MAIDRL: Semi-centralized Multi-Agent Reinforcement Learning using Agent Influence

1st Anthony Harris
Department of Computer Science
Missouri State University
Springfield, U.S.
Anthony999@MissouriState.edu

2nd Siming Liu
Department of Computer Science
Missouri State University
Springfield, U.S.
SimingLiu@MissouriState.edu

***Abstract*—In recent years, reinforcement learning algorithms have been used in the field of multi-agent systems to help the agents with interactions and cooperation on a variety of tasks. Controlling multiple agents simultaneously is extremely challenging as the complexity increases drastically with the number of agents in the system. In this study, we propose a novel semi-centralized deep reinforcement learning algorithm, MAIDRL, for mixed cooperative and competitive multi-agent environments. Specifically, we design a robust DenseNet-style actor-critic structured deep neural network for controlling multiple agents based on the combination of local observation and abstracted global information to compete with opponent agents. We extract common knowledge through influence maps considering both enemy and friendly agents for unit positioning and decision-making in combat. Compared to the centralized method, our design promotes a thorough understanding of the potential influence that a unit has without the need for a complete view of the global state. In addition, this design enables multi-agent understanding of common goals, unlike fully decentralized methods. The proposed method has been evaluated on StarCraft Multi-Agent Challenge scenarios in the real-time strategy game, StarCraft II, and the results show that, statistically, the agents controlled by MAIDRL perform better than or as well as those controlled by centralized and decentralized methods.**

***Index Terms*—Deep reinforcement learning, multi-agent system, influence map, StarCraft II, SMAC, MAIRL, MAIDRL**

I. Introduction

Artificial Intelligence (AI) has made enormous progress in many aspects of our world in recent decades. The rapid evolution in AI has enabled the high performance of a variety of tasks including robotics, autonomous driving, and game playing. For some of the above tasks, AI has reached human-level performance or even outperformed the best human experts. However, the majority of the achievements of AI have been in single-agent scenarios, where collaboration and competition among agents is unnecessary. There are a large number of applications that involve interaction between multiple agents. These applications can be modeled as cooperative or competitive multi-agent systems (MAS). For example, coordination of self-driving cars, multi-robot control, and multiplayer games all operate in a multi-agent domain. Among many AI techniques, reinforcement learning (RL) has been considered one of the most promising methods in the past several years. Unfortunately, solving such problems with traditional RL is non-trivial. Extending RL to enable cooperation and competition among agents is critical to building artificially intelligent systems in multi-agent environments.

There are many challenges in scaling up RL into multi-agent environments. One of the primary challenges of multi-agent RL (MARL) is that the traditional RL techniques like the Q-learning and policy gradient methods do not generalize well to MAS. This is evident when considering the state of the MAS from the perspective of an individual agent. The actions of this agent can be perceived to have non-stationary impacts in the environment as a direct result of the actions of other agents. This leads to significant stability issues, as well as the limitation of applicable stability enhancing techniques such as experience replay. Furthermore, MAS themselves introduce additional layers of complexities. For example, in cooperative MAS, agents must be able to act as a coordinated unit which requires an understanding of the agents with whom they are cooperating. This challenge in particular has been the focus of much research. A common starting point is to utilize complete state information to train agents. This technique, commonly referred as centralized learning, provides each agent with perfect environmental information regarding both the agent's local observations and the complete global information.

Many MAS, however, avoid the use of such perfect information as it may not be available during execution. Such work has led to decentralized learning, which only provides an agent with its local observations. This decentralized method, while more widely applicable in practice due to the lack of dependence upon global information, generally struggles to understand global objectives, leading to lackluster performance. This has been improved by the introduction of hybrid centralized learning, decentralized execution methods as presented by Foerster et al. and Lowe et al. in [1], [2]. In response to the aforementioned challenges, there has been much work regarding an intermediate technique called semi-centralized learning. Semi-centralized learning can be applied in various ways, but it largely revolves around the representation or provision of global information in an imperfect way that is more generalizable to scenarios where perfect information may be otherwise unavailable. An example of a semi-centralized

MARL technique is the Semi-Centralized Deep Deterministic Policy Gradient (SCDDPG) [3]. However, the question of how to use global information effectively for fine-grained decision-making is still open.

Our response to the above question is the definition and application of agent influence maps (AIM), aggregated into a global multi-agent influence map (MAIM), which is used in addition to local agent observations for fine-grained decision-making. By abstracting a subset of the perfect global information into an imperfect but descriptive representation, we demonstrate a significant improvement over centralized, decentralized, and hybridized methods. We evaluate the applicability of MAIMs by defining a simple Artificial Neural Network (ANN) that contains only a single hidden layer and comparing the results of MAIM state representation to a centralized alternative, with no additional changes. This use of semi-centralized MAIM state representation is defined as Multi-Agent Influence Reinforcement Learning (MAIRL). MAIRL is then applied with a DenseNet-style model architecture to improve the robustness and capability of the system. We define this combined use of MAIRL and DenseNet-style model architecture as Multi-Agent Influence Dense Reinforcement Learning (MAIDRL). MAIRL and MAIDRL are evaluated on StarCraft Multi-Agent Challenge (SMAC) scenarios in a real-time strategy game, StarCraft II (SC2) [4]. The $3m$, $8m$, $25m$, and $8m\_vs\_9m$ scenarios are used in our evaluations where the goal of each scenario is simply to defeat the opposing team. Each scenario contains two teams with a certain number of Marines, a SC2 medium-ranged infantry unit, where the number of Marines per team is given in the name of the scenario. The results show that, statistically, the agents controlled by MAIDRL perform better than or as well as those controlled by centralized and decentralized methods.

