Multi-Agent Reinforcement Learning is A Sequence Modeling Problem

Muning Wen^{1,2}, Jakub Grudzien Kuba^{3}, Runji Lin^{4}, Weinan Zhang^{1}, Ying Wen^{1}, Jun Wang^{2,5}, Yaodong Yang^{6}
^{1}Shanghai Jiao Tong University, ^{2}Digital Brain Lab,
^{3}University of Oxford, ^{4}Institute of Automation, Chinese Academy of Science,
^{5}University College London, ^{6}Institute for AI, Peking University

Abstract

Large sequence model (SM) such as GPT series and BERT has displayed outstanding performance and generalization capabilities on vision, language, and recently reinforcement learning tasks. A natural follow-up question is how to abstract multi-agent decision making into an SM problem and benefit from the prosperous development of SMs. In this paper, we introduce a novel architecture named Multi-Agent Transformer (MAT) that effectively casts cooperative multi-agent reinforcement learning (MARL) into SM problems wherein the task is to map agents' observation sequence to agents' optimal action sequence. Our goal is to build the bridge between MARL and SMs so that the modeling power of modern sequence models can be unleashed for MARL. Central to our MAT is an encoder-decoder architecture which leverages the multi-agent advantage decomposition theorem to transform the joint policy search problem into a sequential decision making process; this renders only linear time complexity for multi-agent problems and, most importantly, endows MAT with monotonic performance improvement guarantee. Unlike prior arts such as Decision Transformer fit only pre-collected offline data, MAT is trained by online trials and errors from the environment in an on-policy fashion. To validate MAT, we conduct extensive experiments on StarCraftII, Multi-Agent MuJoCo, Dexterous Hands Manipulation, and Google Research Football benchmarks. Results demonstrate that MAT achieves superior performance and data efficiency compared to strong baselines including MAPPO and HAPPO. Furthermore, we demonstrate that MAT is an excellent few-short learner on unseen tasks regardless of changes in the number of agents. See our project page at `https://sites.google.com/view/multi-agent-transformer`[(1)].

1 Introduction

Multi-agent reinforcement learning (MARL) [32] is a challenging problem for its difficulty which arises not only from identifying each individual agent's policy improvement direction, but also from combining agents' policy updates jointly which should be beneficial for the whole team. Recently, such difficulty in multi-agent learning has been eased owing to the introduction of *centralized training for decentralized execution* (CTDE) [10], which allows agents to access the global information and opponents' actions during the training phase. This framework enables successful developments of methods that directly extend single-agent algorithms by, for example, replacing the policy-gradient (PG) estimate with a multi-agent PG (MAPG) counterpart [10], or endowing all agents with the same set of parameters and then training by a trust-region method [33]. These methods, however, cannot cover the whole complexity of multi-agent interactions; in fact, some of them are shown to fail in

[(1)]Corresponding to <yaodong.yang@pku.edu.cn>

Preprint. Under review.

the simplest cooperative task [14]. To resolve this issue, the *multi-agent advantage decomposition theorem* was proposed [14, Theorem 1], which captures how different agents contribute to the return and provides an intuition behind the emergence of cooperation through a sequential decision making process scheme. Based on it, HATRPO and HAPPO algorithms [14] were derived which, thanks to the decomposition theorem and sequential update scheme, established new state-of-the-art methods for MARL. However, their limitation is that the agents' policies are unaware of the purpose to develop cooperation and still rely on a carefully handcrafted maximization objective. Ideally, a team of agents should be aware of the jointness of their training by design, thereby following a holistic and effective paradigm—an ideal solution that is yet to be proposed.

In recent years, *sequence models* (SM) have made a substantial progress in *natural language processing* (NLP) [18]. For example, GPT series [3] and BERT models [8], built on autoregressive SMs, have demonstrated remarkable performance on a wide range of downstream tasks and achieved strong performance on few-shot generalization tasks. Although SMs are mostly used in language tasks due to its natural fitting with the sequential property of languages, the sequential approaches are not confined to NLP only, but is instead a widely applicable general foundation model [2]. For example, in *computer vision* (CV), one can split an image into sub-images and align them in a sequence as if they were tokens in NLP tasks [8, 9, 11]. Although the idea of solving CV tasks by SMs is straightforward, it serves as the foundation to some of the best-performing CV algorithms [28, 31, 29]. Furthermore, sequential methods are starting to spawn powerful multi-modal visual language models such as Flamingo [1], DALL-E [20], and GATO [22] in the recent past.

