Grid-Wise Control for Multi-Agent Reinforcement Learning in Video Game AI

Lei Han [*1] **Peng Sun** [*1] **Yali Du** [*2 1] **Jiechao Xiong** [1] **Qing Wang** [1] **Xinghai Sun** [1] **Han Liu** [3] **Tong Zhang** [4]

Abstract

We consider the problem of multi-agent reinforcement learning (MARL) in video game AI, where the agents are located in a spatial grid-world environment and the number of agents varies both within and across episodes. The challenge is to flexibly control an arbitrary number of agents while achieving effective collaboration. Existing MARL methods usually suffer from the trade-off between these two considerations. To address the issue, we propose a novel architecture that learns a spatial joint representation of all the agents and outputs grid-wise actions. Each agent will be controlled independently by taking the action from the grid it occupies. By viewing the state information as a grid feature map, we employ a convolutional encoder-decoder as the policy network. This architecture naturally promotes agent communication because of the large receptive field provided by the stacked convolutional layers. Moreover, the spatially shared convolutional parameters enable fast parallel exploration that the experiences discovered by one agent can be immediately transferred to others. The proposed method can be conveniently integrated with general reinforcement learning algorithms, e.g., PPO and Q-learning. We demonstrate the effectiveness of the proposed method in extensive challenging multi-agent tasks in StarCraft II.

1. Introduction

Reinforcement learning (RL) has gained great successes in solving many challenging problems recently, such as game playing (Mnih et al., 2015; Silver et al., 2016; 2017), robotics (Kober et al., 2013), resource management (Mao et al., 2016), etc. Among these, RL in game AI research attracts increasing attentions with recent achievement in Go (Silver et al., 2016), DOTA2 (OpenAI, 2018) and StarCraft II (Deepmind, 2019), etc. An important application in both virtual and real worlds is to use RL to build collaborative multi-agent systems. Examples in virtual games are DOTA (OpenAI, 2018), StarCraft (Vinyals et al., 2017) and multi-player ViZDoom (Kempka et al., 2016; Luo et al., 2018), etc., while real-word examples include multi-robot control (Matignon et al., 2012; Devin et al., 2017; Foerster et al., 2018), traffic light control (Wiering, 2000; Arel et al., 2010), bidding and advertising (Jin et al., 2018).

There have been extensive studies focusing on multi-agent problems and generally three types of methods are used to train multiple agents simultaneously: centralized learning, decentralized learning and a mixture between the two.

Centralized learning method takes input as the global state information or the union of each agent's local observation, performs a joint action and then receives a team reward. Working in the joint action space, centralized learning naturally handles the problem of coordination among agents and traditional RL algorithms can be directly applied for training. Unfortunately, the exponentially expanded joint action space makes fully centralized learning impractical even when the number of agents slightly increases (Busoniu et al., 2008).

On the other side, when the state information is partially observed by the agents, decentralized learning is a natural way to learn each agent's policy based on its own local observation-action trajectory. For example, independent Q-learning (Tan, 1993) learns the state-action value function of each agent independently. However, from the perspective of any individual agent, the algorithm becomes unstable due to the non-stationarity arising from other simultaneously acting agents in the environment. Therefore, decentralized learning usually needs to pay more effort to model the communication among agents. Moreover, fully decentralized learning requires independent rewarding scheme while in many scenarios all the agents share a global team reward and assigning explicit individual scores is impractical.

To address the problems encountered in both centralized and decentralized learning, a number of mixture methods have been developed. For example, some approaches learn

[*]Equal contribution [1]Tencent AI Lab, Shenzhen, Guangdong, China [2]University of Technology Sydney, Sydney, NSW, Australia [3]Northwestern University, IL, USA [4]Hong Kong University of Science and Technology, Hong Kong, China. Correspondence to: Lei Han <leihan.cs@gmail.com>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

a centralized critic or state-action value function with decentralized executions (Foerster et al., 2017a; Lowe et al., 2017) or reversely train a centralized policy for all the agents with per-agent critics (Gupta et al., 2017), or use a centralized network architecture with multiple action outputs (Sukhbaatar & Fergus, 2016; Peng et al., 2017). The spirits in these mixture methods are (1) reducing the large joint action space, (2) acquiring other agents' dynamics to model the non-stationarity and (3) finding effective strategies to achieve communication or credit assignment among agents.

For many multi-agent settings, the number of agents acting in the environment keeps changing both within and across episodes. For example, in video games, the agents may die or be out of control while new agents may join in, e.g., the battle game in StraCraft. Similarly, in real-world traffic, vehicles enter and exit the traffic network over time, inducing complex dynamics. Therefore, a main challenge is to flexibly control an arbitrary number of agents and achieve effective collaboration at the same time. Unfortunately, all the aforementioned MARL methods suffer from trading-off between centralized and decentralized learning to leverage agent communication and individual flexibility. Actually, most of the existing MARL algorithms make a default assumption that the number of agents is fixed before learning. Many of them adopt a well designed reinforcement learning structure, which, however, depends on the number of agents.