The remainder of this article is organized as follows: Section II introduces the relevant RL techniques that have been shown to perform well in RL and MARL scenarios, while Section III described the methodology regarding our experimentation. Section IV subsequently presents the results from said experimentation, and Section V draws the conclusions and suggests future works.

II. Related Work

Extensive studies have been performed on applying different variants of RL algorithms to controlling agents in cooperative and competitive multi-agent systems. Konda et al. introduced the actor-critic (AC) algorithm that combines value-based and policy-based learning methods by utilizing the best features of both Q-learning and policy gradient [5]. This method has been improved and adopted widely in the deep RL (DRL) community, with numerous variants such as deep deterministic policy gradient (DDPG) which applied experience replay and target network to expedite and improve the learning process [6]. Xie and Zhong expanded the DDPG algorithm into semi-centralized DDPG which utilized two-level AC structures to process local and global observations separately to help the agents with interactions and cooperation in StarCraft combat [3]. Lowe et al. utilized DDPG in MARL scenarios and introduced Multi-Agent DDPG (MADDPG) which showed promising results by developing cooperation and competition amongst agents in the grounded communication environment [2], [7]. Additionally, Mnih et al. proposed advantage actor-critic (A2C), which improves the learning curve of AC by considering the advantage of taking an action over another, and asynchronous advantage actor critic (A3C), which builds upon the benefits of A2C by running multiple instances of the simulation in parallel to reduce the overall training time, have shown significant improvements over the standard AC in the ALE [8]. The AC family has also been combined with Monte-Carlo Tree Search by Silver et al. showing remarkable success in AlphaGo, and have been explored in tandem with centralized learning, decentralized execution techniques by Foerster et al. in StarCraft [1], [9]. Our work differs from Foerster et al.'s work in that we focus primarily on representing the global information in an abstracted way as opposed to providing the critic with direct global feature input.

In the MARL spectrum specifically, algorithms such as the bidirectionally coordinated network (BiCNet) introduced by Peng et al. have shown that using a vectorized extension of the AC family can perform well with arbitrary numbers of agents being considered [10]. Schulman et al. improved the way in which DRL models learn with the introduction of trust region policy approximation (TRPO) in 2015 and proximal policy optimization (PPO) in 2017 [11], [12]. The research around PPO is distinct from our work as PPO's focus is on improving the way in which models learn by clipping the gradient, while our focus is on state representation.

There has also been research into several other RL methods that are distinct from DRL, but can be supplementary. For example, Bain and Sammut introduced behavioral cloning, or imitation learning, which is the process of using supervised learning to learn a policy based upon a dataset of state-action pairs from expert replays, which has been shown to improve the performance of DRL methods when used as a warm-start in various complex environments such as Minecraft and SC2 [13]–[16]. Reward shaping, introduced by Ng, Harada, and Russell, allows agents to achieve intermediate rewards for accomplishing goals that may not be inherently clear at a high level. This method of rewarding agents for intermediate actions has been shown to be effective in various scenarios [1], [17], [18]. Heuristic search can also be used in RL, with algorithms such as Monte-Carlo Tree Search and Portfolio Greedy Search showing promising results in various scenarios such as Silver et al.'s work in Go, Churchill et al. and Liu, Louis, and Nicolescu's analysis in SC2, and Churchill and Buro's large-scale combat in SC2 [9], [19]–[22]. Relational reinforcement learning (RRL), inspired by Muggleton and De Raedt [23] and defined by Zambaldi et al. [24], is an interesting and unique approach to RL that aims to represent states, actions, and policies using a relational language, which improves the generalizability of said features.

To our knowledge, the work that is most similar to ours with respect to our interpretation of agent influence maps is

the work done by Liu et al. in 2013 [20], [21]. In these works, potential fields (PF) and influence maps (IM) are defined using various unit parameters and are used to represent unit influence in the environment. These works, however, focus on the use of genetic algorithms (GA) and the comparison of GAs to heuristic search algorithms, whereas our work focuses solely on the use of DRL.

III. Methodology

In order to evaluate MAIDRL, we use the StarCraft Multi-Agent Challenge (SMAC) as our MARL research platform and conduct a series of experiments to compare the performance of MAIDRL with the centralized and decentralized methods. The SMAC environment provides a set of multi-agent micro-management challenges where the objective is to defeat your opponents with the given units.

A. Environmental Description

There are two types of observation spaces that we consider: local and global. The local observation space represents a decentralized perspective, where each agent has access to the information that it can observe in the local vicinity, while the global observation space is the centralized equivalent. The local observation is defined as the following attributes in the form of a one-dimensional vector for both allied and enemy units within sight range: distance, relative x, relative y, health, and unit type. All aforementioned features are provided by SMAC, where the distance is the euclidean distance between the observing and the observed agent, the relative x and y are the directional difference of the same pair of agents, and the health and unit type are of the observed agent. The global observation contains information regarding all units on the map with the relative positions to the center of the map along with all other features from the local observations, albeit from a global perspective, in addition to unit cooldowns and energy levels. All features are normalized by their maximum values in both the local and global observation spaces [4]. The actions that living units are allowed to take are a discretized subset of the full action set that is available in SC2. The rewards received are the default shaped rewards provided by the SMAC environment. Once an episode has finished, we calculate the discounted rewards with a decay rate of 0.9, then normalize and use the resulting reward per environmental step to train our models.