Coming with effective and expressive network architectures such as *Transformer* [30], sequence modeling techniques have also attracted tremendous attention from the RL community, which results in a series of successful offline RL developments based on the Transformer architecture [5, 12, 22]. These methods show great potentials in tackling some of the most fundamental RL training problems, such as long-horizon credit assignment and reward sparsity [27]. For example, by training autoregressive models on pre-collected offline data in a purely supervised way, Decision Transformer [5] bypasses the need for computing cumulative rewards through dynamic programming, but rather generates future actions conditioning on the desired returns, past states and actions. Despite their remarkable successes, none of these methods have been designed to model the most difficult (also unique to MARL) aspect of multi-agent systems—the agents' interactions. In fact, if we were to simply endow all agents with a Transformer policy and train them independently, their joint performance still could not be guaranteed to improve [14, Proposition 1]. Therefore, while a myriad of powerful SMs are available, MARL—an area that would greatly benefit from SMs—has not truly taken advantage of their performance benefit. The key research question to ask is then

How can we solve MARL problems by sequence models?

In this paper, we take several steps to provide an affirmative answer to the above research question. Our goal is to enhance MARL studies with powerful sequential modeling techniques. To fulfill that, we start by proposing a novel MARL training paradigm which establishes the connection between cooperative MARL problems and sequence modeling problems. Central to the new paradigm are the *multi-agent advantage decomposition theorem* and *sequential update scheme*, which effectively transform multi-agent joint policy optimization into a sequential policy search process. As a natural outcome of our findings, we introduce *Multi-Agent Transformer* (MAT), an encoder-decoder architecture that implements generic MARL solutions through SMs. Unlike Decision Transformer [5], MAT is trained online based on trials and errors in an on-policy fashion; therefore, it does not require collecting demonstrations upfront. Importantly, the implementation of the multi-agent advantage decomposition theorem ensures MAT to enjoy monotonic performance improvement guarantee during training. MAT establishes a new state-of-the-art baseline model for cooperative MARL tasks. We justify such a claim by evaluating MAT on the benchmarks of StarCraftII, Multi-Agent MuJoCo, Dexterous Hands Manipulation and Google Research Football; results show that MAT achieves superior performance over strong baselines, such as MAPPO [33], HAPPO [14] and QMIX [21]. Finally, we show that MAT possesses great potentials in task generalizations, which holds regardless of the agent number in new tasks.

2 Preliminaries

In this section, we first introduce the cooperative MARL problem formulation and the multi-agent advantage decomposition theorem, which serves as the cornerstone of our work. We then review existing MARL methods that relate to MAT, and finally familiarize the reader with the Transformer.

2.1 Problem Formulation

Cooperative MARL problems are often modeled by *decentralized Partially Observable Markov Decision Processes* (Dec-POMDPs) $\langle \mathcal{N}, \boldsymbol{\mathcal{O}}, \boldsymbol{\mathcal{A}}, R, P, \gamma \rangle$ [19]. $\mathcal{N} = \{1, \dots, n\}$ is the set of agents, $\boldsymbol{\mathcal{O}} = \prod_{i=1}^{n} \mathcal{O}^i$ is the product of local observation spaces of the agents, namely the joint observation space, $\boldsymbol{\mathcal{A}} = \prod_{i=1}^{n} \mathcal{A}^i$ is the product of the agents' action spaces, namely the joint action space, $R : \boldsymbol{\mathcal{O}} \times \boldsymbol{\mathcal{A}} \to [-R_{\max}, R_{\max}]$ is the joint reward function, $P : \boldsymbol{\mathcal{O}} \times \boldsymbol{\mathcal{A}} \times \boldsymbol{\mathcal{O}} \to \mathbb{R}$ is the transition probability function, and $\gamma \in [0, 1)$ is the discount factor. At time step $t \in \mathbb{N}$, an agent $i \in \mathcal{N}$ observes an observation $\mathrm{o}_t^i \in \mathcal{O}^i$ [(2)] ($\boldsymbol{o} = (o^1, \dots, o^n)$ is a "joint" observation) and takes an action a_t^i according to its policy π^i, which is the i^{th} component of the agents' joint policy $\boldsymbol{\pi}$. At each time step, all agents take actions **simultaneously** based on their observation with no sequential dependencies. The transition kernel P and the joint policy induce the (improper) marginal observation distribution $\rho_{\boldsymbol{\pi}}(\cdot) \triangleq \sum_{t=0}^{\infty} \gamma^t \Pr(\mathbf{o}_t = \boldsymbol{o} | \boldsymbol{\pi})$. At the end of each time step, the whole team receives a joint reward $R(\mathbf{o}_t, \mathbf{a}_t)$ and observe $\mathbf{o}_{t+1}$, whose probability distribution is $P(\cdot | \mathbf{o}_t, \mathbf{a}_t)$. Following this process infinitely long, the agents earn a discounted cumulative return of $R^\gamma \triangleq \sum_{t=0}^{\infty} \gamma^t R(\mathbf{o}_t, \mathbf{a}_t)$.