To address this issue, assuming the state information can be organized into a grid map structure, we propose a novel architecture that learns a spatial joint representation of all the agents and outputs an action per grid. This is similar to the image segmentation task, in which per-pixel labels are predicted. Once a grid node is occupied with an agent, the agent will be controlled independently with the action predicted for that grid. Otherwise, empty grids will be masked out from the joint policy and hence the complexity of this architecture is independent of the resolutions of the grid map. Specifically, we use an encoder-decoder network as the policy that outputs grid-wise actions, and we learn a centralized critic conditioning on the joint representation learned from the encoder. The proposed architecture is referred to as GridNet. There are some benefits of using this encoder-decoder architecture. GridNet allows flexible control over an arbitrary number of agents as long as they are located in the spatial environment. It also naturally handles agent collaboration, because the stacked convolutional layers can provide sufficiently large receptive field for the agents to communicate. The GridNet also enables fast parallel exploration, because experiences from one agent are immediately transferred to others through the shared convolutional parameters. We show that GridNet can be conveniently integrated with many general reinforcement learning algorithms. We study the performance of the GridNet method in many challenging multi-agent tasks in battle games of StarCraft II (Vinyals et al., 2017). We report extensive results comparing GridNet with a number of state-of-the-art MARL methods and provide intuitive case studies for the learned GridNet policy. Results show that the GridNet can learn very strong policies in many complex battle tasks.

2. Related Work

The simplest MARL approach to train multiple agents is learning them independently. This was first attempted with Q-learning (Tan, 1993), which was then extended with deep neural networks applied (Watkins, 1989; Tampuu et al., 2017). The fully decentralized methods are problematic, since from the perspective of an individual agent the environment is non-stationary due to other agents' dynamics. To overcome this problem, existing methods have tried to infer other agents' policies and involve them in the Q function (Lowe et al., 2017), or use importance sampling to correct bias in the experience replay (Foerster et al., 2017b).

For centralized methods, existing approaches have studied using specifically designed policy structures. For example, some policy networks may involve communication layers or modules and directly output multi-head predictions for the agents (Sukhbaatar & Fergus, 2016; OpenAI, 2018). Recurrent neural networks (RNN) have also been introduced to enable a sequence of agents to communicate (Peng et al., 2017). Using RNN allows the number of agents to change, however permutations in the agent ordering will affect the performance of the trained policy, which is unstable.

Another branch of MARL methods proposes to learn a centralized critic that can distinguish the contributions among the agents. The counterfactual multi-agent policy gradient (Foerster et al., 2017a) uses a counterfactual baseline to assign credits for the agents. The value decomposition network (Sunehag et al., 2017) decomposes the centralized value into a sum of individual agent values to discriminate their contributions. The QMIX (Rashid et al., 2018) method adopts a similar idea that assumes the centralized value function is monotonically increasing of the individual values.

However, when considering situations with a varying number of agents, all these methods become problematic. Our approach differs from these works and is specifically devised to solve the considered issues.

3. Background

3.1. Multi-Agent Reinforcement Learning

The considered multi-agent system can be described as a tuple as $\langle \mathcal{A}, S, U, P, r, \gamma, \rho_0 \rangle$. At each time step t, $\mathcal{A} = \{1, 2, \cdots, n_t\}$ is the set of agents; $s_t \in S$ is the global state of the environment; $U = U_1 \times U_2 \times \cdots \times U_{n_t}$ is the joint action space, U_i is the action space of agent i and

$\mathbf{u}_t = \{u_t^i\}_{i=1}^{n_t} \in U$ is the joint action taken by the agents at time step t; $P(s_{t+1}|s_t, \mathbf{u}_t) : S \times U \times S \to [0, 1]$ is the state transition function; $r(s_t, \mathbf{u}_t) : S \times U \to \mathbb{R}$ is the reward function shared by all the agents; $\gamma \in [0, 1)$ is a discount factor and $\rho_0 : S \to \mathbb{R}$ is the distribution of the initial state s_0. As we can observe, when the number of agents increases, the cardinality of the joint action space U increases exponentially.

Let $\pi(\mathbf{u}_t|s_t) : S \times U \to [0, 1]$ be a stochastic joint policy and let $\eta(\pi) = \mathbb{E}_{s_0,\mathbf{u}_0,\cdots}[R_0]$ with $R_t = \sum_{l=0}^{\infty} \gamma^l r_{t+l}$ denoting the expected discounted reward, where $s_0 \sim \rho_0(s_0)$, $\mathbf{u}_t \sim \pi(\mathbf{u}_t|s_t)$, and $s_{t+1} \sim P(s_{t+1}|s_t, \mathbf{u}_t)$. At time step t, the joint state-action value is defined as $Q_\pi(s_t, \mathbf{u}_t) = \mathbb{E}_{s_{t+1},\mathbf{u}_{t+1},\cdots}[R_t]$ and the state value function is $V_\pi(s_t) = \mathbb{E}_{\mathbf{u}_t,s_{t+1},\cdots}[R_t]$. We aim to find an optimal joint policy π^* that achieves the maximum expected reward $\eta(\pi^*)$.

3.2. Q-Learning and Actor-Critic Algorithms

Q-learning is one of the most popular RL methods that directly learns the state-action value function $Q_\pi(s, \mathbf{u})$ for policy π. It can be expanded recursively as $Q_\pi(s, \mathbf{u}) = \mathbb{E}_{s'}[r(s, \mathbf{u}) + \gamma \mathbb{E}_{\mathbf{u}' \sim \pi} Q_\pi(s', \mathbf{u}')]$, where s' is the successive state. For a parameterized Q-function, the optimal value function Q^* can be optimized by minimizing

$$L(\theta) = \mathbb{E}_{s,\mathbf{u},s'}\left[\left[Q^*(s, \mathbf{u}|\theta) - (r + \gamma \hat{Q}^*(s', \mathbf{u}'))\right]^2\right], \quad (1)$$

where $\hat{Q}^*$ is a target Q-function periodically updated by the most recent parameter θ. Q-learning has also been applied to multi-agent settings (Tesauro, 2004; Foerster et al., 2016).