The SMAC combat scenarios that were used in training and evaluation are $3m$, $8m$, $25m$, and $8m\_vs\_9m$. Scenarios such as these can be represented with a Markov game, which is a multi-agent extension of Markov Decision Processes (MDPs) [1], [2], [10]. A Markov game with N agents comprises a set of states S that describe the properties of the agents and the environment, and a set of actions $A_1, A_2, ..., A_N$ and observations $O_1, O_2, ..., O_N$ for each of the N agents. Each of the agents treat the surrounding units as a part of their local information and perform an action in the environment based upon its own observation. $S \times A_1 \times ... \times A_N \rightarrow S'$ denotes the state transition from S to S' where each agent performs an action following a policy π in each environmental step. We used the $8m$ scenario in SMAC as the baseline experimental scenario where each team controls eight Marines to fight against each other. This scenario serves as a foundation for our research, as it has neither very large nor very small numbers of units, and it is representative of a medium-scale MARL scenario, while the other selected scenarios demonstrate the generalizability of the evaluated methods. Each of these scenarios are comprised of medium-ranged infantry units called Marines, with the number of Marines per team given in the name of the scenario. The objective of the agents in each scenario is to defeat the built-in SC2 game AI by eliminating all of the enemies without losing all of the allied troops. Allied agents must work together and focus their fire on enemies to efficiently reduce the amount of damage received while maintaining damage dealt over time by minimizing sustained casualties.

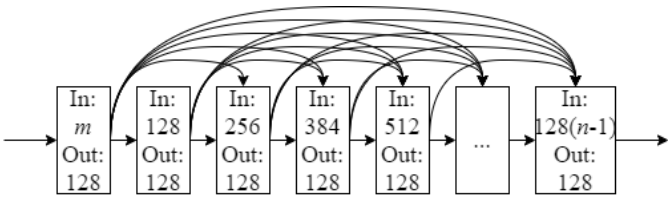


Fig. 1: Example DenseNet-style group of n 128-neuron layers.

B. General Experimental Features

Each explored MARL method was executed in 32 independent instances with corresponding random seeds. All experiments are executed for a duration of 3,000 episodes in their respective combat scenarios. Our experiments promote early exploration followed by incrementally increasing exploitation with an ε-soft approach that initializes epsilon to 1.0 and diminishes it to 0.0001 over the course of 30,000 environmental steps. In addition to ε-soft, we utilized A2C as the core learning algorithm in all of our experiments, with separate neural networks for the actor and critic. The size of the inputs to the actor and critic vary between explored MARL methods and combat scenarios. The size of the actor outputs layer is dependent upon the number of agents in the scenario, while the critic always maintains a single output neuron. The activation function used in all layers with the exception of the output layers is the Exponential Linear Unit (ELU) with $\alpha = 1.0$. The output layer of the actors used softmax, and the output layer of the critics used a simple linear activation. The actor and critic were compiled with losses categorical cross-entropy and mean squared error, respectively, and both use the $adam$ optimizer with a learning rate of 0.00001. The actor and critic are trained using the unique experiences of each of the agents at the end of an episode, meaning that in the case of the $8m$ scenario, the actor and critic learn from the experiences of 8 distinct perspectives of the environment, one for each ally.

C. Simple versus Dense A2C

Before considering the inclusion of the DenseNet-style model architecture, we explored the effectiveness of using a MAIM state representation in tandem with an ANN that

Algorithm 1: Create and Update Agent Influence Map

Result: Square agent influence array based on I_0, λ_I, and d_I, centered on the agent.

if *AIM not defined* **then**
  $N = 2 \times d_I + 1$
  $AIM = N \times N$ array of 0's
end
for $cell \in AIM$ **do**
  if $dist(cell, center) \leq d_I$ **then**
    $AIM[cell] = \lambda_I \times I_0$
  end
end

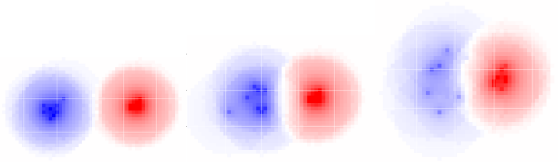
contained only a single hidden 1024-neuron layer. We used the $8m$ scenario in this experiment, noting that $8m$ was chosen as our baseline scenario due to its intermediate scale and difficulty. We improve the simple A2C by defining a DenseNet-style model architecture for both actor and critic [25]. Fig. 1 demonstrates an example of a DenseNet-style grouping of layers which is defined to be an arbitrary number of n layers such that the output of every layer is input to each of the following layers in the group via concatenation. Our network architectures contain two of such groups, each with five 128-neuron dense layers. The only additional layer in our DenseNet architecure is a 256-neuron dense layer that precedes the first DenseNet-style group of layers, immediately after the input layer.