2.2 Multi-Agent Advantage Decomposition Theorem

The agents evaluate the value of actions and observations with $Q_{\boldsymbol{\pi}}(\boldsymbol{o}, \boldsymbol{a})$ and $V_{\boldsymbol{\pi}}(\boldsymbol{o})$, defined as

$$Q_{\boldsymbol{\pi}}(\boldsymbol{o}, \boldsymbol{a}) \triangleq \mathbb{E}_{\mathbf{o}_{1:\infty} \sim P, \mathbf{a}_{1:\infty} \sim \boldsymbol{\pi}} \left[R^\gamma | \mathbf{o}_0 = \boldsymbol{o}, \mathbf{a}_0 = \boldsymbol{a} \right], \tag{1}$$
$$V_{\boldsymbol{\pi}}(\boldsymbol{o}) \triangleq \mathbb{E}_{\mathbf{a}_0 \sim \boldsymbol{\pi}, \mathbf{o}_{1:\infty} \sim P, \mathbf{a}_{1:\infty} \sim \boldsymbol{\pi}} \left[R^\gamma | \mathbf{o}_0 = \boldsymbol{o} \right].$$

The jointness of the objective causes difficulties associated with the *credit assignment* problem—having received a shared reward, individual agents are unable to deduce their own contribution to the team's success or failure [4]. Indeed, applying traditional RL methods which simply employ the above value functions leads to obstacles in training, such as the growing variance of multi-agent policy gradient (MAPG) estimates [13]. Hence, to tackle these, notions of local value functions [16] and counterfactual baselines [10] have been developed. In this paper, we work with the most general notions of this kind—*the multi-agent observation-value functions* [14]. That is, for arbitrary *disjoint*, *ordered* subsets of agents $i_{1:m} = \{i_1, \dots, i_m\}$ and $j_{1:h} = \{j_1, \dots, j_h\}$, for $m, h \le n$, we define the multi-agent observation-value function by

$$Q_{\boldsymbol{\pi}}(\boldsymbol{o}, \boldsymbol{a}^{i_{1:m}}) \triangleq \mathbb{E}\left[R^\gamma | \mathbf{o}_0^{i_{1:n}} = \boldsymbol{o}, \mathbf{a}_0^{i_{1:m}} = \boldsymbol{a}^{i_{1:m}} \right], \tag{2}$$

which recovers the original state-action value function in Equation (1) when $m = n$, and the original observation-value function when $m = 0$ (*i.e.*, when the set $i_{1:m}$ is empty). Based on Equation (2), we can further measure the contribution of a chosen subset of agents to the joint return and define the multi-agent advantage function by

$$A_{\boldsymbol{\pi}}^{i_{1:m}}(\boldsymbol{o}, \boldsymbol{a}^{j_{1:h}}, \boldsymbol{a}^{i_{1:m}}) \triangleq Q_{\boldsymbol{\pi}}^{j_{1:h}, i_{1:m}}(\boldsymbol{o}, \boldsymbol{a}^{j_{1:h}}, \boldsymbol{a}^{i_{1:m}}) - Q_{\boldsymbol{\pi}}^{j_{1:h}}(\boldsymbol{o}, \boldsymbol{a}^{j_{1:h}}). \tag{3}$$

The above quantity describes how much better/worse than average the joint action $\boldsymbol{a}$ will be if agents $i_{1:m}$ take the joint action $\boldsymbol{a}^{i_{1:m}}$, once $j_{1:h}$ have taken $\boldsymbol{a}^{j_{1:h}}$. Again, when $h = 0$, the advantage compares the value of $\boldsymbol{a}^{i_{1:m}}$ to the baseline value function of the whole team. This value-functional representation of agents' actions enables studying interactions between them, as well to decompose the joint value function signal, thus helping alleviate the severity of the credit assignment problem [21, 26, 17]. The insights of Equation (3) is accomplished by means of the following theorem.

Theorem 1 (Multi-Agent Advantage Decomposition [13]). *Let $i_{1:n}$ be a permutation of agents. Then, for any joint observation $\boldsymbol{o} \in \boldsymbol{\mathcal{O}}$ and joint action $\boldsymbol{a} = \boldsymbol{a}^{i_{1:n}} \in \boldsymbol{\mathcal{A}}$, the following equation always holds with no further assumption needed,*

$$A_{\boldsymbol{\pi}}^{i_{1:n}}(\boldsymbol{o}, \boldsymbol{a}^{i_{1:n}}) = \sum_{m=1}^{n} A_{\boldsymbol{\pi}}^{i_m}(\boldsymbol{o}, \boldsymbol{a}^{i_{1:m-1}}, a^{i_m}).$$

