defined by the distance covered using bipedal means. It serves as a measure of success or achievement for the agent's actions in navigating the environment. By quantifying the distance traveled, the reward signal provides a mechanism for the agent to evaluate the effectiveness of its decisions and learn optimal strategies. Maximizing the accumulated reward over time becomes the primary objective for the agent, driving it towards actions that lead to greater distances traveled through bipedal locomotion. The learner or decision-maker acts as the agent, while encompassing all external factors such as the environment. The prevalent modeling approach for such scenarios involves the utilization of Markov Decision Processes (MDP) [24], [25], [26]. MDPs are a type of non-deterministic search problem in which the outcome of an action in a state is independent of preceding states. Each action taken by the agent at a given state leads to a numerical reward, a new state, and can be represented as a 4-tuple: (state, action, reward, state'), denoted by (s, a, r, s') [26]. The agent's behavior is determined by a policy π, which is a mapping from states to probabilities of selecting each possible action, denoted as $\pi(a|s)$. In RL methods, the policy is updated based on the agent's experience, with the overall goal being to maximize the expected return over a certain number of steps. The utility or expected value of a state is defined as the sum of discounted rewards over time. This is shown in Equation 1, where γ is the discount rate satisfying the condition $0 \leq \gamma \leq 1$, R represents the rewards received in the subsequent time steps, k denotes the number of future time steps considered, and T is the terminal time step. The discount rate determines the value of future rewards, with each future

reward being worth γ^{k-1} times its immediate value [24]. This process continues at each time step until the MDP reaches a terminal state specific to the problem, and the sequence of transitions from the initial state to the terminal state is referred to as an episode.

$$G_t = \sum_{k=0}^{T-t-1} \gamma^k R_{t+k+1} \tag{1}$$

Value functions play a crucial role in the majority of RL methodologies as they provide insights into the expected utility and potential usefulness of states or actions. These functions are defined based on the agent's policy and incorporate the discounted reward calculation G_t, as presented in Equation 1. Equation 2 and Equation 3 present the state value and action value functions corresponding to policy π, respectively [24]. It is important to highlight that the state value function is commonly known as the V value function, while the action value function is called as the Q value function.

$$V^{\pi}(s) = E[G_t|S_t = s] \tag{2}$$

$$Q^{\pi}(s,a) = E[G_t|S_t = s, A_t = a] \tag{3}$$

The Q and V value functions can be approximated using agent experience. In Monte-Carlo methods, the agent maintains average rewards for each action in each visited state, allowing Q to converge to the true action-value and V to converge to the true state-value as the number of state visits increases. Equation 4 presents an example of Monte-Carlo state value function update using an every-visit approach, where α represents the step-size parameter or learning rate. To handle large numbers of states, Q and V are often defined as parameterized

functions, tuned to better reflect the actual values. The curse of dimensionality arises from the exponential increase in possible states relative to state variables, and parameterized value functions address this challenge [27]. Parameterized value functions also introduce experience replay, a method for storing and periodically revisiting agent experiences to enhance long-term stability and prevent catastrophic forgetting. Experience replay mitigates the risk of tuning parameters based solely on recent experiences, which may deviate from earlier episodes [28].

$$V(S_t) = V(S_t) + \alpha[G_t - V(S_t)] \tag{4}$$

In certain RL methodologies, an additional component known as the model of the environment is utilized, enabling inferences about the expected behavior of the environment. This model consists of explicit transition and reward functions, denoted as $T(s, a, s')$ and $R(s, a, s')$ respectively. The transition function T represents the probability of the state s' occurring given the state-action pair (s, a) [26]. These methodologies, referred to as dynamic programming (DP), bootstrap their updates, meaning the value function updates are influenced by the current approximations of the value function in a future time step. Equation 5 illustrates this update process using the Bellman equation [25] for V as an update rule, known as iterative policy evaluation [24]. DP methods determine the utility value V of a given state S_t based on the highest expected value at the state S_{t+1}. Essentially, DP simulates the entire episode using the environmental model and a specified reward discount rate to identify the optimal course of action. Although model-based methods like DP rely on a perfect model of the environment and are computationally expensive as the environment complexity increases, they are not as widely used as their model-free counterparts.

$$V(S_t) = \max_a E[R_{t+1} + \gamma V(S_{t+1}) | S_t = s, A_t = a] \quad (5)$$

Temporal-difference (TD) learning is a foundational methodology in RL that combines aspects of Monte-Carlo and dynamic programming methods [24]. Unlike dynamic programming, TD methods do not require a perfect understanding of the environment, and unlike Monte-Carlo methods, they can update estimates online without waiting for the completion of an episode. Equation 6 illustrates the state value function update for a single-step TD method known as TD(0). The key difference between TD and Monte-Carlo methods lies in their update targets, which are $R_{t+1} + \gamma V(S_{t+1})$ and G_t, respectively.

$$V(S_t) = V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \quad (6)$$

In RL, a critical challenge is striking a balance between exploration and exploitation. RL Agents must explore the environment to discover actions that yield desirable rewards and then exploit this knowledge to maximize rewards. Two commonly used algorithms for balancing exploration and exploitation are ε-greedy and softmax. These algorithms, referred to as behavior policies, guide the agent's behavior while distinct from the agent's target policy π. Both on-policy and off-policy RL algorithms can utilize these behavioral policies. On-policy algorithms update action values considering the value of the next state given the current policy, while off-policy algorithms use a greedy approach to select the next action. In the ε-greedy algorithm, ε represents the probability of selecting a random action, and it is initially set between 0 and 1, gradually decreasing over time. When the agent does not take a random action, it selects the

action deemed optimal by the current policy. In contrast, the softmax behavioral policy leans towards exploitation by selecting actions based on their proportionate action values relative to the sum of all valid actions' values, allowing for some level of exploration. These RL methodologies, including TD learning and behavior policies, play a crucial role in addressing the challenges of exploration and exploitation in RL, enabling agents to learn and optimize their actions in complex environments.

Reinforcement Learning Algorithms

At this point, I would like to discuss about the three well-established RL algorithms: Q-Learning [29], Sarsa [30], and Policy Gradients [31]. These algorithms have significantly contributed to the progress of RL and have served as the basis or inspiration for numerous cutting-edge RL algorithms in practice. However, it should be acknowledged that when applied individually, these algorithms face limitations in scaling to Multi-Agent Systems (MAS) due to their inherent assumption of action value stationarity. This assumption assumes that the reward obtained from an action taken in the environment remains constant.

Q-Learning. The influential Q-Learning algorithm, introduced by Watkins and Dayan in 1989, represents a significant breakthrough in the field of RL [24], [29]. Operating as an off-policy TD control algorithm, Q-Learning eliminates the explicit dependence on the policy during the training process. While the policy still governs the exploration of state-action pairs, the analysis and convergence proof are considerably simplified by focusing solely on the ongoing updates of these pairs. Equation 7 presents the update logic for the action value function, which bears a striking resemblance to the original TD learning state value function update logic.

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha \left[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right] \quad (7)$$

Sarsa. Sarsa, an acronym for State, Action, Reward, State, Action [30], represents a slightly customized variant of Q-Learning. Unlike Q-Learning, Sarsa utilizes a 5-tuple instead of a 4-tuple, considering the action in time step $t + 1$ as well. In essence, Sarsa serves as the on-policy counterpart to Q-Learning, as illustrated in Equation 8. Notably, the update logic only differs in the target. The greedy action is denoted by $\max_a Q(S_{t+1}, a)$ during action selection in Q-Learning, whereas Sarsa chooses an action based on the current policy.

$$Q(S_t, A_t) = Q(S_t, A_t) + \alpha[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t)] \quad (8)$$

Policy Gradients. Policy gradient methods differ from Q-Learning and Sarsa in their focus on parameterized state and action value functions, allowing for generalizability [32]. While Q-Learning and Sarsa are tabular methods, policy gradients employ parameterized functions such as neural networks to approximate state and action values. This promotes intelligent decision-making on unseen states. In policy gradient methods, the weights of the neural network serve as the parameters being optimized. The gradients are calculated to maximize policy performance. Equation 9 demonstrates the complex gradient calculation involving reward assignment and the stationary distribution of states under the policy [31].

$$\frac{\partial \rho}{\partial \theta} = \sum_s d^\pi(s) \sum_a \frac{\partial \pi(s, a, \theta)}{\partial \theta} Q^\pi(s, a, \theta) \quad (9)$$

Transfer Reinforcement Learning

Transfer reinforcement learning (TRL) is a subfield of reinforcement learning that focuses on leveraging knowledge or experiences gained from one task or domain to improve learning and performance in another task or domain [33]. It aims to address the challenge of efficiently learning new tasks by transferring the knowledge acquired from previous related tasks. The goal is to accelerate learning, reduce sample complexity, and enhance generalization capabilities. TRL techniques involve transferring policies, value functions, or models from a source domain to a target domain, allowing the agent to benefit from prior knowledge and experiences [34]. This field has gained significant attention as it offers promising avenues for overcoming the limitations of learning from scratch and enables agents to learn more effectively in complex and diverse environments. Figure 6 provides an overview of knowledge transfer between RL methods, with the arrow-pointed section indicating the transfer strategy.

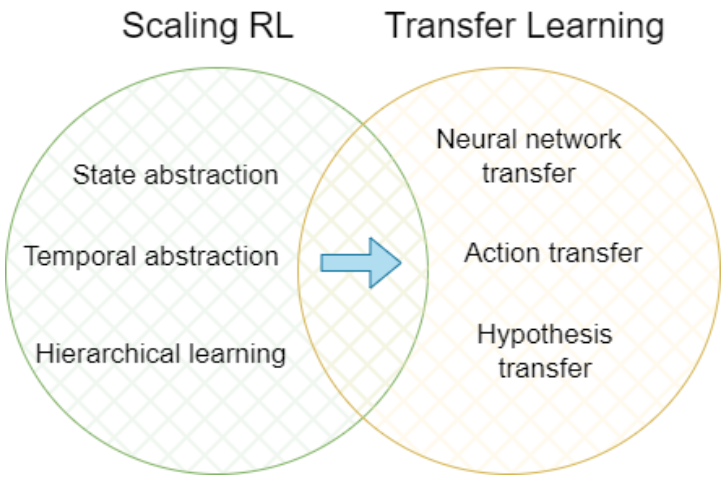

Figure 6. Transfer Between Reinforcement Learning Tasks

RELATED WORK

Extensive research has been conducted to apply various RL algorithms in controlling agents across single-agent systems, and cooperative or competitive MAS. Notably, the DQN method, an enhancement of Q-Learning, further advanced by stabilizing memory replay with prioritized replay memory sampling, facilitating more efficient network parameter tuning by retraining valuable state-action pairs, as demonstrated by Schaul et al. [35]. Simultaneously, Nair et al. introduced parallel learning using a distributed network and replay memory, effectively distributing the learning workload among multiple simulation instances to accelerate agent learning [36]. Van Hasselt, Guez, and Silver proposed the double DQN method to reduce policy over-estimation, decomposing action selection and evaluation for improved value estimations [37]. Wang et al. introduced dueling DQNs, leveraging innovative network architectures to simultaneously update state and action value function parameters, enhancing state value estimation [38]. Dong et al. presented higher sampling efficiency techniques for policy gradients, as discussed in [39]. Their approach combines model-free and model-based RL using a Gaussian process to generate supplementary off-policy samples, thereby enriching the on-policy experience pool and reducing training times.

Figure 7 demonstrates the architecture of the Actor-Critic (AC) that used TD and linear approximator to train the actors. AC has been applied and enhanced by the DRL community, with numerous variants and supplements such as deep deterministic policy gradient, which uses experience replay and target network to increase overall learning efficiency [40]. Nair et al. introduced parallel learning with a distributed network and a shared memory replay to split the learning tasks across multiple instances of simulations, effectively increasing the exploration

speed at which agents learn [36]. Extracting complex features using Network in Network architecture assembles multi-layer perceptron along with convolutional layers is introduced by Lin et al. [41].

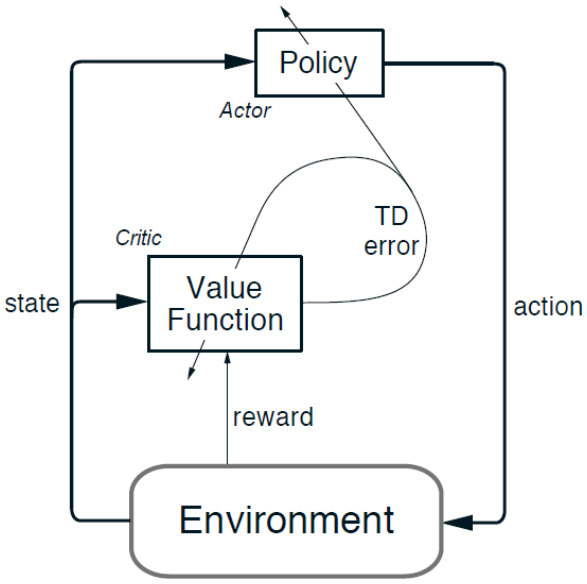

Figure 7. The Actor-Critic Architecture shown in [28]

Hierarchical Deep Q-Network, introduced by Skrynnik et al. in [42], uses a replay buffer to eliminate poor-quality training data in MineCraft. Several researchers investigated the importance of Reward Shaping, which varies rewards based on incremental steps or objectives. Ng et al. proved that these additional training rewards enhance the probability of learning in the MDP [43]. Reward Shaping has been a successful technique in DRL algorithms, although it might cause over-optimization in sparse reward environments. This issue was addressed by Huong and Ontanon bringing Action Guidance in assisting the auxiliary agents learned through the shaped reward [44]. Zambaldi et al. introduced structured perception and relational reasoning enhanced DRL to improve the efficiency, generalization capacity, and interpretability of conventional DRL [45].

Significant works on extending canonical RL from single-agent systems to MAS have been successfully published in recent years. Liu et al. modeled the problem of MARL for a constrained, partially observable MDP, where the agents need to maximize a global reward function subject to 8 both peak and average constraints. They proposed a novel algorithm, CMIX, to enable centralized training and decentralized execution under those constraints [46], [47]. Gu et al. described Multi-Agent Constrained Policy Optimization and MAPPO- Lagrangian algorithms that satisfy safety constraints in DRL where each individual agent has to not only meet its own safety constraints, but also consider those of others so that their joint behavior can be guaranteed safe [48]. Schulman et al. introduced Trust Region Policy Optimization (TRPO) in 2015 and Proximal Policy Optimization (PPO) in 2017, and both of these methods have shown the ability to effectively reduce the variance in the results [49], [50]. Berner et al. further used PPO to train multiple agents capable of defeating top human players in the popular Multiplayer Online Battle Arena (MOBA) game Dota 2 [51]. Yu et al. extended the PPO into Multi-Agent PPO (MAPPO) specializing for multi-agent settings [52]. Their work showed that MAPPO achieved strong performance in three popular multi-agent testbeds with minimal hyperparameter tuning and without any domain specific algorithmic modifications or architectures. The research based on the use and improvement of PPO and MAPPO is distinct from my work as PPO's focus is on improving the way in which models learn by clipping the gradient, while my focus is on state representation and global information extraction.