D. Centralized versus Decentralized

We apply A2C in centralized, decentralized, and hybridized state scenarios to investigate how to effectively use global information for fine-grained decision-making in multi-agent environments. First, we apply A2C in a centralized scenario where each agent utilizes local and global observations to train the actor and critic networks for learning interaction among agents. This method is further used as the baseline for comparison with other algorithms. Second, we explored the option of a global critic with a local actor (GCLA), where the critic received the global information while the actor received the local observations of each agent. This particular method is unique from the other methods in that the critic is arguably centralized but not with respect to each agent, while the actor remains entirely decentralized. Third, we explored the option of total critic and a local actor (TCLA), where the critic is centralized, while the actor is decentralized. This method was inspired by the use of centralized training with decentralized execution as in the case of MADDPG [2].

E. Semi-centralized with Multi-Agent Influence Maps

In addition to the centralized and decentralized experiments, we propose a novel method to abstract the global features in such a way that is representative of how a human usually interpret them. We define an agent influence map that is determined by three values: the source influence I_0, the influence decay rate λ_I, and the range of influence d_I for each unit in the environment. Algorithm 1 demonstrates the creation and update process of each AIM. For our experiments, we set d_I equal to the range of the agent as defined by SC2, I_0 equal to the current relative health of the agent as defined by SMAC, and λ_I equal to the inverse of the distance from the agent, where a distance of 0 was assigned the value of I_0. To allow for a distinction between allied units and enemy units, we set I_0 of the allied units to the negative of the relative health provided by SMAC.



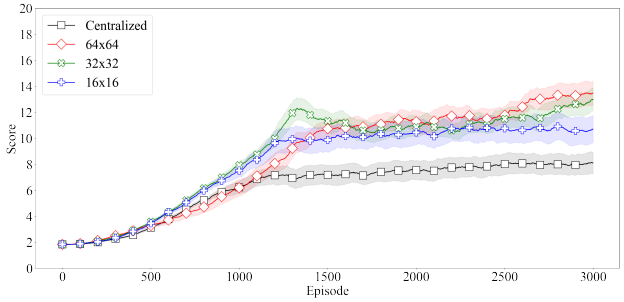
(a) $t = 1$ (b) $t = 5$ (c) $t = 9$

Fig. 2: Example MAIM at environmental step $t = 1, 5, 9$

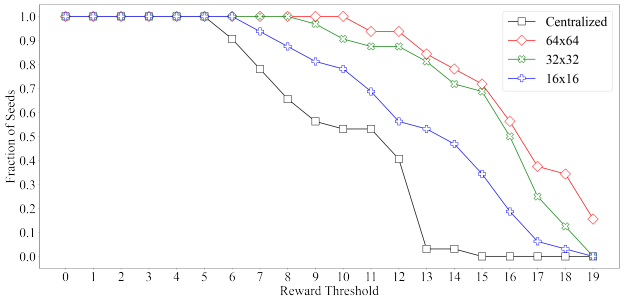
The AIM of each agent is aggregated into a multi-agent influence map, where the AIM is simply added to the MAIM based upon the agents position in the environment. The MAIM is a scaled representation of the units on the map in a SMAC scenario. For example, the $8m$ scenario has a map size of 32×32 units, but the MAIM dimensions can be any positive integers, and our implementation will automatically scale the AIM to fit the MAIM's dimension while also maintaining the scale of the agents influence on $8m$. We explored MAIMs of size 16×16, 32×32, and 64×64, each with the same AIM parameters. Fig. 2 demonstrates three snapshots in the form of a heatmap of a 64×64 MAIM where the blue is portraying allied units and red portrays enemy units. This is a sample MAIM generated from a random agent versus the built-in SC2 AI which shows that the built-in AI overwhelms the random agent. We can see on the MAIM that each team of units starts in separate clusters where the enemy units are scripted to move toward the spawn point of the allied units, so a conflict is almost guaranteed. We used this MAIM representation as a semi-centralized subset of the global state in our MAIRL and MAIDRL experiments, which was then flattened and used as inputs to our networks.

IV. Results and Discussion

We analyze the results of the MARL methods used with three 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 with respect to various thresholds, all with respect to the full set of 32 seeds. Henceforth, these metrics shall be referred to as overall performance, overall stability, and peak performance, respectively. We define the running average episode reward as the average reward of the most recent 50 episodes. We also perform more detailed evaluation regarding the peak performance per method across all seeds, considering the minimum, maximum, average, and standard deviation.

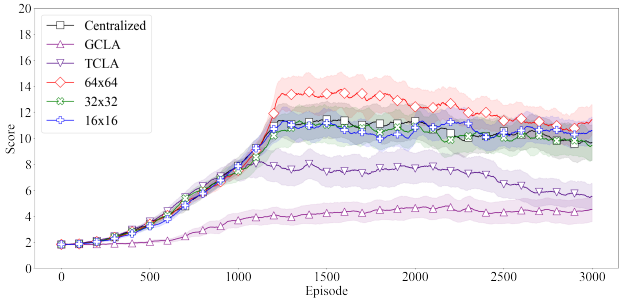


(a) Average of the Running Average Episode Reward

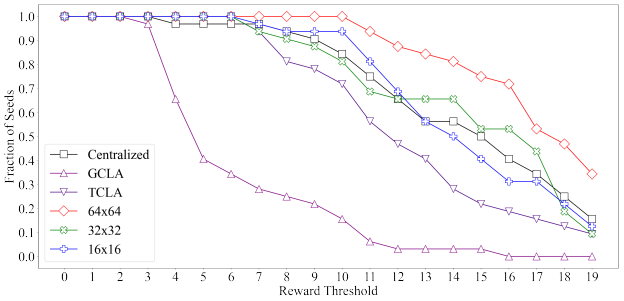


(b) % of Seeds that Achieve a Maximum Running Average Reward Greater Than or Equal to Various Thresholds