Actor-critic (AC) algorithms are another type of commonly adopted RL methods. AC algorithms use a parameterized policy π_θ and update its parameters by directly maximizing the expected reward $J(\theta) = \mathbb{E}_{s,\mathbf{u}}[R]$ using the policy gradient

$$\nabla_\theta J(\theta) = \mathbb{E}_{s,\mathbf{u}}[\nabla_\theta \log \pi_\theta(\mathbf{u}|s) A_\pi(s, \mathbf{u})], \quad (2)$$

where $A(s, \mathbf{u})$ is the critic. There exists several ways to estimate $A(s, \mathbf{u})$, resulting in a variant of AC algorithms (Williams, 1992; Sutton et al., 1998; Schulman et al., 2015; 2017).

3.3. Encoder-Decoder Network

As introduced previously, we will employ a convolutional encoder-decoder network that outputs control signal per grid, which is reminiscent of the image segmentation task. In the Computer Vision community, there has been a long-standing pursue for image segmentation, where it aims at predicting a label for each pixel (e.g., background, person, car, etc.). With the advent of deep learning renaissance (Hinton & Salakhutdinov, 2006; Krizhevsky et al., 2012), the convolutional encoder-decoder based methodology (Long et al., 2015) becomes popular in recent literature and show significant improvement over prior work. Representative contemporary methods include (Noh et al., 2015; Chen et al., 2018; Sandler et al., 2018). Roughly, the encoder serves as a learnable feature extractor, where the conventional convolution/pooling layers are stacked to summarize the entire image as an intermediate representation. The decoder then "translates" the intermediate representation into per-pixel labels via 1×1 convolution. Note the intermediate representation can be either in original image size or in reduced image size. In the latter case, the decoder must involve additional upsampling layers to recover the original image size, where commonly adopted layers include stride deconvolution, unpooling, bilinear upsampling, etc.

In this paper, we investigate both the original-sized and the reduced-sized settings for the proposed GridNet. The original sized feature map maximally preserves the spatial information, yielding satisfactory performance when the size is small. However, when the image size is large, it becomes infeasible since (1) the stacked convolutional layers without down-sized pooling have limited receptive field over the original image, which is risky for an agent to neglect other agents in its neighborhood; (2) the repeated convolutions over the original large image size incur expensive time cost during both training and inference. Therefore, we also try the intermediate feature map in reduced size. In this case, we employ a decoder architecture similar to the DeepLabV3 (Chen et al., 2017), where parallel atrous convolutional layers are combined and followed by a bi-linear upsampling.

4. Grid-Wise Control

Assume the global state s can be represented as a grid feature map with size (w, h, c_s), where (w, h) is the scale of the grid map and c_s is the number of feature channels. Define an action map $\boldsymbol{a} \in \tilde{U}$ with size (w, h, c_a) and action space $\tilde{U}$, where c_a indicates the action dimension of the agents. GridNet predicts an action $a_{i,j}$ for each grid (i, j) in the action map, no matter whether there is an agent located at (i, j) or not. The action map space is $\tilde{U} = \Pi_{1 \le i \le w_a, 1 \le j \le h_a} U_{i,j}$ with $U_{i,j}$ indicating the action space at grid (i, j). Then, the original multi-agent action space is $U \subseteq \tilde{U}$. At first glance, the amplified action space $\tilde{U}$ further enlarges the exponentially expanded joint action space U, making the learning and exploration much more difficult. However, as we will show in the following, the expanded action space can be significantly reduced thanks to the local connectivity and parameter sharing from the convolutional layers.

4.1. The Encoder and Decoder

The GridNet architecture consists of a pair of encoder and decoder which are depicted in the dashed rectangle in Fig. 1. The GridNet model takes an image-like input state

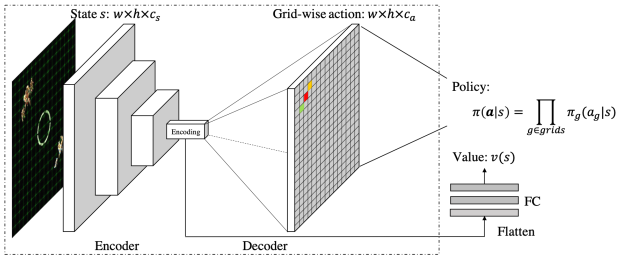


Figure 1. Illustration of the grid-wise control architecture. The policy network is indicated in the dashed rectangle. The policy function and value function are depicted on the right-hand side.

$s \in \mathbb{R}^{w \times h \times c_s}$. Each grid (i, j) for $1 \leq i \leq w, 1 \leq j \leq h$ indicates a location in the spatial environment. The channel dimension characterizes the agent and environment attributes. For example, if an agent is located at $(4, 2)$, then the vector $s(4, 2, :) \in \mathbb{R}^{c_s}$ describes the agent's c_s attributes. For a blank grid (i, j), we fill the vector $s(i, j, :)$ with zeros. The state s is fed into an encoder that is a convolution network consisting of several convolutional and pooling layers, producing a high-level, joint feature map representation. Then, the feature map is processed by a decoder consisting of convolutional layers and optional upsampling layers, constructing the action map $\boldsymbol{a} \in \mathbb{R}^{w \times h \times c_a}$. The channel dimension of $\boldsymbol{a}$ indicates the action set. For example, if the agent takes action from set $U = \{1, 2, ..., 5\}$, then the vector $a(i, j, :) \in \mathbb{R}^5$ represents either logits or normalized probabilities over set U. An agent located in, for example, the grid $(4, 2)$ will take action according to $a(4, 2, :)$. Similarly, if the action spaces are continuous, each channel will indicate a specific action with continuous values. For heterogeneous agents, we let the channel dimension be the disjoint union of all agents' action spaces.

With the above encoder-decoder network, given any state s it is easy to perform a forward pass and output the grid-wise actions for the agents. The next section will discuss how to perform backpropagation.