A number of research has been conducted to explore the integration of CNNs into DRL applications. Stanescu et al. introduced a CNN for evaluating game states in real-time strategy games, incorporating spatial relations between units alongside material-based evaluations [53]. Naoki Kondo and Kiminori Matsuzaki developed a Deep Convolutional Neural Network

(DCNN) that employed supervised learning with multiple layers of convolutional networks, achieving remarkable scores [54]. CNNs have also been applied in various domains, such as checking electricity prices in a multi-microgrid cooperative system [55] and mimicking Go experts [56]. These applications primarily focused on CNN implementation, while my experiment integrates CNN with DenseNet for state representation and information abstraction. Numerous research has been performed using RL algorithms to train collaborative agents to achieve team goals in MAS environments. In multi-agent systems with large state spaces, determining suitable action strategies becomes challenging due to complex learning policies. Researchers have addressed this issue by leveraging information from other training models. However, Silva et al., in a recent survey, highlighted a potential flaw in transfer learning strategies, as the assumption of consistent knowledge during training may not hold in real-world applications where agents lack perfect task understanding [15]. Consequently, even with prior knowledge from previous tasks, agents must still explore the environment to learn the optimal policy. To address this, Koga et al. proposed aggregating multiple policies into a single stochastic abstracted representation, which performs well in multi-agent scenarios but encounters challenges in generalizing information and selecting the optimal policy, particularly when the amount of information is extensive [57]. Scaling issues arise, necessitating further investigation into feature extraction before policy generalization. Mahmud and Ray introduced the concept of negative transfer, utilizing conditional Kolmogorov complexity and the Bayesian framework to identify correlations [58]. Taylor et al. emphasized the reliance on standard metrics tuned during the training phase to determine the optimal knowledge transfer cautioning against arbitrary information selection due to potential learning biases [59]. Jason et al. proposed a methodology to measure the transferability of characteristics at individual layers of a neural network,

providing insights into the level of generalization [60]. Chen et al. introduced the Net2Net technique for knowledge transfer, utilizing weighted values in the input to preserve network function [61]. In contrast, my approach involves incorporating an additional state transformation to the input states without altering the neural network structures.

Xu et al. proposed a novel approach that combines Graph Neural Network (GNN) with RL algorithms for multi-agent combat problems, achieving expeditious convergence in StarCraft by utilizing a generalized input state representation with GNN [62]. Khan et al. developed a transformer-based neural architecture for global state representation and build order prediction, addressing biases in Recurrent Neural Networks (RNN) and demonstrating the advantages of transformers with positional encoding input for the decoder [63]. However, this approach faced limitations in handling parallel loading due to dataset constraints. Tan et al. introduced the Transitive Transfer Learning (TTL) framework, focusing on knowledge transfer between domains through the identification of suitable intermediate domains, with effectiveness influenced by domain difficulty and distance [64]. In contrast, my research concentrates on knowledge transfer within the same domain, specifically addressing similar problems. Liu et al. presented the Thought Game (TG) abstract forward model for transfer learning, achieving significant success against StarCraft's level-10 AI [65]. My work shares a similar goal with Shao et al.'s gradient-based SARSA algorithm, incorporating the agent's local and abstracted global information in the state space, as well as specific move actions in the action space [66]. To address the challenge, I define a uniform state representation of local observations combined with an abstracted global state, regardless of the number of agents present. Furthermore, the agent's previous and current state information is integrated into the multi-agent training process to enhance decision-making.

SIMULATION PLATFORM

SMAC is built upon the StarCraft II Learning Environment [8] and offers a diverse range of challenging scenarios for multi-agent micromanagement. The objective in these scenarios is to eliminate opponents controlled by the built-in StarCraft AI. My study specifically focuses on decentralized decision-making and micromanagement of a team of individual agents who share a common interest, rather than centralized macromanagement of the entire army. Each agent within the environment is treated as a distinct entity with independent actions, not constrained by group commands. SMAC provides both homogeneous and heterogeneous scenarios, featuring symmetric and asymmetric distribution of agents within each group. Though the distribution of scenarios is static and cannot be customized during gameplay, this does not limit the experimentation as SMAC provides a wider range of scenarios from simple to complex ones. Each episode in SMAC provides observations, state rewards, and accepts a list of actions, making it suitable for applying RL algorithms. The naming convention of SMAC scenarios indicates the number of active agents in each team and the type of these agents for instance: marines as m, stalkers as s, zealots as z, colossi as c [67] [68]. While choosing the scenarios, I focused on varying difficulty levels so that my experiments ensure generalizability instead of performing well in specific types of settings. One of the simplest homogeneous scenario $3m$ was chosen to test the experimented features more quickly. Then I upgraded to $8m$ and extendted the evaluation on $25m$ to evaluate intra-agent knowledge transfer among large number of agents. To enable inter-agent knowledge reuse among different types of units, I selected the SMAC provided heterogeneous scenarios such as $2s3z, 1c3s5z$.

Figure 8 displays a selection of heterogeneous SMAC scenarios. These scenarios can be effectively represented using a Markov game framework, which extends MDPs to accommodate multiple agents [69], [10]. In a Markov game with N agents, there exists a set of states S that encapsulates the properties of both the agents and the environment. Additionally, there are sets of actions $A_1, A_2, \ldots, A_N$ and observations $O_1, O_2, \ldots, O_N$ for each of the S agents. Each agent considers the surrounding units as part of its local information and takes an action in the environment based on its own observation. The transition from state S to state S' is denoted by $(S \times A_1 \times \ldots \times A_N \rightarrow S')$, where each agent follows a policy π to determine its action at each step in the environment.

(a) 2c_vs_64zg (b) 8m

Figure 8. Sample SMAC Scenarios [9]

Two types of observation spaces are considered in this study: local and global. The local observation space represents a fully decentralized perspective, where each agent can only access information within its local vicinity. On the other hand, global observation space provides a fully centralized view of the environment. In the local observation space, the attributes are represented as a one-dimensional vector for both allied and enemy units within the agent's sight range, which is set to 9 by the SMAC environment. These attributes include distance, relative x and y coordinates, health, and unit type. The distance is calculated as the Euclidean distance between

the observing agent and the observed agent, while the relative x and y coordinates indicate the directional difference between the two agents. The health and unit type refer to the observed agent's health status and type, respectively. In the global observation space, information about all units on the map is provided, including their relative positions to the center of the map. This perspective also includes all the features from the local observations. Additionally, the global observation incorporates the cooldown and previous action for each unit. The cooldown represents the time interval that a unit must wait after attacking before it can attack again. All features in both the local and global observation spaces are normalized by their respective maximum values. No noise simulation is conducted in the SMAC environment, and it is assumed that the provided features accurately represent the true feature values. The available actions for living units are a discretized subset of the full action set in SC2. These actions include moving in the north, south, east, or west direction, attacking an enemy by identifier, and stopping. Defeated units are restricted to the no-op action and are not visible to surviving units. It is important to note that units in SMAC can only perform attack actions against agents within their shooting range, which is set to 6 [9]. The sight and shooting ranges are depicted in Figure 9, where the cyan and red circles represent sight and shooting ranges, respectively.

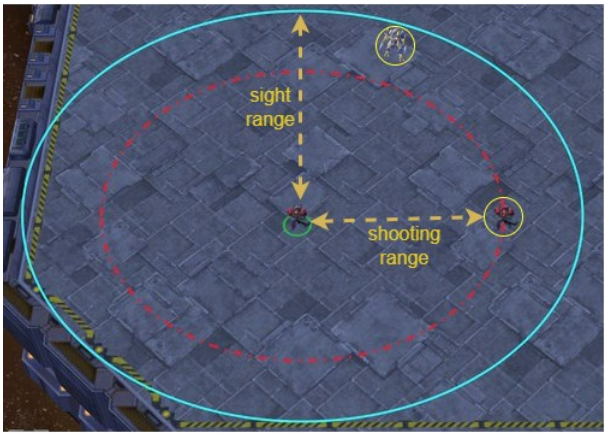

Figure 9. Single Agent's Sight Range in SMAC Scenario

In SMAC, the achievable reward within a single episode is normalized to a range of 0 to 20 [9]. By default, negative rewards are ignored. The reward is distributed to the entire team, considering various factors such as the damage inflicted on enemy units, points obtained from unit eliminations, and a bonus for achieving victory. It is important to note that the reward is not individually assigned to each agent, but rather shared among the team. Equation 10 presents the computation for the shared reward in a full game episode. Here, *t* represents the current environmental step, *T* denotes the terminal step, n is the identifier of the agent, *N* represents the maximum number of agents, D_n indicates the damage inflicted by agent *n* on enemy units at step *t*, *k* signifies the number of defeated enemies at step *t*, R_{max} represents the maximum possible reward, and *w* takes the value of 1 for a win and 0 for a loss. The episode reward is computed at the end of the game, and a discounted reward with a decay factor of 0.9 is employed for training the RL models.

$$R = 20 \times \frac{\sum_{t=1}^{T}(\sum_{n=1}^{N}(D_n) \times 10k) + 200w}{R_{max}} \tag{10}$$

SMAC offers diverse scenarios that feature units from the three races in StarCraft II: Terran, Zerg, and Protoss. The Terran race represents a human faction known for its equilibrium between technological and biological units. On the other hand, the Zerg and Protoss races represent the opposite ends of the spectrum, with the Zerg being a purely biological race that emphasizes evolution and metamorphosis, while the Protoss are highly technologically advanced and rely less on biological units. In this research, the focus is exclusively on Marines for homogeneous scenarios, and zealots and stalkers for heterogeneous scenarios which are medium-ranged infantry units belonging to the Terran race.

The SMAC combat scenarios utilized for training and evaluation purposes are shown Figure 10, Figure 11, Figure 12, and Figure 13. These scenarios include $3m$, $8m$, $25m$, and $2s3z$, respectively. The baseline scenario chosen for experimentation is the $8m$ scenario, where each team controls eight marines engaged in combat against each other. This scenario provides a suitable foundation for the research as it represents a medium-scale MARL scenario with a moderate number of units. The other selected scenarios are used to demonstrate the generalizability of the evaluated methods. Homogeneous scenarios involve only Marines, with the number of Marines per team indicated in the scenario name whereas heterogeneous scenarios involve equal number of Stalkers and Zealots in each side. Scenarios with a single number are symmetric, while the others are asymmetric, favoring the built-in SC2 game AI. The difficulty level of the AI is set to 7 in SMAC, which is equivalent to the Elite level in SC2. This poses a significant challenge to players who are unaware of the AI's aggressive yet predictable behavior. The objective for the agents in each scenario is to defeat the built-in SC2 game AI by eliminating all enemy units without losing all their allied troops. While this task may seem straightforward, it requires a deep understanding of the behaviors and capabilities of the agents on each team. Cooperation among allied agents is crucial for the team's success. They must work together, focusing their attacks on enemies, to efficiently minimize damage received and maximize damage dealt over time, thereby minimizing casualties sustained.

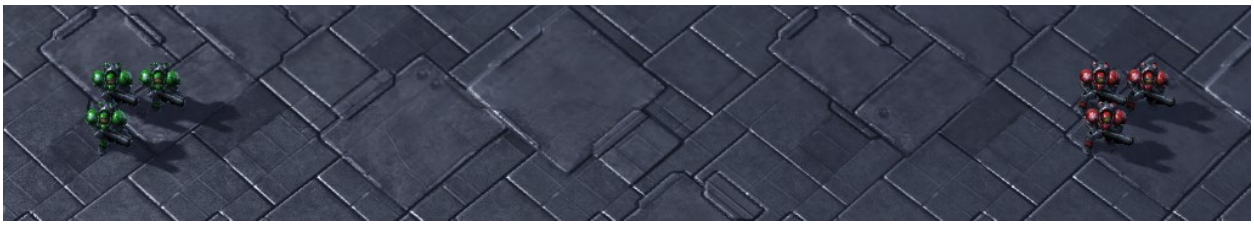

Figure 10. $3m$ SMAC Scenario with 3 Marines on Each Group

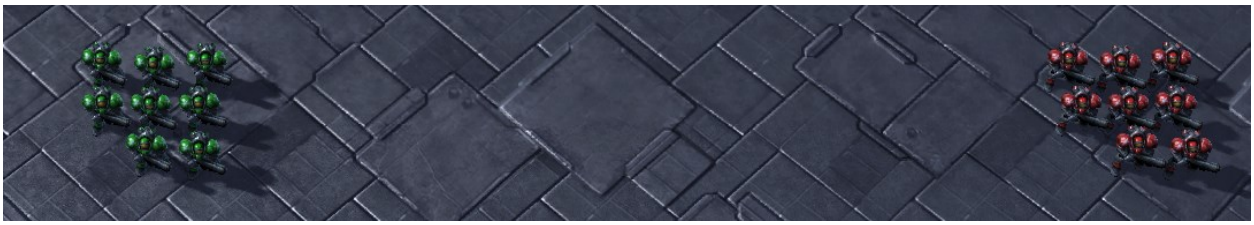

Figure 11. $8m$ SMAC Scenario with 8 Marines on Each Group

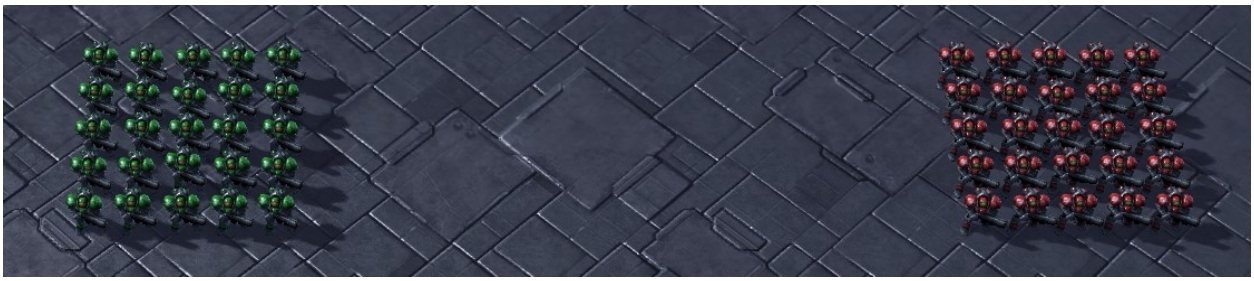

Figure 12. $25m$ SMAC Scenario with 25 Marines on Each Group

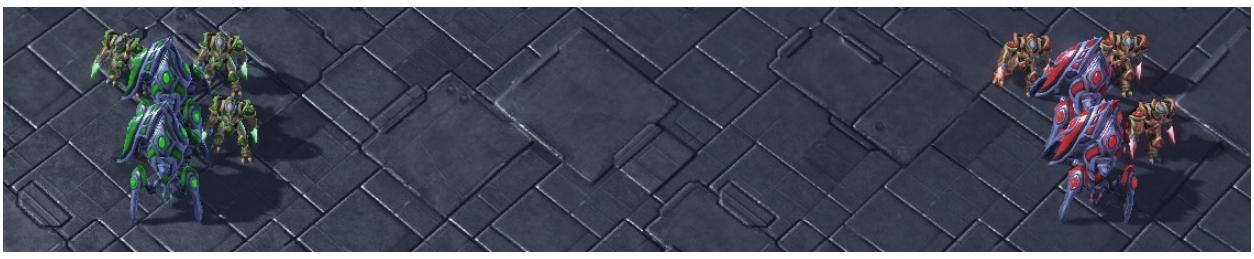

Figure 13. $2s3z$ SMAC Scenario with 2 Stalkers and 3 Zealots on Each Group

METHODOLOGY

All experiments were conducted using Python 3.7.3 with NumPy (NP) 1.19.5 and TensorFlow (TF) 2.4.0 [70] as the framework for creating and training neural networks (NNs). The SMAC 1.0.0 environment, along with SC2 version 4.10 [9], was utilized for conducting the MARL experiments. To ensure the reproducibility of results, Graphics Processing Unit (GPU) acceleration was not employed, and the models were constrained to Central Processing Units (CPU). Random generators were seeded with values ranging from 1 to 31, corresponding to the experiment number relative to the specific methodology used. Although this approach led to a moderate increase in experimental completion time, especially for CNNs with larger parameter counts, it was considered a worthwhile trade-off to ensure the reliability and reproducibility of the experiments.