Fig. 3: Simple Architecture Results on $8m$



(a) Average of the Running Average Episode Reward



(b) % of Seeds that Achieve a Maximum Running Average Reward Greater Than or Equal to Various Thresholds

Fig. 4: DenseNet Architecture Results on $8m$

TABLE I: Running Average Reward Across all Seeds with Simple Architecture

Scenario	Method	Min	Max	Avg	Std
$8m$	Centralized	5.31	14.55	9.95	2.79
	64×64[a]	**10.30**	**19.58**	**16.13**	**2.64**
	32×32[a]	8.16	18.81	15.19	2.73
	16×16[a]	6.44	18.05	12.73	3.39

[a] MAIRL with given MAIM dimensions.

A. *Simple Architecture and MAIRL*

Fig. 3 and Table I display the results achieved by the use of the simple network architecture in the $8m$ scenario. It is evident that MAIRL considerably increased the overall and peak performance of the agents, as well as maintained comparable overall stability to the centralized method. Fig. 3a shows that each of the MAIRL variants achieved an overall performance around 11, while the centralized method only achieved 8. MAIRL is further shown to outperform the centralized method in terms of peak performance in Table I, with the 64×64 MAIRL variant achieving a 4.99 higher minimum, 5.03 higher maximum, and 6.18 higher average, all with a 0.15 lower standard deviation.

B. *DenseNet Architecture and MAIDRL*

Centralized: Due to the underwhelming performance of the centralized method with the simple network architecture, we considered ways to create a more competitive baseline with which to form a basis of comparison for our proposed methods. We found that the introduction of a DenseNet-style model architecture considerably improved the performance of the centralized method, raising the overall performance from 8 to more than 10, and raising the peak performance across seeds 5.45 points to a perfect maximum of 20, while simultaneously improving the average peak performance across seeds to 14.20, a 4.25 point improvement. Fig. 4b shows that this method maintains a high percentage of seeds that are capable of achieving running averages that are close to the upper limit, with almost 16% of all seeds achieving a maximum running average greater than 19. Additionally, Table II shows that the centralized method with the DenseNet architecture is capable of achieving a perfect running average score. The results of the DenseNet-style architecture with the centralized method suggest that the use of such a structure yields better results overall, with a moderate increase in standard deviation, though we argue that the general improvements in the other metrics outweigh this detriment. As such, the centralized DenseNet architecture is considered the baseline for comparison for each of the following methods, and said methods are applied in tandem with the same architecture.

GCLA: The GCLA method is shown in Fig. 4 to perform considerably worse with regard to both overall and peak performance, only achieving an overall performance of 4 and with no seeds managing to exceed a peak performance of 16 at any point. We hypothesize that these poor results are indicative of the incomplete information given to the critic. The maximum and average of the peak performance achieved in any seed was considerably lower than that of

TABLE II: Peak Running Average Reward Across all Seeds with DenseNet-style Architecture

Scenario	Method	Min	Max	Avg	Std
$8m$	Centralized	3.24	**20.00**	14.20	4.22
	GCLA	2.87	15.31	5.99	3.15
	TCLA	6.47	19.53	12.32	3.92
	64×64[a]	**10.35**	19.86	**16.84**	**2.99**
	32×32[a]	6.53	19.86	14.38	4.18
	16×16[a]	6.79	**20.00**	14.30	3.62

[a] MAIDRL with given MAIM dimensions.

the compared methods, as shown in Table II, only achieving 15.31 and 5.99, respectively. Because the actor learns from the advantage, which is determined in part by the critic, the lack of understanding of the local state for each agent on the part of the critic unsurprisingly leads to a poor performance overall. This hypothesis is supported by the results of the TCLA method, where the only change that was made is the provision of the local information in addition to the global.

TCLA: The TCLA method, inspired by [2], is shown in Fig. 4 to perform notably more poorly overall than the MAIDRL and centralized methods, peaking at a score around 8 at episode 1100, then steadily declining from there. TCLA performed somewhat comparably to the other methods with respect to the percentage of seeds at various thresholds, though it is consistently below all other curves for the majority of said thresholds, with the exception of GCLA. Table II further demonstrates the results shown in Fig. 4. The maximum of the peak performance across all seeds is 19.53, which is only 0.33 lower than the next highest and is a 4.22 improvement over the GCLA method, but the average of the maximum running average reward is only 12.32, which suggests that such a peak is less than common. The results show that, while TCLA is capable of performing well in the $8m$ scenario, it fails to do so consistently.