4.2. Integration with RL Algorithms

The GridNet architecture is convenient to be integrated with many state-of-the-art RL algorithms. We will use actor-critic method as example. Defining the joint policy as the product of the probabilities at each corresponding grid as

$$\pi(\boldsymbol{a}|s) = \prod_{g \in \text{grids}} \pi_g(a_g|s), \tag{3}$$

we use the policy gradient defined in Eq. (2) to update the actor, where we choose $A_\pi(s, \boldsymbol{a}) = r(s, \boldsymbol{a}) + v_\varphi(s') - v_\varphi(s)$ as the advantage function (Schulman et al., 2015; 2017) and φ denotes the parameter of the value function v. The actor-critic method is represented by the right-hand side of Fig. 1. In this paper, we will adopt the actor-critic framework. Nevertheless, we still provide a solution that integrates GridNet with state-action learning method below.

It is not straightforward to apply Q-learning with GridNet directly, because the state-action value defined on an action map $\boldsymbol{a}$ is not easy to be formulated. Instead, we can follow the idea of the deterministic policy gradient (DPG) methods (Silver et al., 2014; Lillicrap et al., 2015) to maintain a parameterized actor $\mu_\phi(s)$ and a state-action value function $Q_\theta(s, \boldsymbol{a})$. Then, $Q_\theta(s, \boldsymbol{a})$ can be optimized by Eq. (1), and the deterministic policy can be updated by following the deterministic policy gradient

$$\nabla_\phi J = \mathbb{E}_s \left[\nabla_{\boldsymbol{a}} Q_\theta(s, \boldsymbol{a})|_{\boldsymbol{a}=\mu(s)} \nabla_\phi \mu_\phi(s) \right], \tag{4}$$

where θ could also be a convolutional neural network that outputs a Q value. Although the above method follows an actor-critic style, it learns the state-action value and can be applied to deal with continuous action space for each agent.

4.3. Discussion

We provide a brief discussion of the advantages of the proposed GridNet method in this subsection.

- Communication and collaboration: the GridNet naturally enables the communication among the agents. The stacked convolutional and/or pooling layers provide a reasonably large receptive field, making each agent be aware of other agents in its neighborhood. In this way, the spatial correlation of all the agents is naturally captured, yielding strong feature representation and henceforth accurate per grid control.
- Fast parallel exploration: at first glance, the joint space of the grid-wise action is very large because each grid of the action map takes values in the agent action space, and exploration could become problematic. However, by taking a step deeper, the convolutional parameters are shared by all the agents and once an agent takes a beneficial action during its own exploration, the other agents will acquire the knowledge as well. As we will show in the experiments, some strong tactics can be quickly learned by all the agents.
- Transferrable policy: the trained centralized GridNet policy is easy to be transferred to different multi-agent settings with a various number of agents as long as they act in the grid map. This benefits from the structure of the GridNet. In the experiments, we will design specific settings to show that the policy trained in an environment with fewer agents can be easily transferred to new tasks with much more agents.

5. Experiments

In this section, we perform extensive experimental studies over many challenging battle games in StarCraft II based

(a) 5I (b) 3I2Z (c) MAB

Figure 2. The battle scenarios for 5I, 3I2Z and MAB.

on the learning environment SC2LE (Vinyals et al., 2017). StarCraft II is a famous real-time strategy game with annual career league among professional human players.

5.1. Settings

For the game environment, we consider the following symmetric battle scenarios:

- 5 Immortals vs. 5 Immortals (5I), where Immortal is a range unit of Protoss that can attack enemies at a distance;
- (3 Immortals, 2 Zealots) vs. (3 Immortals, 2 Zealots) (3I2Z), where Zealot is a melee unit of Protoss that attacks enemies standing close to it;
- Mixed army battle (MAB), in which the battle is taken place among a random number of mixed units of Zerg, containing Baneling, Zergling, Roach, Hydralisk and Mutalisk. These units will be explained in details in the supplementary material.

Fig. 2 illustrates the 3 scenarios. For all the 3 settings, we randomly initialize the agent status, including health points, shield, cooling down and locations[1], in the training phrase to provide sufficient diversity. For testing, we remove most of the randomness and only keep randomized locations. Although GridNet is devised for scenarios with a varying number of agents, we still consider the settings 5I and 3I2Z which fix the initial number of agents. This is because we will compare the GridNet method with several state-of-the-art MARL methods, which cannot work with a varying number of agents. It needs to be clarified that for battle games, e.g., any settings mentioned above, agents might die once the combat takes place, and the number of agents will decrease (however it will not increase for 5I and 3I2Z). To enable other compared MARL methods to work in such settings, we will append fake observations with some additional features to indicate whether an agent is dead or not, which has been adopted in (Foerster et al., 2017a;b).

The observation space of the agents contains a global grid feature map which will be detailed for each scenario in the supplementary material. GridNet only takes this feature map as input. For the other compared methods, in addition to the feature map we also provide them the individual status vector features. The action space of each agent in 5I and 3I2Z contains 9 discrete actions with 8 move directions and the action of attacking the nearest enemy. The action space of the agents in MAB contains 10 discrete actions with 8 move directions, attacking the nearest ground unit in priority, and attacking the nearest air unit in priority[2].

In addition to the diversified environmental settings, we also study different training strategies by switching the opponent policy:

- Random agents (Rand): the opponent agents take random actions;
- Attack-Nearest (AN): the opponent agent chooses its nearest enemy to attack;
- Hit-and-Run (HR): a tactic commonly used by human players that will 1) attack first and then move during weapon cooling down; 2) run away when the unit is weak and fight back when safe;
- Self-play (SP): training with self historical models.

We will train all the methods under all the training strategies. Evaluations will compare all the trained policies in terms of winning rate against Rand, AN and HR, and the winning rate against each other among the trained policies.