Experimental Features

The proposed policies were evaluated using a diverse set of SMAC scenarios ranging from simple homogeneous to complex heterogeneous cases. To ensure robust statistical analysis, I conducted each experiment with 31 different random seeds. For statistical analysis, I assessed the performance of my model in various SMAC scenarios in which each experiment encompassed a total of 2000 episodes, taking into account the considerable number of tuning parameters embedded in my neural network architecture. To implement the neural network, I built it upon the widely-used Advantage Actor-Critic (A2C) algorithm [71]. Both the actor and critic networks were incorporated into my architecture. I utilized an ε-soft method, which combines the greedy and softmax techniques. The initial value of ε was set to 1 and gradually

decreased over time. To determine ε at a specific time step, Equation 11 was followed, where ε_0 was assigned a value of 0.0001, ε_{min} is 0.0001 and γ represented the total number of environmental steps, set to 30,000.

$$\varepsilon_t = max\left(\varepsilon_0 - \frac{t \times \varepsilon_0}{\gamma}, \varepsilon_{min}\right) \tag{11}$$

For all experiments, the A2C algorithm was incorporated, utilizing separate neural networks for the actor and critic components. The size of inputs to these networks varied depending on the specific MARL methods and combat scenarios being explored. Throughout the network layers, excluding the output layers, the Exponential Linear Unit (ELU) activation function with α=1.0 was utilized. The actor network employed softmax activation in its output layer, while the critic network used a simple linear activation. Categorical cross-entropy loss was applied to the actor, while mean squared error loss was used for the critic. Optimization of both networks was performed using the adam optimizer with a learning rate of 0.00001 [72]. Multiple layer of maxpooling and dropout were considered during the integration of CNNs [73], which have been explained in the detailed model architecture. The models have been saved using TF saved model format to store the actor and critic networks for future evaluation after the initial training phase. The networks were saved selectively, based on the condition that the running average reward at the end of an episode surpassed the previous maximum. As a result, the saved networks accurately represent the top-performing models obtained for each specific combination of seed, method, and scenario.

Advantage Actor Critic (A2C) Algorithm

In the A2C RL algorithm, the actor and critic components are essential. To address this, Harris et al. utilized two separate neural networks with an identical hidden layer architecture for MAIRL and MAIDRL experiments [14]. This A2C implementation employs a single controller responsible for individual agent control. The shared parameter approach enables faster and more intelligent learning as the parameters are updated using the collective experiences of all agents.

The A2C RL algorithm comprises two key components: the actor and the critic. To address this, I employed separate neural networks for each component, both sharing the same hidden layer architecture. Figure 14 displays the A2C implementation that involves a single controller responsible for individual agent control, enabling the utilization of shared parameters. This shared parameter approach facilitates accelerated and intelligent learning as the parameters are updated based on the collective experiences of all agents.

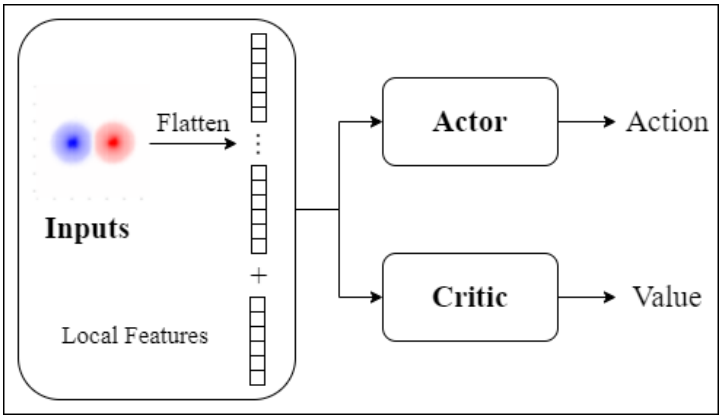

Figure 14. Advantage Actor Critic (A2C) Model Architecture

The critic component can be regarded as a model-free, value-based approximation of the state value function, as depicted in Equation 2, which estimates the expected reward for a given state based on the policy. Let $V^*(s)$ denote the optimal state value function, representing the

maximum reward achievable. The objective of the critic is to approximate $V^{\pi}(s;\theta_c)$ to closely match $V^{*}(s)$, where θ_C represents the critic's parameters. The critic parameters θ_C are updated using standard gradient ascent at the conclusion of an episode, employing the discounted reward values for a supervised update [74]. As the critic's update is crucial for calculating the advantage used in actor training, the update takes place subsequent to the advantage calculation. The actor component can be considered as a model-free, value-based approximation of the action value function which estimates the expected reward for a given action in a particular state and following the policy $\pi(a|s)$. Let $Q^{*}(s,a)$ denote the optimal action value function, representing the maximum achievable reward. The objective of the actor is to validate $Q^{\pi}(s,a;\theta_A) \approx Q^{*}(s,a)$, where θ_A represents the actor's parameters. It is worth noting that, following the A2C algorithm, $Q^{\pi}(a_t,s_t;\theta_A) - V^{\pi}(s_t;\theta_C)$ serves as a target network representing the advantage of taking one action over another. This target network is influenced by both the action value estimation determined by the actor and the state value estimation determined by the critic. At the conclusion of an episode, the actor's parameters θ_A are updated based on the gradient ascent direction specified in Equation 12.

$$\nabla_{\theta_A} log\, \pi(a_t|s_t)(Q^{\pi}(a_t,s_t;\theta_A) - V^{\pi}(s_t;\theta_C)) \tag{12}$$

The performance of the basic A2C model was enhanced through the introduction of a DenseNet-inspired model architecture for both the actor and critic components. The utilization of this DenseNet-style grouping of layers, as depicted in Figure 15, was incorporated into the proposed model design. The architecture consists of an arbitrary number of n layers, where the output of each layer is concatenated with the inputs of subsequent layers within the group. My

network architecture consists of two such groups, each comprising five dense layers with 128 neurons. Additionally, an extra dense layer with 256 neurons is included in the DenseNet architecture, positioned before the first group of DenseNet-style layers, immediately following the input layer [75].

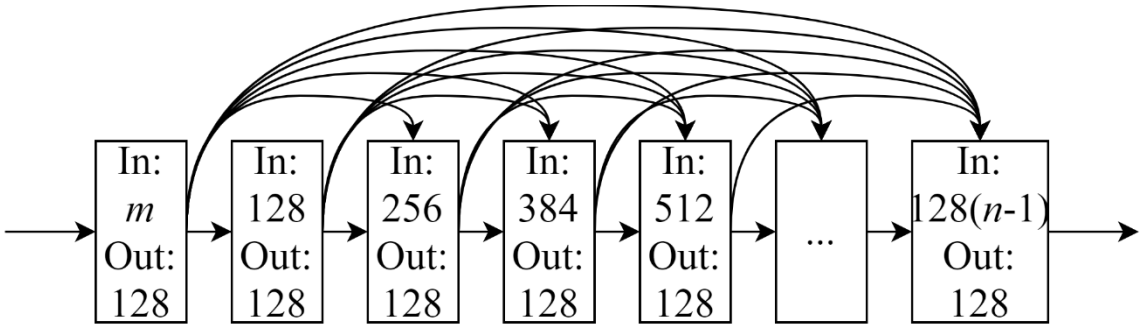

Figure 15. DenseNet-style Grouping of n 128-neuron Layers [14]

Semi-centralized Multi-Agent Influence Map

The A2C algorithm was incorporated to investigate the effective utilization of global information for decision-making in multi-agent environments across different state scenarios. The first scenario explores a centralized approach, where agents utilize both local and global observations to train the actor and critic networks. The second scenario introduces the global critic with a local actor (GCLA), where the critic receives global information while each actor only receives local observations. This unique approach decentralizes the actors while maintaining a centralized critic. In the third scenario, a total critic and local actor (TCLA) approach is explored, wherein the critic is centralized while the actors remain decentralized. This method draws inspiration from the centralized training and decentralized execution concept introduced in MADDPG. By examining these different scenarios, I aim to gain insights into the

optimal utilization of global information for effective decision-making in multi-agent environments.

In this study, I abstract global features in a manner that aligns with human interpretation. The proposed method involves the utilization of AIMs which rely on three key parameters: source influence (I_0), influence decay rate (λ_I), and range of influence (d_I) for individual units. A notable advantage of this approach is its inherent capability to dynamically represent information by adjusting the values. Specifically, for the conducted experiments, the range of influence (d_I) is set to the predefined agent range in SC2, while I_0 corresponds to the relative health value extracted from SMAC. Furthermore, λ_I is determined as the inverse of the distance from the agent, with a value of I_0 assigned when the distance is zero. To differentiate between allied and enemy units, I_0 of the allied units is negated relative to their health information obtained from SMAC. The individual AIMs are combined to create a MAIM by summing the AIM values based on the agents' positions in the environment. The AIM is aligned with the agent's position on the MAIM, and overlapping cells are added together. Any portion of the AIM that does not overlap with the MAIM is disregarded. The MAIM serves as a scaled representation of the units in the SMAC scenario map. While the SMAC scenario map has a size of 32x32 units, the dimensions of the MAIM can be customized with positive integers. The implementation automatically adjusts the AIM to fit the MAIM dimensions while preserving the scale of the agent's influence in the $8m$ scenario. MAIMs of sizes 16x16, 32x32, and 64x64 were investigated, utilizing the same AIM parameters.

Multi-Agent Influence Dense-CNN Reinforcement Learning (MAIDCRL)

In this research, I aim to investigate the effects of incorporating CNN layers on the performance of MAIM inputs in the context of MARL. To achieve this, the existing MAIDRL model architecture was enhanced by integrating multiple CNN layers that directly accept the MAIM as input. Based on the performance of MAIDRL, three dense blocks were modeled to take the input of local observation [14]. The performance of the resulting CNN-enabled model, called MAIDCRL, was then compared to the previous MAIDRL algorithm using the same set of scenarios from the SMAC environment. In the MAIDCRL architecture, each group within the dense blocks of the model incorporated the MAIM input by concatenating it with the default CNN layer provided by DenseNet architecture. Based on the parameter tuning, four blocks of Conv2D were incorporated within the dense block when the global IM passed as input parameter to the network. Additionally, two separate CNN layers were configured with 32 filters, a stride rate of 1, and a kernel size of 3. This design choice was aimed at extracting spatial features from the MAIM, allowing the model to effectively capture and utilize the relevant information contained within the influence map.

To handle negative values and promote the learning process, ELU activation function was employed in the DRL architecture of MAIDCRL. Additionally, MaxPooling with a 2x2 kernel was applied after each convolutional layer to reduce the dimensionality of the feature maps and further enhance computational efficiency. In order to improve generalization and robustness, a dropout layer was incorporated into the MAIDCRL model. Dropout rates ranging from 0.1 to 0.5 were explored to assess their impact on the overall performance of the model. To evaluate the effectiveness of the proposed MAIDCRL representation, it was compared against the previous MAIDRL version in a series of experiments. Various combinations of scenarios and

parameters were tested, and statistics were collected to assess and compare the performance of the two models. For a more detailed visual representation of the MAIDCRL architecture, refer to Figure 16, which provides a comprehensive overview of the model's structure and the flow of information within its components.

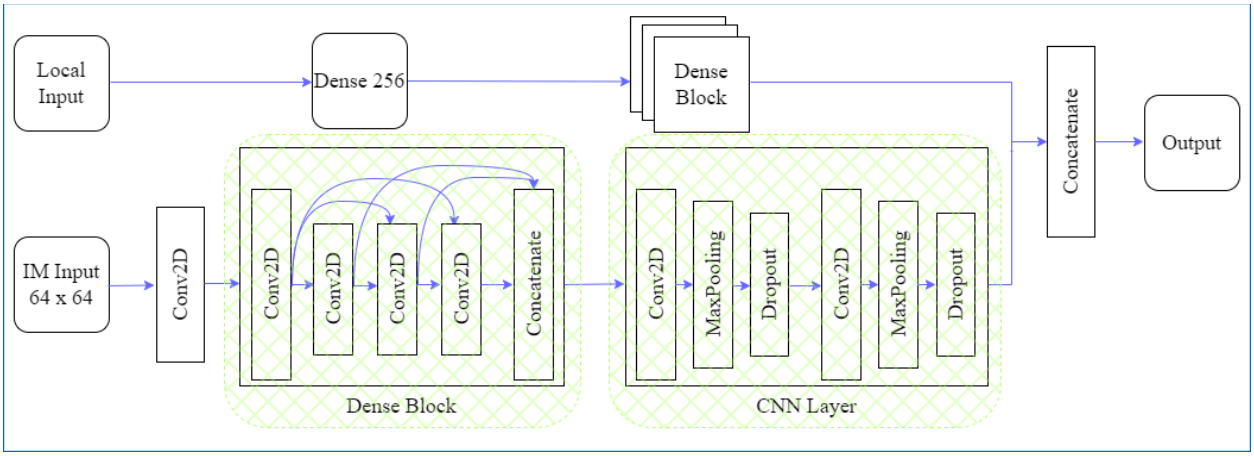

Figure 16. Outline of MAIDCRL Architecture

Multi-Agent Transfer Reinforcement Learning (MATRL)

Leveraging existing knowledge has proven to be a valuable approach in expediting the learning process of MARL and enhancing the capabilities of agents in multi-agent systems to handle complex tasks. This involves creating a mapping between a knowledge space, which encompasses samples from current and previous tasks, as well as insights gained from other agents, to a policy that guides agent behavior. However, model-free MARL techniques often necessitate a substantial number of samples to train an optimal policy, which becomes particularly challenging in scenarios with vast state and action spaces, such as those encountered in SMAC, which features diverse unit and terrain conditions. Developing effective strategies for each scenario from scratch can be time-consuming and resource-intensive.

In SMAC, the default state representation provided by the environment is tailored to reflect the number of units present in each scenario. For example, in scenarios $3m$, $8m$, and $25m$, the local state is represented by 30, 80, and 250 one-dimensional vectors, respectively, as illustrated in Figure 17. Although this default representation is compact and captures precise agent information necessary for MARL training, it poses challenges in terms of knowledge transfer across multiple scenarios due to its dependency on scenario-specific state sizes.