MAIDRL: Our semi-centralized MAIDRL method produced varying levels of results depending upon the size of the MAIM. The 16×16 and 32×32 MAIM MAIDRL variants both performed very similarly to the centralized baseline on each of the three metrics that we consider, though they both do fall just below the baseline in terms of peak performance greater than 19. The 64×64 MAIM significantly outperformed each of the other MARL methods that we tested, notably without a significant decrease in overall stability, maintaining a standard deviation just below 4. Fig. 4a demonstrates that the learning curve for the 64×64 MAIM grew to the highest point around 14 in a comparable amount of time as the other methods. While the steady decline thereafter is somewhat worrisome, we note that there is a distinct improvement with respect to the peak performance above 19, with a total of just over 34%. Furthermore, Table II demonstrates that each MAIDRL variant outperformed the baseline in every regard when considering the peak performance across all seeds, with the exception of the maximum, where the 16×16 MAIM variant matched the perfect score of the baseline, while the others fell short by 0.14 points.

TABLE III: Peak Running Average Reward Across all Seeds with DenseNet-style Architecture on Extended Scenarios

Scenario	Method	Min	Max	Avg	Std
$3m$	Centralized	4.00	18.79	14.41	4.19
	64×64[a]	3.74	**18.83**	11.48	5.04
	32×32[a]	5.89	18.80	14.86	4.10
	16×16[a]	**12.42**	18.76	**16.61**	**1.87**
$25m$	Centralized	**8.18**	13.27	**10.81**	1.35
	64×64[a]	7.20	12.73	10.08	1.50
	32×32[a]	5.34	**13.53**	9.82	1.77
	16×16[a]	7.97	13.20	10.26	**1.25**
$8m\_vs\_9m$	Centralized	**7.69**	10.68	**9.59**	**0.76**
	64×64[a]	6.25	**10.95**	9.21	1.20
	32×32[a]	7.05	10.67	9.33	0.98
	16×16[a]	6.55	10.12	9.0	0.99

[a] MAIDRL with given MAIM dimensions.

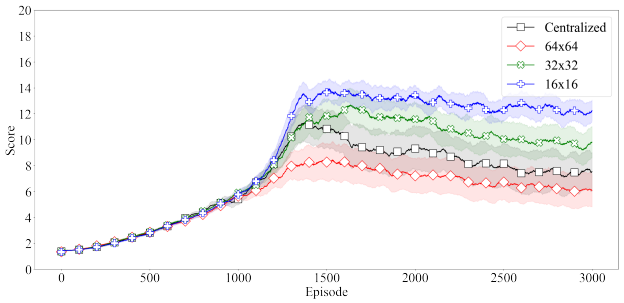


Fig. 5: DenseNet Architecture Results on $3m$

C. Generalizability of MAIDRL

In order to evaluate the generalizability of our approach, we further applied the DenseNet Architecture and MAIDRL on three new SMAC scenarios: $3m$, $25m$, and $8m\_vs\_9m$. Note that we are interested in the generalizability of our methodology itself, rather than that of the trained neural networks, so each scenario starts with fresh networks.

1) SMAC-$3m$: $3m$ is a similar scenario to $8m$, but the unit count on each team is reduced to three. We chose this scenario to explore the applicability of MAIDRL in combat scenarios of varying complexity. In theory, $3m$ should be simpler to learn than $8m$. A consistent trend that was found amongst the baseline and each MAIDRL variant is that none of them seemed to accomplish a peak performance greater than 19. We hypothesize that this is due to the somewhat different score distribution in $3m$ as a result of the lower number of units on the field.

Centralized: The most concerning phenomenon that was recorded by the centralized baseline was the notable decrease in overall performance to 8 after reaching the initial maximum of 11. The percentage of seeds that achieved various peak performance thresholds also behaved similarly as in the $8m$ scenario, but with a notably more drastic decline at the highest threshold, decreasing from nearly 35% of seeds achieving peak performance above 18, to none above 19. Table III further illustrates that the baseline method performed consistently in $3m$ and $8m$, with the most significant difference being in the maximum of the peak performance across all seeds.

MAIDRL: The MAIDRL variants performed much differ-

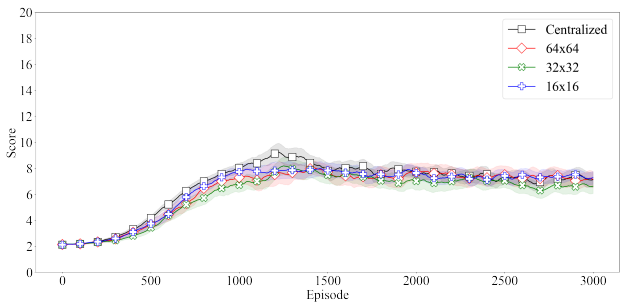


Fig. 6: DenseNet Architecture Results on $25m$

ently on $3m$ when compared to $8m$. In the $3m$ scenario, the 64×64 MAIM yielded the worst overall results, while it had previously shown to yield the best. The overall performance across seeds decreased considerably, as did the average peak performance per threshold, from 14 to 9 and 16.84 to 11.48, respectively. The most notable difference occurred in the 16×16 MAIM, which wholly outperformed the other variants, as well as the centralized method, achieving an overall performance of 14, with minimal decline thereafter. It further improved with respect to the standard deviation, leveling out at only 2. This is markedly better than the other MAIDRL variants, and better still compared to the centralized method, particularly when considering that it simultaneously achieved the highest overall performance. Table III shows that the 16×16 variant also dominated in regard to the peak performance across all seeds. Interestingly, the 64×64 variant achieved the highest peak overall, though less consistently than the other tested methods, which is reflected in the average of the peak performance across all seeds.