5.2. Compared Methods

We compare a number of multi-agent reinforcement learning algorithms, including

- independent Q-learning (IQL) (Tan, 1993), which trains decentralized Q-functions for each agent. Since the observation and action spaces of the agents are the same within a specific environmental setting, a policy will be shared across all the agents;
- independent actor-critic (IAC) (Foerster et al., 2017a), which, similar to IQL, is an advantage actor-critic method that learns a shared policy to all the agents;
- decentralized policy with centralized critic (Central-V) (Foerster et al., 2017a), which trains a decentralized shared policy for all the agents while using a centralized advantage based critic;
- CommNet (Sukhbaatar & Fergus, 2016), which is a centralized policy that takes the joint observation of

[1] The locations of the agents in the same team will be randomized within a certain region, and the two teams are always split away from each other at the beginning.

[2] Mutalisk is an air unit. Please refer to the supplementary material for the descriptions of the units appeared in the experiments.

the agents as input and output multi-head actions for the agents. The network is designed to let the agents communicate through some neural network layers;

- GridNet: the proposed method with an actor-critic implementation as shown in Fig. 1.

Despite IQL which uses the Q-learning algorithm, all the other methods use PPO (Schulman et al., 2017) for training. All the methods use a distributed learning architecture that is similar to IMPALA (Espeholt et al., 2018). Among these methods, Central-V and CommNet only work with a fixed number of agents, because the centralized value function in Central-V and the communication structure in CommNet take fixed input size that is dependent on the number of agents. In order to provide a fair comparison, we will evaluate all the above methods on 5I and 3I2Z, where the maximum number of agents is fixed and use the trick of additional features. However, for the MAB setting, Central-V and CommNet are not able to work and we will not report results for them. The detailed network structures and the training settings are provided in the supplementary material.

5.3. Results on 5I and 3I2Z

5.3.1. Performance

For each method, we store the trained policies every 100 training batches and evaluate each stored policy over 100 testing episodes to report the average test winning rate. The results are given in Fig. 3. As we can observe, under all the settings, the independent learning methods, IQL and IAC, fail to learn good policies for multiple agents, since the methods do not take any collaboration among the agents into account. For both scenarios 5I and 3I2Z with Rand opponent, we see that GridNet, Central-V and CommNet can achieve almost 100% test winning rate after training over very few steps. When training against AN and HR, which are two stronger handcraft policies, Central-V obtains better performance compared with CommNet, while GridNet shows significant improvement over all the other MARL methods, reaching nearly 100% winning rate on 5I and above 80% winning rate on 3I2Z.

In addition to the test winning rate in the original scenarios, we provide a more straightforward comparison among all the methods. That is, we let them compete against each other to vote for a champion. For each method, 4 policies have been trained over Rand, AN, HR and SP. At the first stage, we let these 4 policies from the same method compete and select the best one (whose winning rates against others are all above 50%). The detailed results of the first stage are given in the supplementary material. It is worth mentioning that for the GridNet method, the strongest policy is the one trained from self-play. Then, at the second stage, we let the strongest policy selected from each method compete.

Table 1. Cross combat winning rate over the selected strongest policies of the 5 methods on 5I. The winning rates are for "row against column". Any pair of symmetric values sum to 1.

	IAC	IQL	Central-V	CommNet	GridNet
IAC	–	0.43	0.14	0.05	**0.00**
IQL	0.57	–	0.39	0.08	**0.06**
Central-V	0.86	0.61	–	0.52	**0.27**
CommNet	0.95	0.92	0.48	–	**0.01**
GridNet	**1.00**	**0.94**	**0.73**	**0.99**	–

Table 2. Cross combat winning rate over the selected strongest policies of the 5 methods on 3I2Z. The table format follows that used in Table 1.

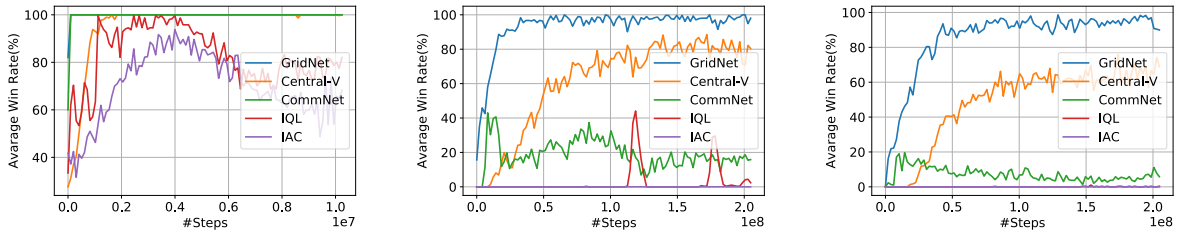
	IAC	IQL	Central-V	CommNet	GridNet
IAC	–	0.60	0.24	0.20	**0.00**
IQL	0.40	–	0.17	0.24	**0.04**
Central-V	0.76	0.83	–	0.29	**0.12**
CommNet	0.80	0.76	0.71	–	**0.32**
GridNet	**1.00**	**0.96**	**0.88**	**0.68**	–

The cross combat winning rates averaged over 100 episodes are reported in Tables 1 and 2 for 5I and 3I2Z, respectively. As we can clearly observe, GridNet shows great advantage during the matches in both scenarios 5I and 3I2Z.