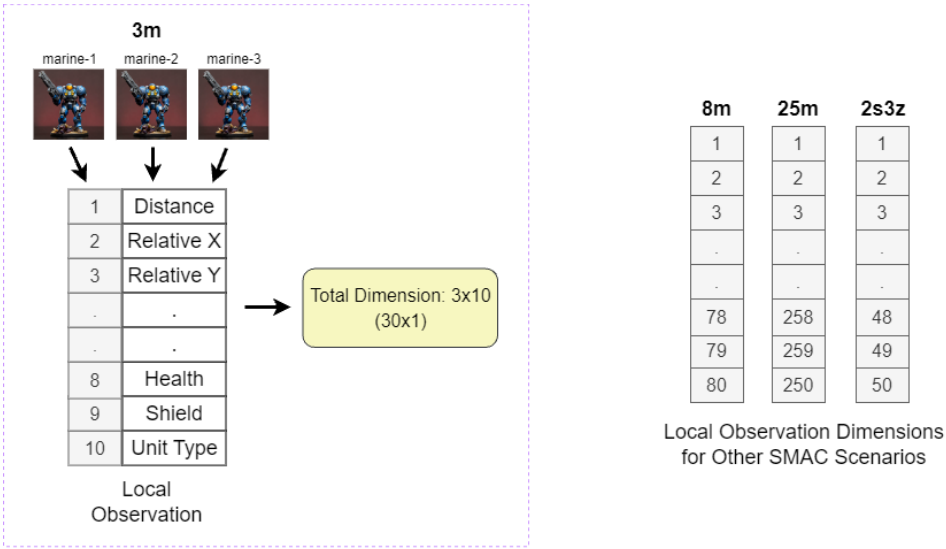

Figure 17. Scenario-dependent Local Observation Across Different SMAC Scenarios

To mitigate this limitation, my research focuses on constructing a fixed-size state representation that remains consistent irrespective of the number of agents in the scenario. This unified representation aims to overcome the hurdles associated with scenario-dependent state sizes, thereby facilitating knowledge transfer across diverse scenarios. By establishing a constant state size, I alleviate the restrictions imposed by the default representation and enable more effective utilization of existing knowledge in a generalized manner. This approach offers

potential improvements in the performance and adaptability of MARL algorithms in SMAC and similar environments.

Scenario Independent State Representation. Various approaches have been explored by researchers to address the challenge of unifying state representations in order to facilitate knowledge transfer in MARL. However, a universally accepted scenario-independent state representation has yet to be established. In this research, I propose a novel approach that leverages two types of state information, namely local observations and abstracted global information, to feed reusable neural networks across multiple scenarios in SMAC.

To extract and filter aggregated spatial representations from global information, a spatial information technique called AIM was introduced. This technique proved effective in identifying common objectives and promoting the learning of collaborative behaviors among agents. Building upon this foundation, I extend the application of AIM in this study to construct a scenario-independent local state representation. The aim is to enable effective knowledge transfer across all scenarios provided in SMAC. By integrating both local observations and abstracted global information using AIM, I aim to create a comprehensive state representation that captures both local and global aspects of the environment. This approach facilitates the transfer of learned knowledge and collaborative behaviors across diverse scenarios in SMAC, thereby promoting enhanced performance and adaptability in MARL.

My research contributes to addressing the ongoing challenge of scenario-independent state representation in MARL and offers a promising avenue for achieving effective knowledge transfer and collaboration among agents in complex multi-agent environments. For the state representation of local observation, I considered the local observations of the prior step, the actions performed in that step, along with the current step information. The local observations

include distance, relative position, health, shield, and unit type for allied and enemy units within the sight range as demonstrated in Figure 9. The default states received from SMAC depend on the number of active agents in the game environment. In order to remove the dependency on the number of agents across SMAC scenarios, I extended the use of IM from global information abstraction to local observation aggregation.

The local IM transformation yields a fixed dimension of sight range with different resolutions such as 19 × 19, 37 × 37, and 55 × 55, each with the same local state parameters. Figure 18 shows a sample heatmap with cell values transformed from a local state observation in a two-dimensional 19 × 19 matrix on $8m$ scenario. This matrix is derived from the perspective of the highlighted agent where 1 denotes the allies, and -1 indicates the enemy units. The input dimension is determined by the agent's sight range in all four directions, resulting in a unified two-dimensional scaled matrix transformed from a local observation. In SMAC, the agents' sight range is a fixed value of 9, which means the agent can see an enemy within a distance of 9 cells. The unified subset of global and local information as shown in the green-colored rectangle in Figure 19b is then flattened and propagate through the neural network, which applies to all SMAC scenarios. To unify the local observation using IMs without losing essential information, I evaluate different resolutions to construct local IMs for various levels of granularity on unit manipulations during combats in SMAC. The generated IMs will then serve as scenario-independent input parameters for the neural network displayed in Figure 19c, which carries the shared knowledge across all SMAC scenarios. This unified representation of the local states resolves the dependency of a different number of agents in the environment, and the input of fixed dimensional local state, along with the abstracted global information from MAIM, is

trained via neural networks to execute the probable attacks or actions for winning strategies in SMAC.

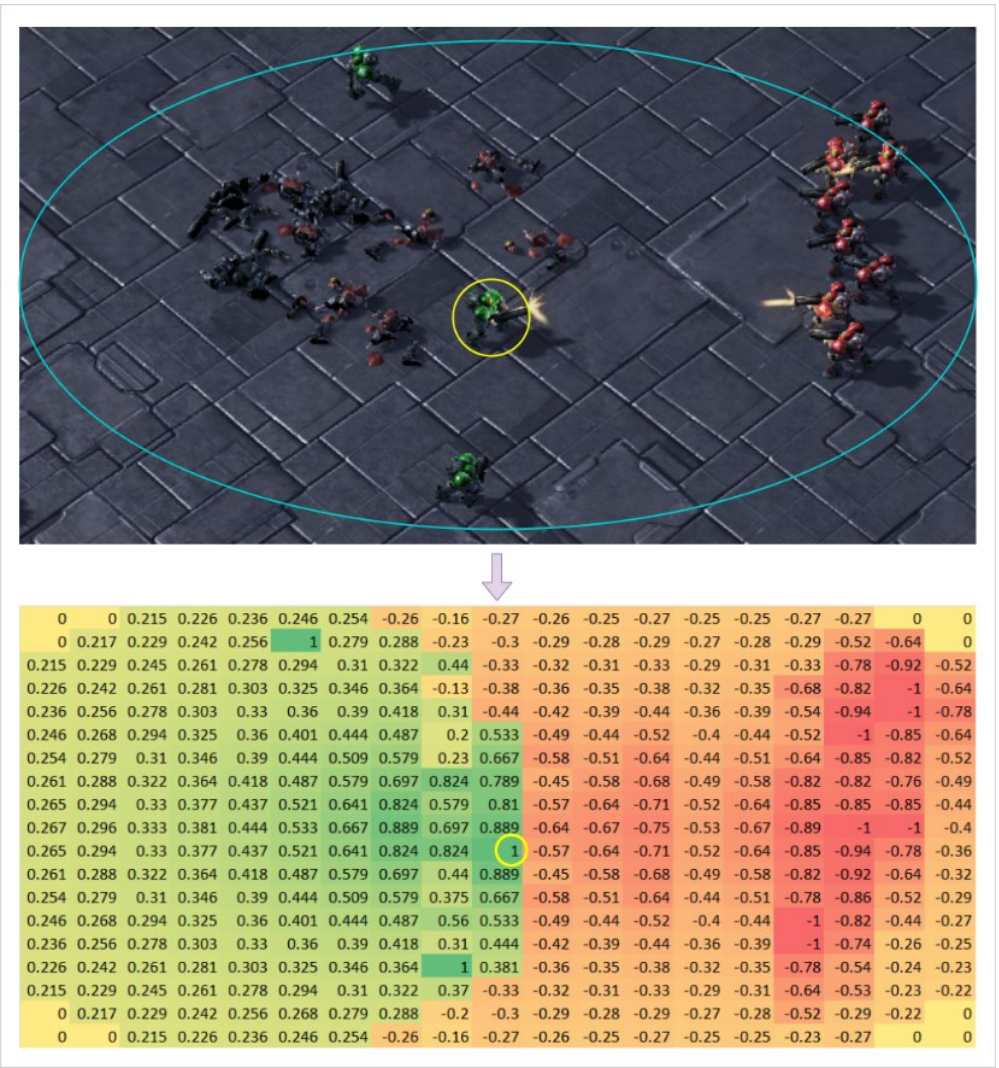

Figure 18. A Sample 19 × 19 Heatmap Generated from Local Observation on $8m$

Scenario Independent Action Representation. In the context of SMAC, agents are faced with the task of selecting actions from a finite action space based on the state information described in the preceding subsection. While move actions offer four directional choices (north, south, east, and west), the decision-making process becomes more complex for attack actions. In

the default action space provided by SMAC, agents must consider the number of enemies within their sight range in their local observation when making attacking decisions. However, the varying number of enemies across different scenarios poses a challenge. To address this challenge, I propose a generalized approach that focuses on the closest enemy position for the attacking action, eliminating the dependence on the specific number of agents within the sight range. This approach is depicted in Figure 19d, where my custom-generated policy replaces the scenario-dependent output, enabling the agent to execute attack actions without specific knowledge of enemy agents.

By training the neural networks (NNs) with detailed policies based on the spatial information of the agent's current position, targeting the closest enemy instead of relying on the default scenario-dependent action does not result in any loss of valuable information necessary for decision-making. This generalized approach facilitates the sharing of attacking policies among agents, thereby promoting knowledge transfer across a diverse range of SMAC environments. Through the formulation of a unified solution, I have achieved both improved performance in large-scale scenarios and enhanced efficiency in the simulation environment.

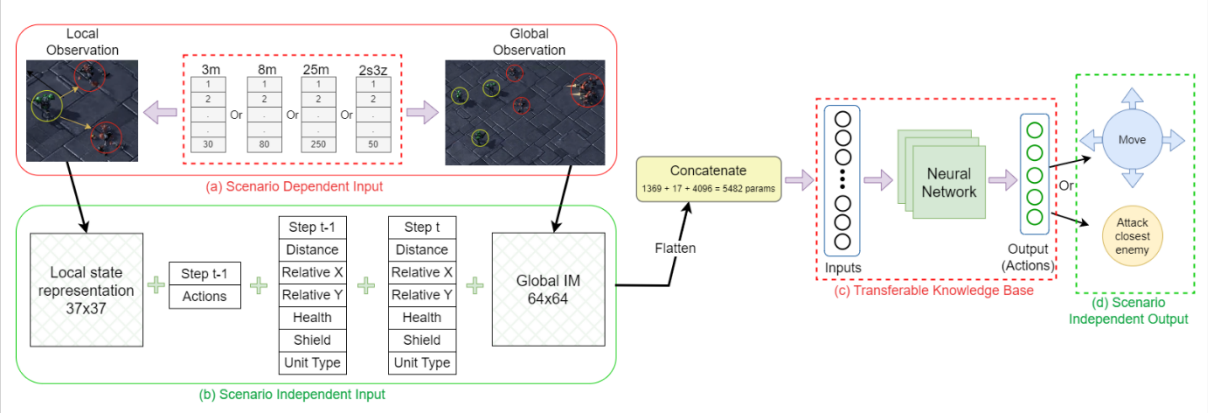

Figure 19. Transfer Learning Model Representation for Single Unit

Curriculum Transfer Learning (CTL)

Curriculum learning (CL) is a learning methodology that follows a predefined sequence of tasks based on their increasing complexity [76]. It involves training agents in a game-like environment against simulated opponents that gradually become more proficient. This approach allows the agents to develop effective strategies and acquire knowledge that can be transferred to real-world challenges [77]. In this study, I explored the effectiveness of CTL, a technique that combines curriculum learning with transfer learning to enhance the learning process and achieve improved outcomes within a fixed timeframe. My focus was on examining how transferring winning strategies, learned by Marines in scenarios $3m$ and $8m$, could positively impact the behavior of Stalkers and Zealots in a more complex and heterogeneous environment known as $2s3z$. The curriculum transfer learning process, as depicted in Figure 20, follows a sequential training approach.

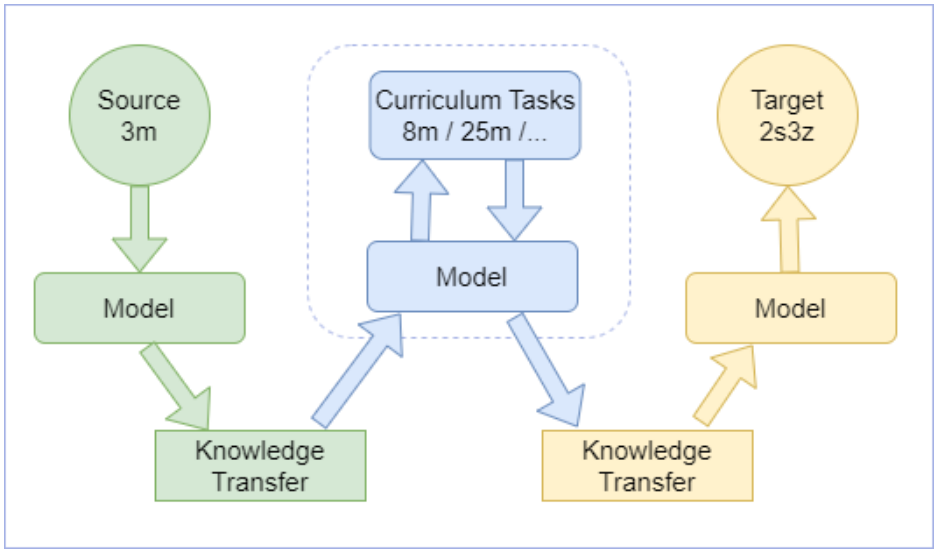

Figure 20. Curriculum Transfer Learning Architecture

I initially trained the policy using the simplest scenario, $3m$, allowing the agents to grasp fundamental concepts and develop a foundation of skills. Subsequently, I retrained the learned model in a medium-level scenario, $8m$, to further refine the acquired strategies and adapt them to more challenging situations. Finally, I leveraged the knowledge obtained from both scenarios to tackle the significantly more intricate scenario, $2s3z$, characterized by diverse unit types and the presence of heterogeneous team units that were not encountered in the previous training scenarios. By employing curriculum transfer learning, I aimed to expedite the learning process and enable the agents to effectively navigate the complexities of the $2s3z$ environment. This approach offers a practical way to transfer learned strategies across different scenarios, facilitating the application of acquired knowledge to novel and demanding situations.

RESULTS AND DISCUSSION

The performance of MARL methods is evaluated using three key metrics: the average of the running average episode reward, the standard deviation of the running average episode reward, and the percentage of seeds that achieve a maximum running average reward across the entire set of 31 seeds. These metrics are referred to as overall performance, overall stability, and peak performance, respectively. Additionally, a detailed analysis of the peak performance is conducted for each method, considering the minimum, maximum, average, and standard deviation across all seeds. For each scenario and method, I determine the best performance achieved by each seed and calculate the minimum, maximum, average, and standard deviation based on these peak performances. The corresponding tables highlight the highest minimum, maximum, and average values per scenario in bold font, along with the lowest standard deviation. Furthermore, I examine the learned behaviors exhibited by each MARL method when deploying the trained actor of the best-performing seed in each scenario. These evaluations are conducted in a headed environment without any additional training. By analyzing these metrics and behaviors, I gain insights into the effectiveness and robustness of the MARL methods assessed in this study.

MAIDCRL Learning Performance

For the evaluation, I compared the performance of my CNN-enabled MAIDCRL model with the previously developed MAIDRL model, rather than considering other baseline architectures. This choice was motivated by the demonstrated superiority of MAIDRL over other baseline models in terms of both performance and efficiency [14].

$3m$. The $3m$ scenario is a variant of the $8m$ scenario, characterized by a reduction in the number of units to three per team. This scenario was selected to assess the suitability of MAIDCRL in combat scenarios with varying levels of complexity. While it was expected that $3m$ would be easier to learn compared to $8m$, the results presented in Table 1 and Figure 21 indicate that the performance was poorer than anticipated. This discrepancy can be attributed to the disproportionate reward structure associated with winning the scenario in relation to the limited availability of incremental rewards. It is hypothesized that the distinct score distribution in $3m$, stemming from the lower number of units on the field, coupled with the reward calculation method that emphasizes aggregate damage dealt to enemies with a substantial bonus for achieving victory, contributes to the observed performance.