2) SMAC-$25m$ and SMAC-$8m\_vs\_9m$: The $25m$ and $8m\_vs\_9m$ scenarios were selected due to their highly challenging nature. $25m$ represents one of the largest combat scenarios that SMAC offers, with 25 marines on either team. As such, it is a *considerably* more challenging scenario to master. We chose the $25m$ scenario because it is representative of large-scale combat. While $3m$ and $8m$ represent small- and medium-scale respectively, they are both significantly smaller and less complex than $25m$. $8m\_vs\_9m$, unlike the symmetric scenarios we consider, gives a considerable advantage to the enemy forces, as they have an extra unit on their side. In order to win, a MARL system must demonstrate remarkable micromanagement techniques and be able to wholly outperform the built-in SC2 game AI to an overwhelming degree. While success in this scenario is indicative of a superior methodology, failure is not necessarily indicative of a poor methodology. This scenario is used deliberately as an unfair challenge to the tested methodologies. The difficulty of these scenarios are reflected universally across each of the tested methods in that neither the centralized nor the MAIDRL methods managed to achieve a peak performance greater than 14. We note here that the results of each method in $25m$ and $8m\_vs\_9m$ are similar enough that we only show $25m$ visually in Fig. 6.

Centralized: The centralized baseline method yielded a definite decrease in the overall performance as shown in Fig. 6, maintaining scores around 7-8. The standard deviation demonstrated a notable improvement over the $8m$ and $3m$ scenarios, settling around 2 for the large majority of the episodes, though we note that this is most likely a direct result of the overall performance being consistently lower than the aforementioned scenarios. The most significant indicator that the baseline failed to perform as well as it had in the smaller scenarios is apparent when considering that there were no seeds that managed to break a maximum running average greater than 14 and 11 in $25m$ and $8m\_vs\_9m$, respectively. This is not surprising, as the complexity of MAS increases with more or unfair numbers of agents, though it does show that this method was able to make progress.

TABLE IV: Welch's Unequal Variance t-test with null hypothesis $H_0 : \mu_{Centralized} \leq \mu_{MAIDRL}$ where μ is the average of the average episode reward over all episodes per seed

Scenario	Method	Test Statistic	P-Value	Power
$8m$	64×64[a]	-1.5045	0.9312	0.0001
	32×32[a]	0.2910	0.3860	0.1071
	16×16[a]	0.1013	0.4598	0.0662
$3m$	64×64[a]	1.6634	0.0507	0.7440
	32×32[a]	-1.9117	0.9697	0.0000
	16×16[a]	-4.4780	1.0000	0.0000
$25m$	64×64[a]	1.3098	0.0979	0.5663
	32×32[a]	2.5923	0.0071	0.9680
	16×16[a]	1.1923	0.1189	0.5018
$8m\_vs\_9m$	64×64[a]	1.2604	0.1062	0.5393
	32×32[a]	1.7636	0.0414	0.7865
	16×16[a]	2.4109	0.0095	0.9544

[a] MAIDRL with given MAIM dimensions.

MAIDRL: The $25m$ and $8m\_vs\_9m$ scenarios are the only scenarios that we tested whose results are not immediately apparent with respect to whether our semi-centralized MAIDRL method is as performant or better than the centralized baseline. Fig. 6 suggests that MAIDRL may be equivalent to the baseline, though we note that each MAIDRL variant fell below the baseline in terms of peak performance. Table III shows that, when comparing the peak performance across all seeds of each method, MAIDRL variants still achieved the highest maximum values in both $25m$ and $8m\_vs\_9m$, with values that are 0.26 and 0.27 better than the respective baselines.

D. Centralized versus MAIDRL Discussion

Table IV contains the results of performing Welch's Unequal Variances t-test on the averages of the running average episode reward comparing the baseline method to the MAIRL variants, as well as the calculated statistical power of the test. We use Welch's t-test in lieu of Student's because of the underlying assumption of Student's t-test that the variances of the samples are equivalent.We note here that the null hypothesis used in all of the t-tests is $H_0 : \mu_{Centralized} \leq \mu_{MAIDRL}$, with $\alpha = 0.05$ as the chosen level of significance. The power of the test is representative of the probability that we would accept the alternative hypothesis, if it were true.

From the table, it is clear that in $8m$, the overall performance of each MAIDRL variant is shown statistically to

be greater than or equal to that of the centralized baseline, with each p-value much larger than any common level of significance, albeit with small powers. Furthermore, this trend continues in the $3m$ scenario, with even the poorly performing 64×64 MAIDRL variant demonstrating statistical significance. The $25m$ scenario shows that the 64×64 and 16×16 MAIDRL variants tested fulfill this same condition, while the 32×32 variant definitively does not come close to our selected level of significance, though we note that it is close to the commonly selected 0.01. Finally, the $8m\_vs\_9m$ scenario shows that the 64×64 MAIDRL variant fulfills the condition for statistical significance, the 32×32 variant comes close to our selected level of significance, and the 16×16 variant is very close to the common 0.01 level of significance. When considering this statistical analysis, we claim that our semi-centralized MAIDRL method is statistically shown to be capable of matching or outperforming the overall performance of a centralized alternative in MAS of varying complexities.