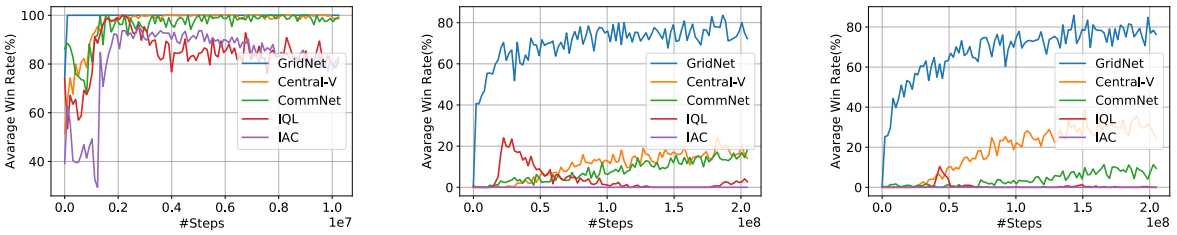
5.3.2. Transferability

As we have claimed throughout the paper, a main advantage of GridNet is that it is specially devised to handle varying number of agents. After obtaining the trained policies in 5I and 3I2Z, it is straightforward to evaluate whether the policies can be used directly in new scenarios with more agents. To this end, we create new combat maps with 10 Immortals vs. 10 Immortals (10I), 20 Immortals vs. 20 Immortals (20I), (5 Immortals, 5 Zealots) vs. (5 Immortals, 5 Zealots) (5I5Z), and (10 Immortals, 10 Zealots) vs. (10 Immortals, 10 Zealots) (10I10Z). For these new games, CommNet is not testable, and we report winning rates of IQL, IAC, Central-V and GridNet in Table 3. In the following, we will only report results against AN and HR, since competing with Rand opponent is of less interest.

From Table 3, the trained models for IQL and IAC on 5I and 3I2Z can barely beat the handcraft AN and HR policies, and the transferability of Central-V is very limited as well. In contrast, the average winning rate of GridNet almost keeps identical with those reported on 5I and 3I2Z, and specifically the policy from 5I obtains 100% winning rate against both AN and HR when transferring to 10I and 20I. The results demonstrate that the policy generated by GridNet can be



(a) 5I with Rand opponent (b) 5I with AN opponent (c) 5I with HR opponent



(d) 3I2Z with Rand opponent (e) 3I2Z with AN opponent (f) 3I2Z with HR opponent

Figure 3. Average test wining rate vs. training steps (i.e., number of passed data points) on 5I and 3I2Z when training against Rand, AN and HR. We omit the curves on SP, because using self-play, the winning rate will always be a value slightly above 50%.

Table 3. The results of the policies trained on 5I and 3I2Z while testing on new scenarios with more agents.

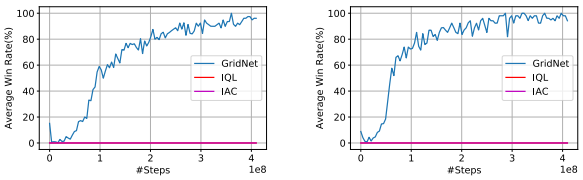
Scenario	10I		20I		5I5Z		10I10Z	
Opponent	AN	HR	AN	HR	AN	HR	AN	HR
IAC	0.02	0.00	0.01	0.00	0.00	0.00	0.00	0.00
IQL	0.03	0.00	0.01	0.00	0.02	0.00	0.01	0.00
Central-V	0.78	0.76	0.55	0.49	0.02	0.10	0.02	0.05
GridNet	**1.00**	**1.00**	**1.00**	**1.00**	**0.70**	**0.87**	**0.79**	**0.71**

successfully transferred to new scenarios with much more agents, without losing any performance.

5.4. Results on MAB

Training with fewer agents while testing with increased number of agents is a simple way to demonstrate whether GridNet can handle varying number of agents, but this should be regarded as the transferability of the method. In this experiment, we propose to study a very complex battle scenario that both the initial number of agents and agent types are random, i.e., the MAB scenario introduced in Section 5.1, and we directly train GridNet on it from scratch. The setting in MAB closely approaches the battles in a formal game of StarCraft II, so successes in MAB can be viewed as a promising possibility of creating strong AI in formal game of StarCraft II.

As mentioned in Section 5.2, Central-V and CommNet cannot be trained on MAB. Fig. 4 provides the test average



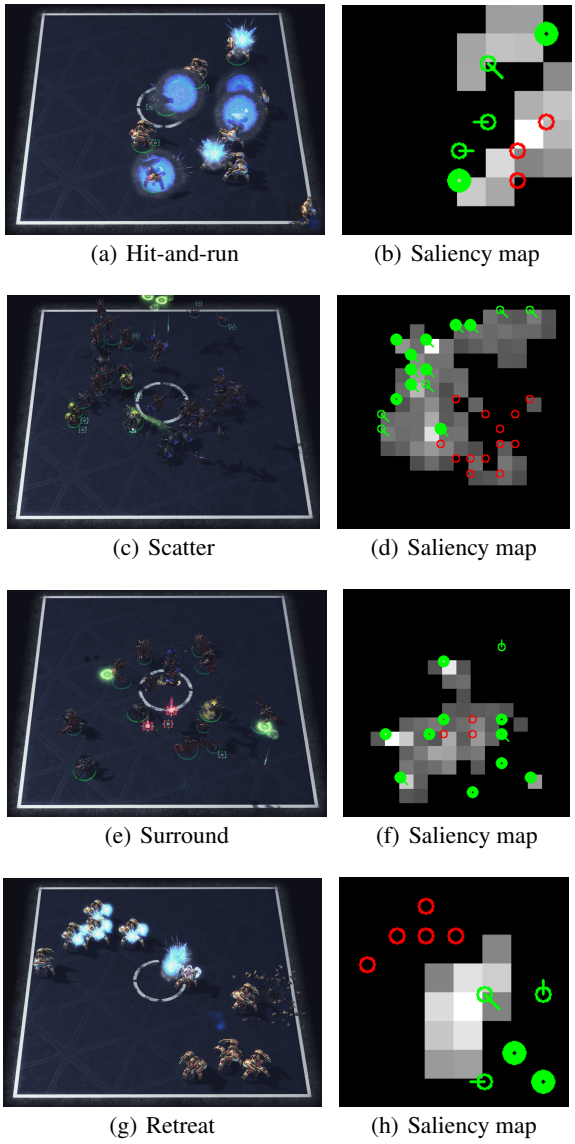
(a) MAB against AN (b) MAB against HR

Figure 4. Average test wining rate vs. training steps on MAB when training against AN and HR.