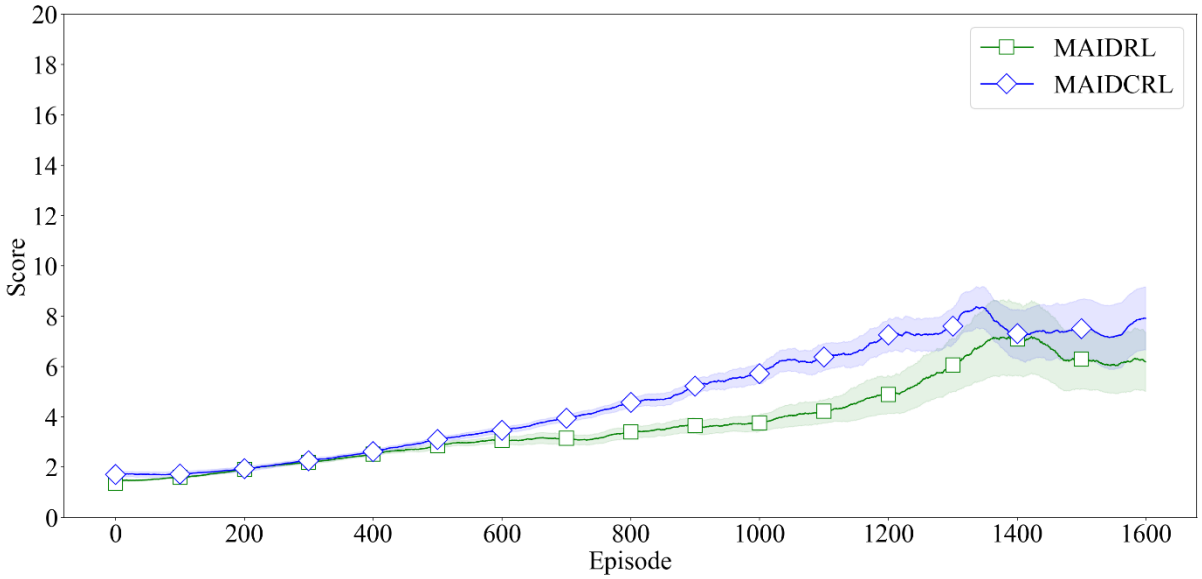

Figure 21. Average of the Running Average Episode Reward on $3m$ Scenario

$8m$. Figure 22 illustrates the performance of MAIDRL and MAIDCRL models on the $8m$ homogeneous scenario. This scenario is considered moderately challenging due to the

presence of a significant number of agents in the environment. Additionally, it is regarded as an ideal scenario since it exhibits overall stability in terms of performance. In the $8m$ scenario, MAIDCRL achieves a highest running average episode reward of 18.79, which is 10.98% higher than MAIDRL without CNN layers. Additionally, the running average improves by 5.96% in the, showcasing the efficacy of MAIDCRL. The learning speed of MAIDCRL is also faster than MAIDRL, providing further evidence of the enhanced learning process facilitated by the CNN layers.

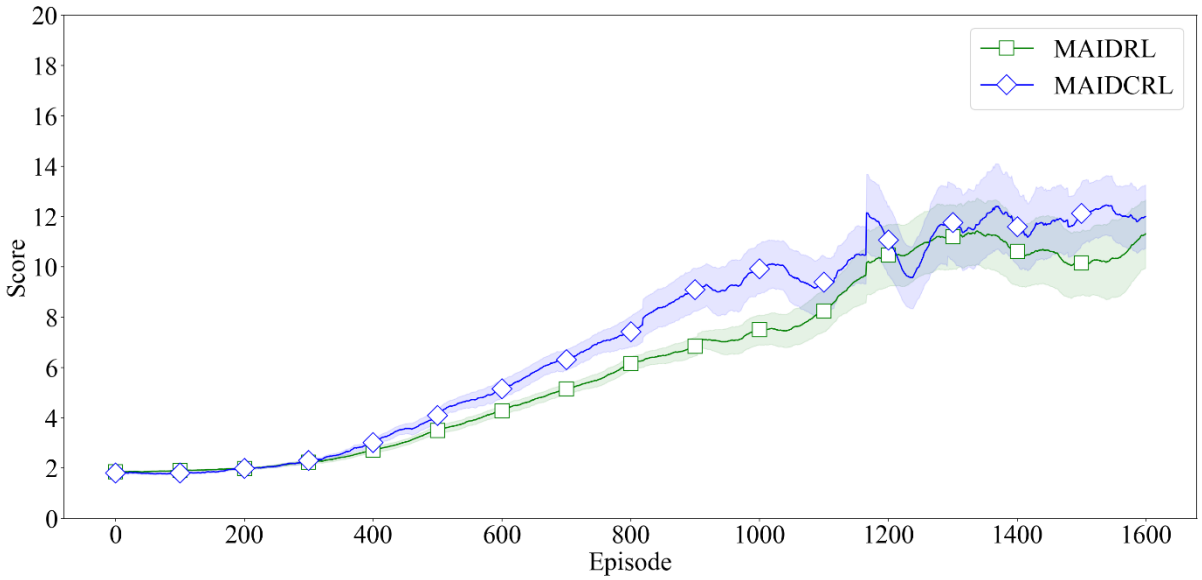

Figure 22. Average of the Running Average Episode Reward on $8m$ Scenario

$25m$. In Figure 23, the performance of MAIDCRL and MAIDRL in the $25m$ homogeneous scenario, which involves a larger number of agents, is examined. The initial learning phase reveals that MAIDRL has a slightly faster learning rate compared to MAIDCRL until approximately 600 episodes. However, beyond this point, MAIDCRL surpasses MAIDRL in terms of performance. This improvement in performance can be attributed to the learning

capabilities of MAIDCRL, as it continues to learn from the environment and adapt its strategies over time. The significant episode reward of 12.95 achieved by MAIDCRL after 1013 episodes demonstrates its ability to continuously enhance its performance in the $25m$ scenario. This suggests that MAIDCRL is capable of effectively adapting and making more informed decisions as it gains more experience in dealing with a larger number of agents in the complex homogeneous $25m$ scenario.

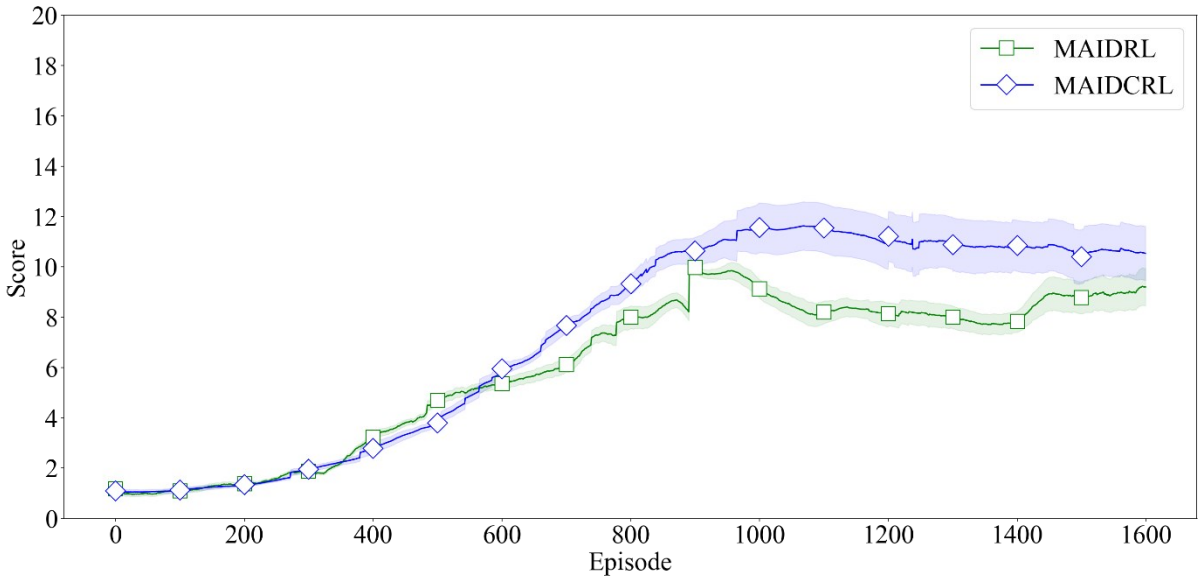

Figure 23. Average of the Running Average Episode Reward on $25m$ Scenario

$2s3z$. Figure 24 reflects the performance for complex heterogeneous scenario, $2s3z$, which involves two different types of agents working together to attack opponents: two stalkers and three zealots. Initially, the MAIDRL model exhibits better performance compared to MAIDCRL in the early episodes. However, a closer examination reveals that MAIDRL fails to effectively learn from the environment, leading to a stagnant and steady performance over time. In contrast, MAIDCRL demonstrates a different learning pattern. It gradually learns from the

environment, acquiring knowledge and adapting its strategies to improve performance. After approximately 800 episodes on average, MAIDCRL shows a noticeable improvement in performance, consistently achieving victories in most instances. This suggests that MAIDCRL has the ability to effectively utilize the learned knowledge and apply it to overcome challenges in the complex heterogeneous $2s3z$ scenario. The maximum average episodic reward has been improved from 13.7 to 18.88, which is 37.81% higher than MAIDRL without the CNN-enabled architecture. The average episodic reward is also enhanced by 17.5% with an overall stability in the performance. The lower STD value also demonstrates that the scores are clustered closely towards the average score ensuring the robustness of the proposed model.

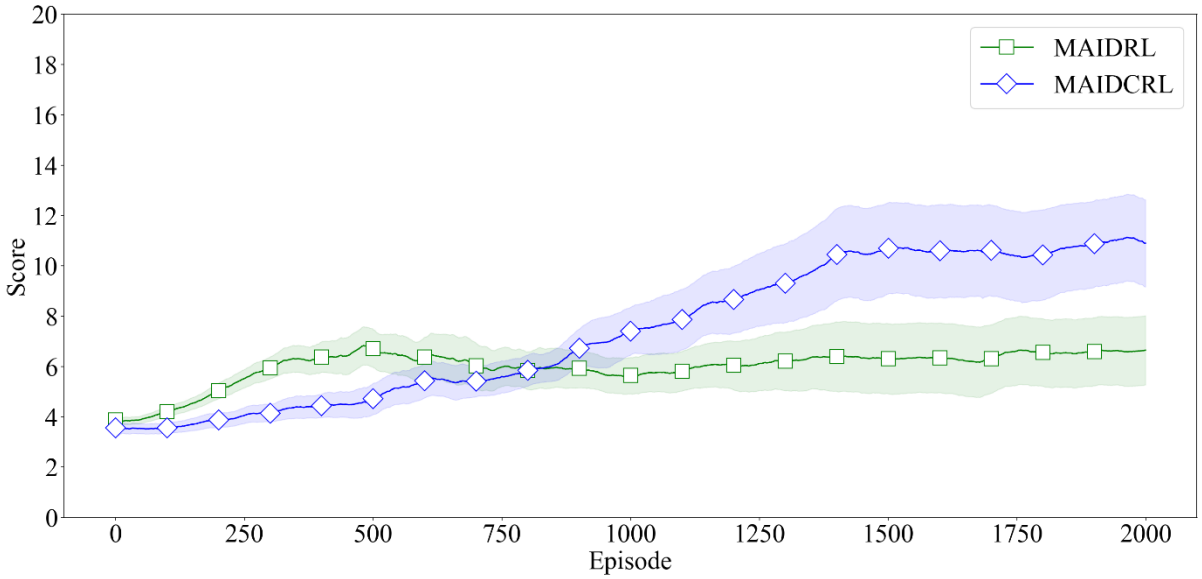

Figure 24. Average of the Running Average Episode Reward on $2s3z$ Scenario

$1c3s5z$. The $1c3s5z$ SMAC scenario refers to a specific setup within the SMAC environment. In this scenario, there are two teams, one composed of a single commanding agent (1c) and the other consisting of three stalkers (3s) and five zealots (5z). The objective of the

scenario is for the commanding agent and its team of units to engage in combat against the opposing team, which also consists of three stalkers and five zealots. This scenario is designed to test the performance and coordination of agents in a complex and dynamic combat situation. The commanding agent must make strategic decisions to control and guide its team of units effectively, while the stalkers and zealots must work together to attack and defeat the enemy units. It is important to note that this scenario is highly complex and does not provide a clear winning condition. Despite the initial challenges, I observed a significant improvement in the model's learning capability after this point, as indicated by the increased average score of 10.61 from 6.73, shown in Figure 25. While the lack of winning instances in this scenario might be seen as a limitation, I remain optimistic about the insights that can be gained by running the model for a larger number of episodes. Unfortunately, due to time constraints at this stage of my research, I was unable to explore the full potential of the model in this complex environment.

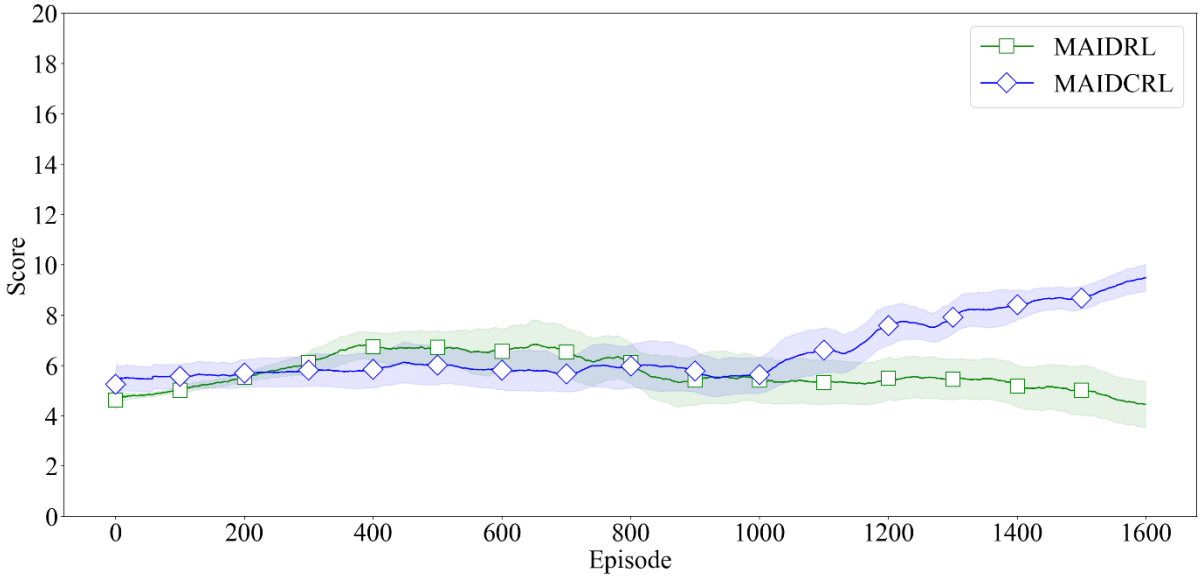

Figure 25. Average of the Running Average Episode Reward on $1c3s5z$ Scenario

Table 1 provides a detailed comparison of MAIDCRL and MAIDRL across all scenarios, including $3m$, $8m$, $25m$, $2s3z$ and $1c3s5z$. Boldly marked values indicate the best performance in each scenario. The results show that MAIDRL achieves a higher maximum running average reward in the $3m$ scenario. However, MAIDCRL outperforms MAIDRL in the more complex scenarios of $8m$, $25m$, and $2s3z$. Moreover, MAIDCRL achieves higher minimum running average scores in all homogeneous scenarios. Specifically, MAIDCRL surpasses MAIDRL by 15.65, 12.95, 12.14 and 10.61 in the $8m$, $25m$, $2s3z$ and $1c3s5z$ scenarios, respectively. Overall, the addition of CNN layers significantly enhances the performance of MAIDCRL in both complex homogeneous and heterogeneous scenarios. The minimum running average score without using CNN is 6.45, whereas MAIDCRL achieves a value of 9.22 in the $25m$ scenario.