V. Conclusion and Future Work

In this article, we introduced MAIRL and MAIDRL, novel semi-centralized methodologies to solve various MARL scenarios using abstracted feature information in the form of MAIMs and a robust model architecture inspired by DenseNet. MAIDRL was demonstrated in SMAC combat scenarios of varying complexity to be statistically capable of matching or bettering the overall performance of a comparable centralized baseline, even in scenarios with large or unfair numbers of agents. It also demonstrated significant improvements in the overall performance, overall stability, and peak performance shown in some of said scenarios, with MAIDRL variants being the top performers by a large degree in both small- and medium-scale MARL scenarios. We hypothesize that the varying results of the considered MAIM resolutions may suggest a positive correlation between the complexity of the environment and the optimal MAIM resolution, though more research is required to verify this hypothesis.

While we did not demonstrate full comprehension of all scenarios by MAIDRL, there are several aspects that can be improved in our future work. First, a logical next step would be to incorporate the use of convolutional neural networks (CNN) in our network architectures. The MAIM representation of features lends itself very well to CNNs and would allow for a more intelligent interpretation of the MAIM by the networks. Second, we plan to use multiple MAIMs to represent various features. We focused on the relative health of the unit as described in Section III-E, but there are other features that could be used as well, e.g. cooldown or possible damage per second. These various feature MAIMs might be considered as channels by the CNN, allowing the interpretation of the relationships between them. Finally, there is need to investigate the generalizability of MAIDRL in heterogeneous scenarios. We demonstrated generalizability in a range of homogeneous scenarios, but we acknowledge that the application of MAIDRL in heterogeneous scenarios is a necessary consideration for future work.

References

[1] J. Foerster, G. Farquhar, T. Afouras, N. Nardelli, and S. Whiteson, "Counterfactual multi-agent policy gradients," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[2] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent Actor-Critic for mixed cooperative-competitive environments," *arXiv preprint arXiv:1706.02275*, 2017.

[3] D. Xie and X. Zhong, "Semicentralized deep deterministic policy gradient in cooperative StarCraft games," *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

[4] M. Samvelyan *et al.*, "The StarCraft multi-agent challenge," *CoRR*, vol. abs/1902.04043, 2019.

[5] V. R. Konda and J. N. Tsitsiklis, "Actor-Critic algorithms," in *Advances in Neural Information Processing Systems*. Citeseer, 2000, pp. 1008–1014.

[6] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *International Conference on Machine Learning*. PMLR, 2014, pp. 387–395.

[7] I. Mordatch and P. Abbeel, "Emergence of grounded compositional language in multi-agent populations," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[8] V. Mnih *et al.*, "Asynchronous methods for deep reinforcement learning," in *International Conference on Machine Learning*. PMLR, 2016, pp. 1928–1937.

[9] D. Silver *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[10] P. Peng *et al.*, "Multiagent bidirectionally-coordinated nets: emergence of human-level coordination in learning to play StarCraft combat games," *arXiv preprint arXiv:1703.10069*, 2017.

[11] J. Schulman, S. Levine, P. Abbeel, M. Jordan, and P. Moritz, "Trust region policy optimization," in *International Conference on Machine Learning*. PMLR, 2015, pp. 1889–1897.

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

[13] M. Bain and C. Sammut, "A framework for behavioural cloning," in *Machine Intelligence 15*, 1995, pp. 103–129.

[14] 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, pp. 74–78, 2021.

[15] A. Amiranashvili, N. Dorka, W. Burgard, V. Koltun, and T. Brox, "Scaling imitation learning in Minecraft," *arXiv preprint arXiv:2007.02701*, 2020.

[16] O. Vinyals *et al.*, "Grandmaster level in StarCraft II using multi-agent reinforcement learning," *Nature*, vol. 575, no. 7782, pp. 350–354, 2019.

[17] A. Y. Ng, D. Harada, and S. Russell, "Policy invariance under reward transformations: theory and application to reward shaping," in *ICML*, vol. 99, 1999, pp. 278–287.

[18] C. Berner *et al.*, "Dota 2 with large scale deep reinforcement learning," *arXiv preprint arXiv:1912.06680*, 2019.

[19] D. Churchill, Z. Lin, and G. Synnaeve, "An analysis of model-based heuristic search techniques for StarCraft combat scenarios," in *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, vol. 13, no. 1, 2017.

[20] 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*. IEEE, 2013, pp. 1371–1378.

[21] ——, "Using CIGAR for finding effective group behaviors in rts game," in *2013 IEEE Conference on Computational Intelligence in Games (CIG)*. IEEE, 2013, pp. 1–8.

[22] D. Churchill and M. Buro, "Portfolio greedy search and simulation for large-scale combat in StarCraft," in *2013 IEEE Conference on Computational Intelligence in Games (CIG)*. IEEE, 2013, pp. 1–8.

[23] S. Muggleton and L. De Raedt, "Inductive logic programming: theory and methods," *The Journal of Logic Programming*, vol. 19, pp. 629–679, 1994.

[24] V. Zambaldi *et al.*, "Relational deep reinforcement learning," *arXiv preprint arXiv:1806.01830*, 2018.

[25] G. Huang, Z. Liu, L. Van Der Maaten, and K. Q. Weinberger, "Densely connected convolutional networks," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 4700–4708.