Table 4. Cross combat winning rate over the trained policies of GridNet in MAB. The table format follows that used in Table 1.

GridNet	AN	HR	SP
AN	–	0.48	**0.18**
HR	0.52	–	**0.12**
SP	**0.82**	**0.88**	–

winning rate vs. training steps curves of IQL, IAC and GridNet. From the figure, the independent learning methods fail to learn any informative policies and their scores keep zeros along the horizontal axis. In contrast, GridNet can still reach a winning rate approaching 100% against both AN and HR after training over 40M steps. Moreover, we also let the trained policies of GridNet with AN, HR and SP compete with each other. Table 4 shows the cross combat winning rates averaged over 100 episodes, and again the



(a) Hit-and-run (b) Saliency map

(c) Scatter (d) Saliency map

(e) Surround (f) Saliency map

(g) Retreat (h) Saliency map

Figure 5. Examples of the learned tactics of GridNet. Green arrows and solid circles in the saliency map indicate the move directions and the attack-nearest action, respectively. Red circles denote the enemies. Brighter grids indicate more attention from the agents.

strongest policy is the one trained from self-play.

Since all the other baseline methods either cannot be applied or cannot learn informative policies in this setting, we invite several expert human players ranging from Platinum to Diamond level in the ranking system of StarCraft II Battle.net League to play with GridNet on MAB. Although this is not fair to human players, because GridNet can manipulate multiple agents at each time step, we will not change the settings in this work and only give a performance evaluation of GridNet. After playing with the human players over 10 games, GridNet keeps a 100% winning rate.

5.5. Visualizing the Trained GridNet

In this subsection, we show some examples of what a well-trained GridNet has learned. We first summarize some strong behaviors that can be frequently observed from the trained GridNet policies as below.

- Hit-and-run: a commonly used human tactic, which has been explained in Section 5.1;
- Scatter and surround: the agents always try to scatter as quickly as possible and then surround the enemies, creating a concave formation to maximize the damage output in the self side;
- Retreat when alone: at the initial step of the game, the locations of the agents are randomized and some agents may be too close to the enemies. These agents will retreat to join the team and then fight back;
- Protect core unit: in MAB, the air unit Mutalisk, which is an important agent in the scenario, has learned professional hit-and-run tactic, and the ground units always try to draw fire and protect Mutalisks.

A demo video can be found at `https://youtu.be/LTcr01iTgZA`. In Fig. 5, we provide some corresponding examples, where we also show the saliency map in each case by adopting the ConvNet visualization technique proposed by (Simonyan et al., 2013). The highlighted grids in the saliency map indicate the areas that the agents are paying attention to when taking the current actions. For example, in Fig. 5(h), the attention is paying to the empty grids (where to retreat) around the most dangerous agent.

6. Conclusion

We considered a challenging problem in multi-agent reinforcement learning in game AI, where the number of agents could randomly change over time in a spatial grid-world environment. We proposed to control the agents grid-wisely using an encoder-decoder architecture to predict per grid actions. Once a grid is occupied with an agent, the agent will be controlled independently with the corresponding action. The proposed method can naturally handle varying number of agents as long as the agents are allocated in the grid map. Moreover, by using convolutional neural networks, the large receptive field can naturally promote agent collaboration and the shared convolutional parameters also enable parallel exploration among the agents. We demonstrated the effectiveness of the proposed method in many challenging battle games in StarCraft II and provided comprehensive studies on the battle games. For future directions, we are interested in applying the GridNet method to multi-agent problems with continuous action space and different grid-world environments.

Acknowledgements

We thank the anonymous reviewers for their valuable comments. We would like to thank Yuan Gao, Zequn Jie, Han Hu for their enlightening discussions on image segmentation.

References

Arel, I., Liu, C., Urbanik, T., and Kohls, A. Reinforcement learning-based multi-agent system for network traffic signal control. *IET Intelligent Transport Systems*, 4 (2):128–135, 2010.

Busoniu, L., Babuska, R., and De Schutter, B. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, And Cybernetics-Part C: Applications and Reviews, 38 (2)*, 2008.

Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017.

Chen, L.-C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 40(4):834–848, 2018.

Deepmind. Alphastar. https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/, 2019.

Devin, C., Gupta, A., Darrell, T., Abbeel, P., and Levine, S. Learning modular neural network policies for multi-task and multi-robot transfer. In *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 2169–2176. IEEE, 2017.

Espeholt, L., Soyer, H., Munos, R., Simonyan, K., Mnih, V., Ward, T., Doron, Y., Firoiu, V., Harley, T., and Dunning, I. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. *arXiv preprint arXiv:1802.01561*, 2018.

Foerster, J., Assael, I. A., de Freitas, N., and Whiteson, S. Learning to communicate with deep multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2137–2145, 2016.

Foerster, J., Farquhar, G., Afouras, T., Nardelli, N., and Whiteson, S. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926*, 2017a.

Foerster, J., Nardelli, N., Farquhar, G., Afouras, T., Torr, P. H., Kohli, P., and Whiteson, S. Stabilising experience replay for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1702.08887*, 2017b.