Table 1. Performance Comparison between MAIDRL and MAIDCRL on Extended Scenarios

Scenario	Method	Min	Max	Avg	Std
$3m$	MAIDRL	4.29	**17.14**	11.01	4.19
	MAIDCRL	**5.01**	16.98	**14.84**	**3.12**
$8m$	MAIDRL	5.01	16.93	14.77	**3.77**
	MAIDCRL	**6.82**	**18.79**	**15.65**	3.81
$25m$	MAIDRL	6.45	13.59	11.82	**0.94**
	MAIDCRL	**9.22**	**16.09**	**12.95**	1.25
$2s3z$	MAIDRL	**5.32**	13.70	10.33	3.43
	MAIDCRL	5.12	**18.88**	**12.14**	**2.25**
$1c3s5z$	MAIDRL	6.71	12.16	6.73	1.72
	MAIDCRL	**8.27**	**14.27**	**10.61**	**1.35**

MAIDRL and MAIDCRL with 64 × 64 MAIM dimension.

Pre-trained Policy Selection for Transfer Learning

In the initial phase of transfer learning, I conducted training on selected homogeneous scenarios for a total of 2000 epochs, using 31 independent instances running in parallel. During this training process, the agents leveraged the information gathered from the ongoing simulations to update their respective neural networks. In order to compare the performance of these models, I conducted evaluations with different input state resolutions. Table 2 presents the results obtained from the evaluations of the $3m$, $8m$, and $25m$ scenarios, using various local state dimensions ranging from 19×19 to 55×55 resolutions. The results indicated that the agents in the $3m$ and $8m$ scenarios achieved a score of 20, which indicates their successful defeat of the opposing team in the respective scenarios. However, in the $25m$ scenario, the highest score attained by one of the well-trained models was 15.97, using a local state representation dimension of 37×37. Considering the overall stability of the performance, I made a decision to choose the 37×37 dimension for the local state reformulation in a unified manner. This dimension exhibited a desirable balance between performance and stability across the evaluated scenarios.

The best-performing RL model from each of the $3m$, $8m$, and $25m$ scenarios was selected as a seed for the subsequent training of shared neural networks on other scenarios, utilizing the powerful technique of transfer learning. By leveraging the knowledge gained from the initial training on the selected scenarios, the transfer learning approach enables the shared neural networks to benefit from the learned strategies and insights. This process allows for more efficient and effective training of the agents on additional scenarios, enhancing their overall performance and adaptability in a range of heterogeneous environments.

Table 2. Best Performing RL Models Learning from Scratch

Scenario	Dimension[1]	Min	Max	Avg	Std
$3m$	19×19	2.32	16.57	8.34	**3.24**
	37×37	**4.29**	**20**	**12.39**	5.21
	55×55	1.29	15.21	12.37	3.26
$8m$	19×19	4.61	19.57	9.29	4.93
	37×37	**8.19**	**20**	**11.74**	**3.34**
	55×55	6.45	18.3	8.74	5.22
$25m$	19×19	3.36	14.51	6.42	2.67
	37×37	4.93	**15.97**	**7.45**	**1.28**
	55×55	**5.16**	11.90	8.32	1.96

[1] the dimension of unified local observation.

Robustness of MAIDCRL Architecture

To assess the effectiveness and reliability of the approach presented in this study, I conducted a comprehensive analysis of the number of successful instances across multiple seeds for both the CNN-enabled MAIDCRL and the previously developed MAIDRL models. Specifically, I focused on three types of homogeneous scenarios: $3m$, $8m$, $25m$ and two types of heterogeneous scenarios $2s3z$ and $1c3s5z$. The results, as depicted in Figure 26, clearly demonstrate the superior performance of MAIDCRL. In scenario $3m$, MAIDCRL achieved victory in all runs, while on $8m$ it secured victory in 28 out of 31 runs. Additionally, in the more complex scenario $2s3z$, MAIDCRL emerged winning in 18 out of 31 runs. In contrast, MAIDRL was unable to surpass my proposed model's performance in any of these challenging scenarios.

Furthermore, I conducted a detailed analysis of the learning speed exhibited by the proposed solution. Figure 27 provides insights into the number of episodes required for both MAIDCRL and MAIDRL to discover their first winning strategy in each scenario. On average,

CNN-enabled MAIDCRL achieved its initial victory after approximately 770 episodes in scenario $3m$, 950 episodes in scenario $8m$, and 1053 episodes in scenario $2s3z$. In comparison, MAIDRL required 800 episodes to secure victory in scenario $3m$, 1200 episodes in scenario $8m$, and 1240 episodes in scenario $25m$. It is worth noting that neither model achieved a winning strategy in scenario $25m$ and $1c3s5z$, and thus these particular scenarios were excluded from the evaluation.

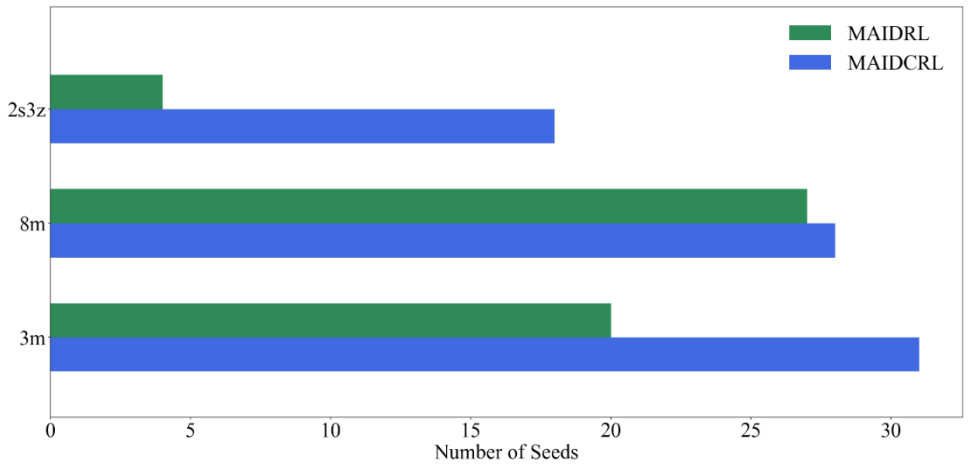

Figure 26. Total Number of Winning Across All Seeds

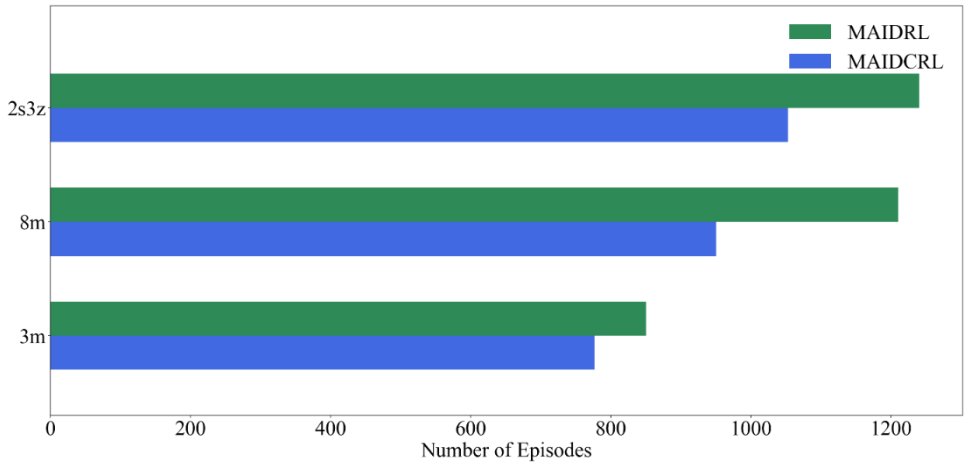

Figure 27. Average Number of Episodes for First Winning

These findings provide compelling evidence of the superior performance and faster learning capabilities of the CNN-enabled MAIDCRL model compared to MAIDRL across various homogeneous and heterogeneous scenarios. The CNN integration in MAIDCRL allows for more efficient and effective decision-making, leading to a higher number of successful outcomes. These results underscore the significance of incorporating convolutional neural networks in multi-agent influence map inputs in the field of multi-agent reinforcement learning. The robustness and accelerated learning exhibited by the CNN-enabled MAIDCRL model offer promising prospects for its application in real-world scenarios, where the ability to quickly adapt and make optimal decisions is of utmost importance.

Transfer Learning Performance

The performance of transfer learning was assessed by leveraging seeds from diverse homogeneous scenarios to train additional scenarios based on the level of difficulty in the SMAC environment. This evaluation aimed to determine the effectiveness of transferring knowledge across the selected environments. The following subsection provides detailed evaluation about the observed TL performance improvements in the $3m$, $8m$, and $25m$ homogeneous scenarios with varying seed policies.

SMAC-$3m$. Figure 28 highlights the notable enhancement in performance observed in the $3m$ scenario when transferring knowledge from the $8m$ and $25m$ scenarios, as opposed to starting from scratch without shared information. Utilizing the pre-trained policies of $8m$ and $25m$ results in an average running episode reward improvement of 9.7% and 19.97% in $3m$, respectively. The presence of a larger number of actively learning Marines in $8m$ and $25m$

contributes to improved decision making compared to the $3m$ scenario, where only three Marines are engaged in combat against enemy units.

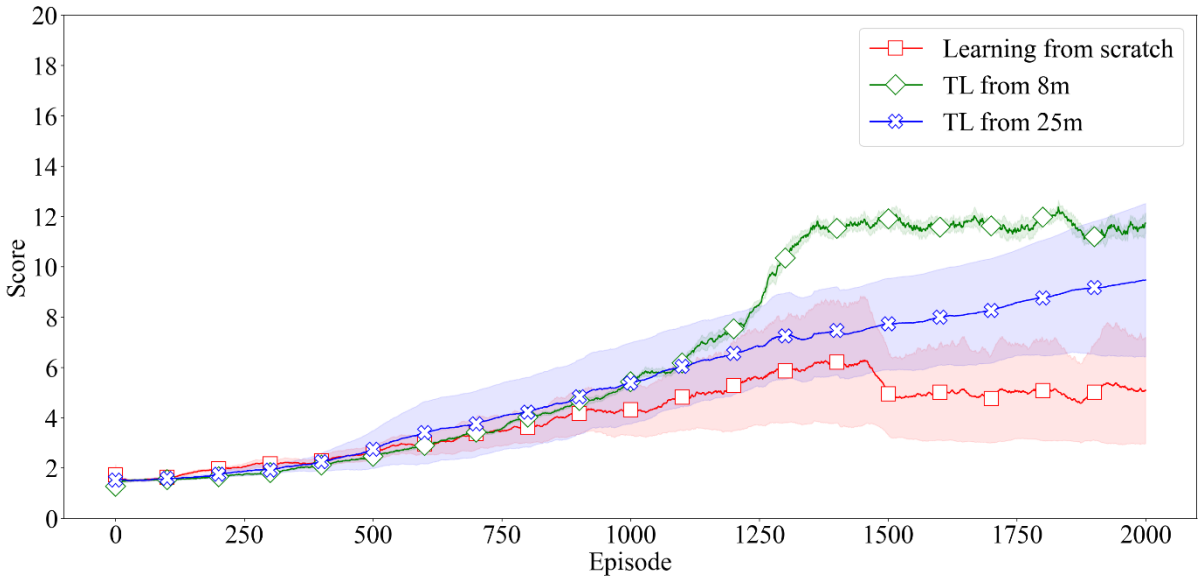

Figure 28. Average of the Running Average Episode Reward on $3m$ TL

SMAC-$8m$. Figure 29 illustrates the impact of the $3m$ and $25m$ seed policy on $8m$ scenario. The TL-$3m$ curve demonstrates an overall average score improvement of 45.3% compared to the learn from scratch method. Initially, the pre-trained models exhibit lower performance due to the transition to an unseen new scenario, but after 1000 iterations, it surpasses other models and progresses towards a winning strategy in each episode. Additionally, reusing the pre-trained $25m$ model enhances the average performance by 8.01%, surpassing the performance of the base units.

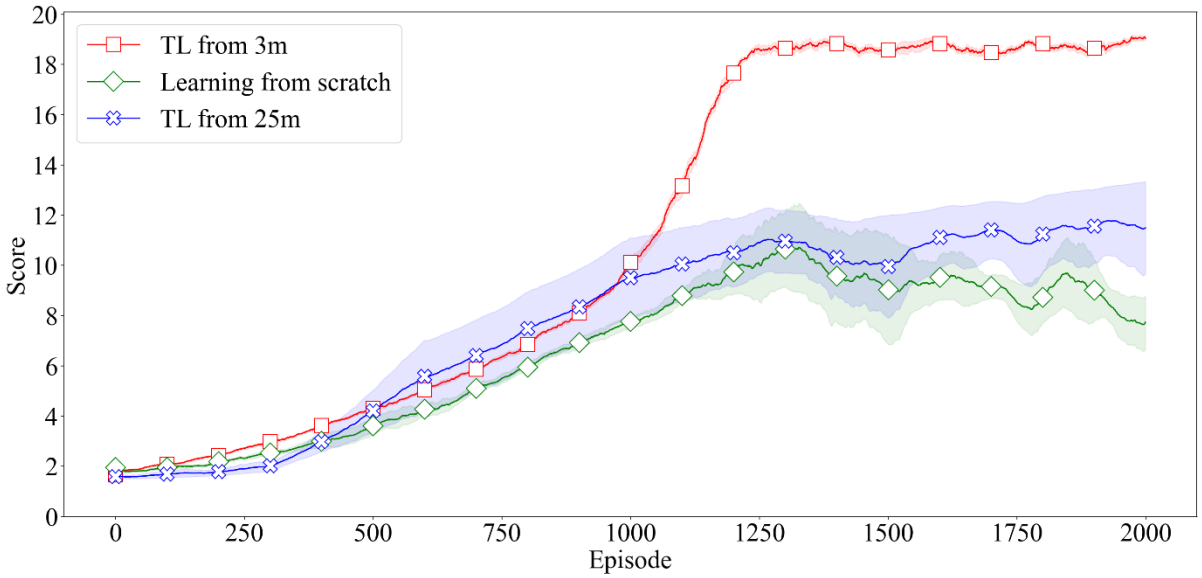

Figure 29. Average of the Running Average Episode Reward on $8m$ TL

SMAC-$25m$. In Figure 30, we can observe the significant impact of utilizing the seeds from the 3m and 8m scenarios on the performance of the 25m scenario. The introduction of these pre-trained seeds leads to notable improvements in the average running episode reward, with a remarkable 36.4% enhancement when employing the 3m seed and an even more substantial 41.3% improvement with the 8m seed, both compared to the learning from scratch approach. Achieving stable winning outcomes in the 25m scenario can be challenging due to the presence of a large number of active agents, making it a complex and dynamic environment. However, the model demonstrates its adaptability and ability to learn from the provided seeds. By incorporating a better representation of the state space, the model becomes more proficient in handling enemy units, which results in a performance that slightly surpasses that of the base units.

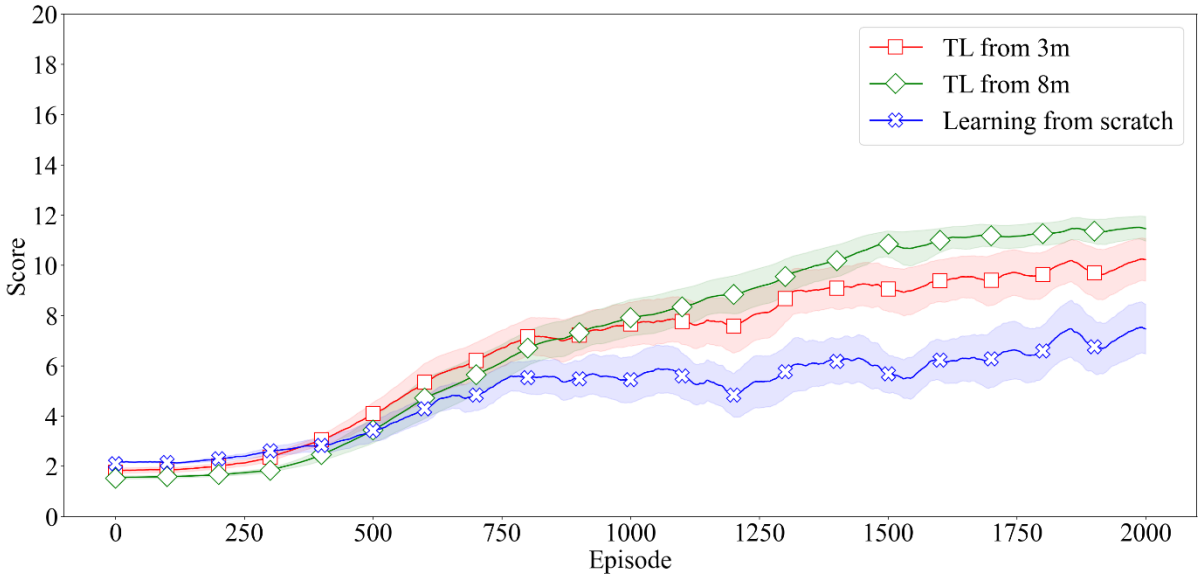

Figure 30. Average of the Running Average Episode Reward on $25m$ TL

Table 3 presents the statistical analysis of the transfer learning model in the extended homogeneous SMAC environment. The evaluation considered four primary metrics: maximum, minimum, average, and standard deviation (STD) of the running average episode reward across 31 seeds. Incorporating pre-trained seed policies $8m$, $3m$, and $25m$ during the training phase, instead of starting from scratch, yielded the highest minimum average running episode reward scores of 5.21, 11.5, and 6.54 for the $3m$, $8m$, and $25m$ scenarios, respectively. The maximum and average episode rewards outperformed the base scenarios, highlighting the effectiveness of transfer learning and the utilization of knowledge from other scenarios. The reduced STD value indicates a decrease in score fluctuation, indicating a more stable model in terms of performance. The highest values are distinctly marked and highlighted across the various SMAC scenarios.

Table 3. Performance Evaluation of TL in SMAC scenarios

Scenario	Pretrained Policy	Min	Max	Avg	Std
$3m$	–	4.29	18.34	12.37	3.74
	$8m$	**5.21**	**20**	13.57	**0.28**
	$25m$	3.31	17.78	**14.84**	3.12
$8m$	$3m$	**11.5**	**20**	**17.06**	**0.1**
	–	8.19	18.41	11.74	3.34
	$25m$	10.95	17.8	12.68	1.89
$25m$	$3m$	4.31	13.59	11.35	1.69
	$8m$	**6.54**	**14.39**	**11.76**	**1.28**
	–	5.16	11.9	8.32	1.96

– indicates learning from scratch.

Curriculum Transfer Learning Performance

This study focuses on investigating the impact of TL on homogeneous scenarios to evaluate the transfer of knowledge among agents. The experimental results highlight a significant improvement in MARL performance. In addition, I aim to extend the evaluation to complex heterogeneous scenarios, examining not only the transfer of knowledge within the same unit type but also the transfer of knowledge between different types of units. To achieve this, I selected the $2s3z$ as a testing bed and compared the performance using seeds obtained from previous experiments. Specifically, I examined the outcome of CTL where the $3m$ model, representing one of the simplest maps in the SMAC environment, was trained initially. After training each $3m$ instance for 2000 episodes, I saved the best-performing model and employed it for training in the more challenging $8m$ scenario. The knowledge acquired in controlling Marines from the $3m$ and $8m$ scenarios was then applied to the most difficult scenario in my test environment, $2s3z$, which consists of Stalkers and Zealots that the model had not encountered in

the previous training scenarios. Figure 31 illustrates the results for $2s3z$ model across different TL and CTL scenarios. The graph demonstrates an upward trend in the performance of my curriculum model as knowledge is transferred from $3m$ to $8m$, then $8m$ to $2s3z$. The TL model, utilizing pre-trained seeds from $3m$ and $8m$, improved the average reward of the $2s3z$ scenario by 38.6% and 38.2% respectively compared to learning from scratch on $2s3z$. However, the performance of all these simulations was surpassed by the CTL approach with an overall improvement of 72.2%.

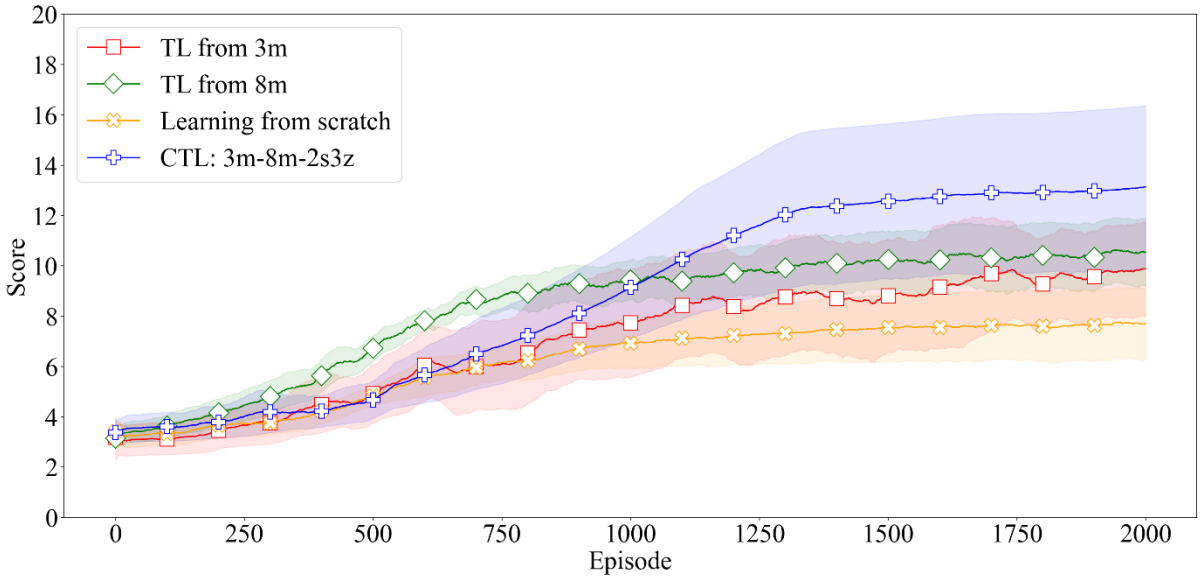

Figure 31. Result of Curriculum Transfer Learning on $2s3z$ Scenario

Table 4 presents a comprehensive statistical analysis of the performance metrics in this study. Specifically, it highlights the effectiveness of the curriculum transfer approach, denoted as $3m \rightarrow 8m \rightarrow 2s3z$, in achieving remarkable results in my simulation environment. This approach yielded the highest maximum average reward recorded as 17.2, indicating the exceptional performance achieved by the curriculum architectures. Moreover, it also attained the

highest average reward of 13.83, further solidifying the success of this proposed methodology. These impressive outcomes provide clear evidence of the robustness of the curriculum architectures in facilitating the transfer of knowledge within and between agents. This study showcases the significant advancements in both intra-agent knowledge transfer, within the same unit type, and inter-agent knowledge transfer, among different types of units. This demonstrates the effectiveness and adaptability of my approach in promoting the exchange of knowledge among agents, ultimately enhancing their overall performance in complex scenarios.

Table 4. Performance Evaluation of CTL on $2s3z$ Scenario

Scenario	Pretrained Policy	Min	Max	Avg	Std
$2s3z$	$3m$	5.85	18.34	12.37	3.74
	$8m$	**8.91**	16.19	11.1	**3.24**
	$-$	5.68	13.61	8.03	3.66
	CTL[a]	4.93	**17.2**	**13.83**	4.98

$-$ indicates learning from scratch.

[a] CTL: $3m \rightarrow 8m \rightarrow 2s3z$

Learned Behavior Analysis of StarCraft Agents

The focus of my analysis thus far has primarily been on the statistical evaluation of performance and robustness. However, it is equally important to provide a qualitative comparison of the learned behaviors exhibited in both cooperative and competitive environments. Therefore, I have conducted an in-depth exploration of these behaviors to gain further insights into their characteristics and impact.

In order to assess the learned behaviors, I selected the best performing RL models for each scenario to serve as the trained controllers in the test runs. During the observation of

episodes played by the MAIDCRL controller, I identified two key strategies consistently employed by the agents to achieve victory in skirmishes. The first strategy involves a collaborative approach to enemy targeting, where agents prioritize attacking enemy units together. This collaborative focus on fire power significantly increases the likelihood of success compared to individualistic targeting strategies. Furthermore, my analysis revealed the importance of effective unit positioning. Agents that are able to reposition themselves strategically after sustaining damage, with minimal movement, demonstrate higher levels of success in the game. This finding emphasizes the significance of agility and adaptability in combat situations, where the ability to quickly adjust positioning to optimize defensive and offensive maneuvers can greatly influence the outcome of engagements.

The specific characteristics of these observed behaviors and their impact on overall performance are thoroughly illustrated in the subsequent subsections. By delving into the underlying strategies and tactics employed by the trained models, I gain valuable insights into the dynamics of multi-agent interactions and the factors that contribute to successful outcomes. This qualitative analysis provides a deeper understanding of the behaviors exhibited by the RL models and their implications within the context of the game environment. By elucidating these behavior-specific characteristics and their impact, I contribute to the broader knowledge and understanding of multi-agent reinforcement learning, shedding light on the intricate dynamics of collaborative and competitive decision-making.

SMAC-3*m*. This particular scenario examined in this study is characterized by high instability compared to the other evaluated environments. The limited number of agents in this scenario poses a challenge in identifying optimal unit positions for coordinated enemy attacks. My observations indicate that when the ally agents successfully determine the appropriate

position, they are able to easily overcome the SC2 AI agents. However, in instances where they fail to identify the optimal position, the outcome of the game is determined by the inherent randomness embedded within the game engine. As a result of this uncertainty, even after training the model for a significant number of episodes, the results continue to fluctuate. To provide visual representation of the random states observed in this scenario, Figure 32 illustrates the scattered nature of $3m$ agents in which both controlled agents and SC2 AI agents randomly attempting to determine the attacking position.

Figure 32. Random Agents Positioning on $3m$ Scenario

SMAC-$8m$. In order to investigate the learning behavior, I conducted experiments by deploying my trained RL model on the $8m$ scenario, which revealed several noteworthy characteristics. During the games controlled by MAIDCRL models, I observed a tendency towards collective movement among the agents, despite each agent making its decisions in a fully decentralized manner. The agents demonstrated collaborative attacks, with three units targeting enemy units together. This coordinated approach resulted in faster elimination of

individual enemy units, leading to a higher rate of success in winning the game compared to randomly selecting target units.

It is important to note that I utilized a 64×64 influence map for the MAIDCRL controller, as it yielded the best performance on the $8m$ scenario compared to other influence map dimensions. Another interesting finding from the behavioral analysis is that the agents encountered difficulty in finding the appropriate shooting position when only a few enemies remained. However, after several time steps, the agents were able to successfully engage the remaining enemies and move closer to victory. One possible explanation for this behavior is that during the late stages of each training episode, the MAIDCRL model has limited exposure to scenarios with a small number of remaining enemies, thus hindering its ability to learn optimal actions effectively. To provide a visual representation of the dominance of the MAIDCRL model over the SC2 AI agents, Figure 33 illustrates a scenario where four controlled agents, organized into two groups, target a single enemy unit. This exemplifies the collaborative and effective strategy employed by MAIDCRL models in achieving success in the game.

Figure 33. MAIDCRL Agents dominance over SC2 AI on $8m$ Scenario

SMAC-25*m*. Efficiently arranging all 25 marines to effectively combat the SC2 AI enemies proved to be a challenging task in the observations. I discovered that some of the marines positioned themselves strategically as frontliners, allowing them to readjust their formation with minimal movement in the event of sustaining damage and requiring healing. This adaptive positioning strategy showcased the agents' ability to optimize their defensive capabilities while minimizing disruption to their offensive formation.

During the analysis of over ten models, I also observed an intriguing behavior exhibited by the agents. Before initiating a collective attack on the enemies, the agents positioned themselves in close proximity to the enemy units within their firing range. This tactical maneuver forced the enemies to assess the potential damage they might incur while attempting to engage the agents. This approach, as depicted in Figure 34, demonstrates the agents' cautiousness and emphasis on mitigating potential damage, thereby highlighting their active focus on countering opponents' attacks. Moreover, it is worth mentioning that the agents exhibited a higher level of deliberation in unit positioning compared to the SC2 AI counterparts. This deliberation further underscores their strategic awareness and their proactive approach in dealing with opponent attacks.

Figure 34. Travelling Units position on $25m$ Scenario

SMAC-$2s3z$. During the evaluation of the heterogeneous scenario $2s3z$, I made an intriguing observation regarding the importance of target prioritization when facing an enemy team consisting of multiple types of units. In this scenario, where both the enemy team and the ally agents possessed a mix of melee zealots and ranged stalkers, I noticed a distinct behavior exhibited by the zealot units. As depicted in Figure 35, the zealots demonstrated a strategic decision-making process by moving past the enemy zealots positioned in the front line. Instead of engaging in direct combat with the nearest enemy units, the zealots showcased an intelligent approach by prioritizing their target selection. They focused their fire on the enemy stalkers located in the rear lines before engaging the closer enemy zealots.

This observed behavior highlights the agents' ability to assess the potential threats and opportunities presented by different unit types within the enemy team. By strategically prioritizing their targets, the zealot units effectively neutralized the enemy stalkers, which are typically more dangerous to the zealots due to their ranged attack capabilities. This tactical decision contributed to the overall success of the agents in the $2s3z$ scenario. The observed

target prioritization behavior of the zealots illustrates the strategic sophistication and adaptability of my RL model. This capability enables the ally agents to make informed decisions based on the varying threat levels posed by different enemy units. Such intelligent target selection enhances the agents' overall combat effectiveness and demonstrates their ability to dynamically adjust their tactics based on the specific context of the scenario.

Figure 35. Zealots Attacking Stalkers on $2s3z$ Scenario

CONCLUSION AND FUTURE WORK

In this study, I extended the semi-centralized RL model MAIDRL by introducing a novel CNN-enabled approach, MAIDCRL, to address various MARL systems [78]. This evaluation focused on assessing the performance of MAIDCRL in both homogeneous and heterogeneous SMAC scenarios of varying complexity, providing statistical results to support my findings. Notably, MAIDCRL exhibited significant improvements in overall performance, robustness, generalizability, and peak performance across selected scenarios. These findings offer valuable insights into the design of influence maps and feature discovery in Multi-Agent Reinforcement Learning algorithms.