Foerster, J., Chen, R. Y., Al-Shedivat, M., Whiteson, S., Abbeel, P., and Mordatch, I. Learning with opponent-learning awareness. In *International Conference on Autonomous Agents and MultiAgent Systems*, pp. 122–130, 2018.

Gupta, J. K., Egorov, M., and Kochenderfer, M. Cooperative multi-agent control using deep reinforcement learning. In *International Conference on Autonomous Agents and Multiagent Systems*, pp. 66–83, 2017.

Hinton, G. E. and Salakhutdinov, R. R. Reducing the dimensionality of data with neural networks. *Science*, 313 (5786):504–507, 2006.

Jin, J., Song, C., Li, H., Gai, K., Wang, J., and Zhang, W. Real-time bidding with multi-agent reinforcement learning in display advertising. *arXiv preprint arXiv:1802.09756*, 2018.

Kempka, M., Wydmuch, M., Runc, G., Toczek, J., and Jaśkowski, W. ViZDoom: A Doom-based AI research platform for visual reinforcement learning. In *IEEE Conference on Computational Intelligence and Games*, pp. 341–348, Sep 2016.

Kober, J., Bagnell, J. A., and Peters, J. Reinforcement learning in robotics: A survey. *The International Journal of Robotics Research*, 32(11):1238–1274, 2013.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 1097–1105, 2012.

Lillicrap, T. P., Hunt, J. J., Pritzel, A., Heess, N., Erez, T., Tassa, Y., Silver, D., and Wierstra, D. Continuous control with deep reinforcement learning. *arXiv preprint arXiv:1509.02971*, 2015.

Long, J., Shelhamer, E., and Darrell, T. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 3431–3440, 2015.

Lowe, R., Wu, Y., Tamar, A., Harb, J., Abbeel, O. P., and Mordatch, I. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 6379–6390, 2017.

Luo, W., Sun, P., Zhong, F., Liu, W., Zhang, T., and Wang, Y. End-to-end active object tracking via reinforcement learning. In *International Conference on Machine Learning (ICML)*, pp. 3286–3295, 2018.

Mao, H., Alizadeh, M., Menache, I., and Kandula, S. Resource management with deep reinforcement learning. In

Proceedings of the 15th ACM Workshop on Hot Topics in Networks, pp. 50–56, 2016.

Matignon, L., Jeanpierre, L., and Mouaddib, A.-I. Coordinated multi-robot exploration under communication constraints using decentralized markov decision processes. In *The AAAI Conference on Artificial Intelligence (AAAI)*, 2012.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., and Ostrovski, G. Human-level control through deep reinforcement learning. *Nature*, 518(7540): 529, 2015.

Noh, H., Hong, S., and Han, B. Learning deconvolution network for semantic segmentation. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 1520–1528, 2015.

OpenAI. Openai five. https://blog.openai.com/openai-five/, 2018.

Peng, P., Yuan, Q., Wen, Y., Yang, Y., Tang, Z., Long, H., and Wang, J. Multiagent bidirectionally-coordinated nets for learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

Rashid, T., Samvelyan, M., de Witt, C. S., Farquhar, G., Foerster, J., and Whiteson, S. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 4510–4520, 2018.

Schulman, J., Levine, S., Abbeel, P., Jordan, M., and Moritz, P. Trust region policy optimization. In *International Conference on Machine Learning (ICML)*, pp. 1889–1897, 2015.

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

Silver, D., Lever, G., Heess, N., Degris, T., Wierstra, D., and Riedmiller, M. Deterministic policy gradient algorithms. In *International Conference on Machine Learning (ICML)*, 2014.

Silver, D., Huang, A., Maddison, C. J., Guez, A., Sifre, L., Van Den Driessche, G., Schrittwieser, J., Antonoglou, I., Panneershelvam, V., and Lanctot, M. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484, 2016.

Silver, D., Schrittwieser, J., Simonyan, K., Antonoglou, I., Huang, A., Guez, A., Hubert, T., Baker, L., Lai, M., and Bolton, A. Mastering the game of go without human knowledge. *Nature*, 550(7676):354, 2017.

Simonyan, K., Vedaldi, A., and Zisserman, A. Deep inside convolutional networks: Visualising image classification models and saliency maps. *arXiv preprint arXiv:1312.6034*, 2013.

Sukhbaatar, S. and Fergus, R. Learning multiagent communication with backpropagation. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 2244–2252, 2016.

Sunehag, P., Lever, G., Gruslys, A., Czarnecki, W. M., Zambaldi, V., Jaderberg, M., Lanctot, M., Sonnerat, N., Leibo, J. Z., and Tuyls, K. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Sutton, R. S., Barto, A. G., and Bach, F. *Reinforcement learning: An introduction*. MIT press, 1998.

Tampuu, A., Matiisen, T., Kodelja, D., Kuzovkin, I., Korjus, K., Aru, J., Aru, J., and Vicente, R. Multiagent cooperation and competition with deep reinforcement learning. *PloS one*, 12(4), 2017.

Tan, M. Multi-agent reinforcement learning: independent versus cooperative agents. In *International Conference on Machine Learning (ICML)*, pp. 330–337, 1993.

Tesauro, G. Extending q-learning to general adaptive multi-agent systems. In *Advances in Neural Information Processing Systems (NIPS)*, pp. 871–878, 2004.

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

Watkins, C. J. C. H. *Learning from delayed rewards*. PhD thesis, King's College, Cambridge, 1989.

Wiering, M. Multi-agent reinforcement learning for traffic light control. In *International Conference on Machine Learning (ICML)*, pp. 1151–1158, 2000.

Williams, R. J. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, 8(3-4):229–256, 1992.