To enable transfer learning and curriculum transfer learning in different MARL systems, this study further introduced a novel approach for unifying various state sizes into fixed-size inputs. Limited data availability and the need for agents to quickly adapt to new environments pose challenges in designing effective MAS. Transfer learning enables agents to leverage knowledge acquired from previous experiences, accelerating learning in new domains and reducing the reliance on extensive data collection and training. Through evaluating the proposed TL and CTL approaches using uniformed state and action representations in both homogeneous and heterogeneous SMAC scenarios, I observed substantial improvements in overall learning performance, generalizability, and peak performance compared to the previous approach [79]. This highlights the potential for developing unified representations and discovering relevant features in MARL algorithms.

While this study provides valuable insights, there are several areas that can be further explored. Firstly, my model's performance was tested on a limited number of scenarios, and

future research should investigate its effectiveness in a wider range of heterogeneous environments featuring more complex maps. Additionally, the use of multiple influence maps and other kernel functions can be explored to uncover correlations between spatial features at local, shared, and global levels. Another key aspect of the future scope lies in further developing and refining coevolutionary algorithms that will tailor the knowledge from both ally and enemy units. These algorithms might enable agents to coevolve their strategies by competing and collaborating with each other. By iteratively evolving and evaluating their strategies, agents can adapt to dynamic game environments, discover novel tactics, and respond to opponents' strategies more efficiently. This approach might instigate the potential to unlock new levels of strategic depth and evolve for pushing the boundaries of AI research and game development.

REFERENCES

[1] S. Grigorescu, B. Trasnea, T. Cocias and G. Macesanu, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics,* vol. 37, p. 362–386, 2020.

[2] R. R. Murphy, Introduction to AI robotics, MIT press, 2019.

[3] J. Jumper, R. Evans, A. Pritzel, T. Green, M. Figurnov, O. Ronneberger, K. Tunyasuvunakool, R. Bates, A. Žídek, A. Potapenko and others, "Highly accurate protein structure prediction with AlphaFold," *Nature,* vol. 596, p. 583–589, 2021.

[4] F.-Y. Wang, J. J. Zhang, X. Zheng, X. Wang, Y. Yuan, X. Dai, J. Zhang and L. Yang, "Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond," *IEEE/CAA Journal of Automatica Sinica,* vol. 3, p. 113–120, 2016.

[5] 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.

[6] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski and others, "Human-level control through deep reinforcement learning," *nature,* vol. 518, p. 529–533, 2015.

[7] M. Johnson, K. Hofmann, T. Hutton and D. Bignell, "The Malmo Platform for Artificial Intelligence Experimentation.," in *Ijcai*, 2016.

[8] O. Vinyals, T. Ewalds, S. Bartunov, P. Georgiev, A. S. Vezhnevets, M. Yeo, A. Makhzani, H. Küttler, J. Agapiou, J. Schrittwieser and others, "Starcraft ii: A new challenge for reinforcement learning," *arXiv preprint arXiv:1708.04782,* 2017.

[9] M. Samvelyan, T. Rashid, C. S. De Witt, G. Farquhar, N. Nardelli, T. G. J. Rudner, C.-M. Hung, P. H. S. Torr, J. Foerster and S. Whiteson, "The starcraft multi-agent challenge," *arXiv preprint arXiv:1902.04043,* 2019.

[10] 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*, 2018.

[11] 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.

[12] D. Xie and X. Zhong, "Semicentralized deep deterministic policy gradient in cooperative StarCraft games," *IEEE transactions on neural networks and learning systems,* vol. 33, p.

1584–1593, 2020.

[13] X. Wang, J. Song, P. Qi, P. Peng, Z. Tang, W. Zhang, W. Li, X. Pi, J. He, C. Gao and others, "SCC: An efficient deep reinforcement learning agent mastering the game of StarCraft II," in *International conference on machine learning*, 2021.

[14] A. Harris and S. Liu, "Maidrl: Semi-centralized multi-agent reinforcement learning using agent influence," in *2021 IEEE Conference on Games (CoG)*, 2021.

[15] F. L. Da Silva and A. H. R. Costa, "A survey on transfer learning for multiagent reinforcement learning systems," *Journal of Artificial Intelligence Research,* vol. 64, p. 645–703, 2019.

[16] G. Zhang, Y. Li, X. Xu and H. Dai, "Efficient training techniques for multi-agent reinforcement learning in combat tasks," *IEEE Access,* vol. 7, p. 109301–109310, 2019.

[17] A. Shantia, E. Begue and M. Wiering, "Connectionist reinforcement learning for intelligent unit micro management in starcraft," in *The 2011 international joint conference on neural networks*, 2011.

[18] A. Das, S. Roy, U. Bhattacharya and S. K. Parui, "Document image classification with intra-domain transfer learning and stacked generalization of deep convolutional neural networks," in *2018 24th international conference on pattern recognition (ICPR)*, 2018.

[19] J. Tao and X. Fang, "Toward multi-label sentiment analysis: a transfer learning based approach," *Journal of Big Data,* vol. 7, p. 1–26, 2020.

[20] M. I. Jordan and T. M. Mitchell, "Machine learning: Trends, perspectives, and prospects," *Science,* vol. 349, p. 255–260, 2015.

[21] M. Mohri, A. Rostamizadeh and A. Talwalkar, Foundations of machine learning, MIT press, 2018.

[22] M. Bain and C. Sammut, "A Framework for Behavioural Cloning.," in *Machine Intelligence 15*, 1995.

[23] P. J. Rousseeuw, "Silhouettes: a graphical aid to the interpretation and validation of cluster analysis," *Journal of computational and applied mathematics,* vol. 20, p. 53–65, 1987.

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

[25] R. Bellman, "A Markovian decision process," *Journal of mathematics and mechanics,* p. 679–684, 1957.

[26] S. J. Russell, Artificial intelligence a modern approach, Pearson Education, Inc., 2010.

[27] R. Bellman, "Dynamic programming and Lagrange multipliers," *Proceedings of the National Academy of Sciences,* vol. 42, p. 767–769, 1956.

[28] R. M. French, "Catastrophic forgetting in connectionist networks," *Trends in cognitive sciences,* vol. 3, p. 128–135, 1999.

[29] C. J. C. H. Watkins and P. Dayan, "Q-learning," *Machine learning,* vol. 8, p. 279–292, 1992.

[30] G. A. Rummery and M. Niranjan, On-line Q-learning using connectionist systems, vol. 37, University of Cambridge, Department of Engineering Cambridge, UK, 1994.

[31] R. S. Sutton, D. McAllester, S. Singh and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *Advances in neural information processing systems,* vol. 12, 1999.

[32] D. Zhao, H. Wang, K. Shao and Y. Zhu, "Deep reinforcement learning with experience replay based on SARSA," in *2016 IEEE symposium series on computational intelligence (SSCI)*, 2016.

[33] E. Parisotto, J. L. Ba and R. Salakhutdinov, "Actor-mimic: Deep multitask and transfer reinforcement learning," *arXiv preprint arXiv:1511.06342,* 2015.

[34] R. Laroche and M. Barlier, "Transfer reinforcement learning with shared dynamics," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

[35] T. Schaul, J. Quan, I. Antonoglou and D. Silver, "Prioritized experience replay," *arXiv preprint arXiv:1511.05952,* 2015.

[36] A. Nair, P. Srinivasan, S. Blackwell, C. Alcicek, R. Fearon, A. De Maria, V. Panneershelvam, M. Suleyman, C. Beattie, S. Petersen and others, "Massively parallel methods for deep reinforcement learning," *arXiv preprint arXiv:1507.04296,* 2015.

[37] H. Van Hasselt, A. Guez and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, 2016.

[38] 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*, 2016.

[39] D. Li, D. Zhao, Q. Zhang and C. Luo, "Policy gradient methods with gaussian process modelling acceleration," in *2017 International Joint Conference on Neural Networks*

(IJCNN), 2017.

[40] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra and M. Riedmiller, "Deterministic policy gradient algorithms," in *International conference on machine learning*, 2014.

[41] M. Lin, Q. Chen and S. Yan, "Network in network," *arXiv preprint arXiv:1312.4400,* 2013.

[42] A. Skrynnik, A. Staroverov, E. Aitygulov, K. Aksenov, V. Davydov and A. I. Panov, "Hierarchical deep q-network from imperfect demonstrations in minecraft," *Cognitive Systems Research,* vol. 65, p. 74–78, 2021.

[43] A. Y. Ng, D. Harada and S. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Icml*, 1999.

[44] S. Huang and S. Ontañón, "Action guidance: Getting the best of sparse rewards and shaped rewards for real-time strategy games," *arXiv preprint arXiv:2010.03956,* 2020.

[45] V. Zambaldi, D. Raposo, A. Santoro, V. Bapst, Y. Li, I. Babuschkin, K. Tuyls, D. Reichert, T. Lillicrap, E. Lockhart and others, "Relational deep reinforcement learning," *arXiv preprint arXiv:1806.01830,* 2018.

[46] S. Liu, S. J. Louis and M. Nicolescu, "Comparing heuristic search methods for finding effective group behaviors in RTS game," in *2013 IEEE Congress on Evolutionary Computation*, 2013.

[47] S. Liu, S. J. Louis and M. Nicolescu, "Using CIGAR for finding effective group behaviors in RTS game," in *2013 IEEE Conference on Computational Inteligence in Games (CIG)*, 2013.

[48] S. Gu, J. G. Kuba, M. Wen, R. Chen, Z. Wang, Z. Tian, J. Wang, A. Knoll and Y. Yang, "Multi-agent constrained policy optimisation," *arXiv preprint arXiv:2110.02793,* 2021.

[49] J. Schulman, S. Levine, P. Abbeel, M. Jordan and P. Moritz, "Trust region policy optimization," in *International conference on machine learning*, 2015.

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

[51] C. Berner, G. Brockman, B. Chan, V. Cheung, P. Dębiak, C. Dennison, D. Farhi, Q. Fischer, S. Hashme, C. Hesse and others, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680,* 2019.

[52] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen and Y. Wu, "The surprising effectiveness of ppo in cooperative multi-agent games," *Advances in Neural Information*

Processing Systems, vol. 35, p. 24611–24624, 2022.

[53] M. Stanescu, N. A. Barriga, A. Hess and M. Buro, "Evaluating real-time strategy game states using convolutional neural networks," in *2016 IEEE Conference on Computational Intelligence and Games (CIG)*, 2016.

[54] N. Kondo and K. Matsuzaki, "Playing game 2048 with deep convolutional neural networks trained by supervised learning," *Journal of Information Processing,* vol. 27, p. 340–347, 2019.

[55] O. Samuel, N. Javaid, A. Khalid, W. Z. Khan, M. Y. Aalsalem, M. K. Afzal and B.-S. Kim, "Towards real-time energy management of multi-microgrid using a deep convolution neural network and cooperative game approach," *IEEE Access,* vol. 8, p. 161377–161395, 2020.

[56] I. Sutskever and V. Nair, "Mimicking go experts with convolutional neural networks," in *Artificial Neural Networks-ICANN 2008: 18th International Conference, Prague, Czech Republic, September 3-6, 2008, Proceedings, Part II 18*, 2008.

[57] M. L. Koga, V. Freire and A. H. R. Costa, "Stochastic abstract policies: Generalizing knowledge to improve reinforcement learning," *IEEE Transactions on Cybernetics,* vol. 45, p. 77–88, 2014.

[58] M. Mahmud and S. Ray, "Transfer learning using Kolmogorov complexity: Basic theory and empirical evaluations," *Advances in neural information processing systems,* vol. 20, 2007.

[59] M. E. Taylor and P. Stone, "Transfer learning for reinforcement learning domains: A survey.," *Journal of Machine Learning Research,* vol. 10, 2009.

[60] J. Yosinski, J. Clune, Y. Bengio and H. Lipson, "How transferable are features in deep neural networks?," *Advances in neural information processing systems,* vol. 27, 2014.

[61] T. Chen, I. Goodfellow and J. Shlens, "Net2net: Accelerating learning via knowledge transfer," *arXiv preprint arXiv:1511.05641,* 2015.

[62] D. Xu, P. Qiao and Y. Dou, "Aggregation Transfer Learning for Multi-Agent Reinforcement learning," in *2021 2nd International Conference on Big Data & Artificial Intelligence & Software Engineering (ICBASE)*, 2021.

[63] M. J. Khan, S. Hassan and G. Sukthankar, "Leveraging Transformers for StarCraft Macromanagement Prediction," in *2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA)*, 2021.

[64] B. Tan, Y. Song, E. Zhong and Q. Yang, "Transitive transfer learning," in *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data*

Mining, 2015.

[65] R.-Z. Liu, H. Guo, X. Ji, Y. Yu, Z.-J. Pang, Z. Xiao, Y. Wu and T. Lu, "Efficient reinforcement learning for starcraft by abstract forward models and transfer learning," *IEEE Transactions on Games,* vol. 14, p. 294–307, 2021.

[66] K. Shao, Y. Zhu and D. Zhao, "Starcraft micromanagement with reinforcement learning and curriculum transfer learning," *IEEE Transactions on Emerging Topics in Computational Intelligence,* vol. 3, p. 73–84, 2018.

[67] P. Garćıa-Sánchez, A. Tonda, A. M. Mora, G. Squillero and J. J. Merelo, "Towards automatic StarCraft strategy generation using genetic programming," in *2015 IEEE Conference on Computational Intelligence and Games (CIG)*, 2015.

[68] A. A. Sánchez-Ruiz and M. Miranda, "A machine learning approach to predict the winner in StarCraft based on influence maps," *Entertainment Computing,* vol. 19, p. 29–41, 2017.

[69] P. Peng, Y. Wen, Y. Yang, Q. Yuan, Z. Tang, H. Long and J. Wang, "Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games," *arXiv preprint arXiv:1703.10069,* 2017.

[70] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin and others, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467,* 2016.

[71] I. Grondman, L. Busoniu, G. A. D. Lopes and R. Babuska, "A survey of actor-critic reinforcement learning: Standard and natural policy gradients," *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews),* vol. 42, p. 1291–1307, 2012.

[72] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980,* 2014.

[73] H. Wu and X. Gu, "Max-pooling dropout for regularization of convolutional neural networks," in *Neural Information Processing: 22nd International Conference, ICONIP 2015, Istanbul, Turkey, November 9-12, 2015, Proceedings, Part I 22*, 2015.

[74] V. Mnih, A. P. Badia, M. Mirza, A. Graves, T. Lillicrap, T. Harley, D. Silver and K. Kavukcuoglu, "Asynchronous methods for deep reinforcement learning," in *International conference on machine learning*, 2016.

[75] Y. Zhu and S. Newsam, "Densenet for dense flow," in *2017 IEEE international conference on image processing (ICIP)*, 2017.

[76] Y. Bengio, J. Louradour, R. Collobert and J. Weston, "Curriculum learning," in *Proceedings of the 26th annual international conference on machine learning*, 2009.

[77] A. Gupta, Y.-S. Ong and L. Feng, "Insights on transfer optimization: Because experience is the best teacher," *IEEE Transactions on Emerging Topics in Computational Intelligence,* vol. 2, p. 51–64, 2017.

[78] A. S. Nipu, S. Liu and A. Harris, "MAIDCRL: Semi-centralized Multi-Agent Influence Dense-CNN Reinforcement Learning," in *2022 IEEE Conference on Games (CoG)*, 2022.

[79] A. S. Nipu, S. Liu and A. Harris, "Enabling Multi-Agent Transfer Reinforcement Learning via Scenario Independent Representation," in *2023 IEEE Conference on Games (CoG)*, 2023.