Importantly, this theorem provides an intuition guiding the choice of incrementally improving actions. Suppose that agent i_1 picks an action a^{i_1} with positive advantage, $A_{\boldsymbol{\pi}}^{i_1}(\boldsymbol{o}, a^{i_1}) > 0$. Then, imagine

[(2)]For notation convenience, we omit defining agents' observation functions that take the global state as the input and outputs a local observation for each agent, but rather define agents' local observations directly.

that for all $j = 2, \ldots, n$, agent i_j knows the joint action $\boldsymbol{a}^{i_{1:j-1}}$ of its predecessors. In this case, it can choose an action a^{i_j} for which the advantage $A^{i_j}_{\boldsymbol{\pi}}(\boldsymbol{o}, \boldsymbol{a}^{i_{1:j-1}}, a^{i_j})$ is positive. Altogether, the theorem assures that the joint action $\boldsymbol{a}^{i_{1:n}}$ has positive advantage. Furthermore, notice that the joint action has been chosen in n steps, each of which searched an individual agent's action space. Hence, the complexity of this search is additive, $\sum_{i=1}^{n} |\mathcal{A}^i|$, in the sizes of the action spaces. If we were to perform the search directly in the joint action space, we would browse a set of multiplicative size, $|\boldsymbol{\mathcal{A}}| = \prod_{i=1}^{n} |\mathcal{A}^i|$. Later, we will build upon this insight to design an SM that optimises joint policies efficiently, agent by agent, without the necessity of considering the joint action space at once.

2.3 Existing Methods in MARL

We now briefly summarize two state-of-the-art MARL algorithms. Both of them build upon *Proximal Policy Optimization* (PPO) [25]—an RL method famous for its simplicity and its performance stability.

Multi-Agent Proximal Policy Optimisation (MAPPO) [33] is the first, and the most direct, approach for applying PPO in MARL. It equips all agents with one shared set of parameters and use agents' aggregated trajectories for the shared policy's update; at iteration $k+1$, it optimizes the policy parameter θ_{k+1} by maximizing the clip objective of

$$\sum_{i=1}^{n} \mathbb{E}_{\mathbf{o}\sim\rho_{\boldsymbol{\pi}_{\theta_k}}, \mathbf{a}\sim\boldsymbol{\pi}_{\theta_k}} \left[\min\left(\frac{\pi_\theta(\mathrm{a}^i|\mathbf{o})}{\pi_{\theta_k}(\mathrm{a}^i|\mathbf{o})} A_{\boldsymbol{\pi}_{\theta_k}}(\mathbf{o}, \mathbf{a}), \mathrm{clip}\left(\frac{\pi_\theta(\mathrm{a}^i|\mathbf{o})}{\pi_{\theta_k}(\mathrm{a}^i|\mathbf{o})}, 1 \pm \epsilon \right) A_{\boldsymbol{\pi}_{\theta_k}}(\mathbf{o}, \mathbf{a}) \right) \right],$$

where the clip operator clips the input value (if necessary) so that it stays within the interval $[1-\epsilon, 1+\epsilon]$. However, enforcing parameter sharing is equivalent to putting a constraint $\theta^i = \theta^j, \forall i, j \in \mathcal{N}$ on the joint policy space, which can lead to an exponentially-worse sub-optimal outcome [14]. This motivates a more principled development of heterogeneous-agent trust-region methods, e.g., HAPPO.

Heterogeneous-Agent Proximal Policy Optimisation (HAPPO) [14] is currently one of the SOTA algorithm that fully leverages Theorem (1) to implement multi-agent trust-region learning with monotonic improvement guarantee. During an update, the agents choose a permutation $i_{1:n}$ at random, and then following the order in the permutation, every agent i_m picks $\pi^{i_m}_{\text{new}} = \pi^{i_m}$ that maximizes the objective of

$$\mathbb{E}_{\mathbf{o}\sim\rho_{\boldsymbol{\pi}_{\text{old}}}, \mathbf{a}^{i_{1:m-1}}\sim\boldsymbol{\pi}^{i_{1:m-1}}_{\text{new}}, \mathrm{a}^{i_m}\sim\pi^{i_m}_{\text{old}}} \left[\min\left(\mathrm{r}(\pi^{i_m}) A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(\boldsymbol{o}, \mathbf{a}^{i_{1:m}}), \mathrm{clip}(\mathrm{r}(\pi^{i_m}), 1 \pm \epsilon) A^{i_{1:m}}_{\boldsymbol{\pi}_{\text{old}}}(\mathbf{o}, \mathbf{a}^{i_{1:m}}) \right) \right],$$

where $\mathrm{r}(\pi^{i_m}) = \pi^{i_m}(\mathrm{a}^{i_m}|\mathbf{o}) / \pi^{i_m}_{\text{old}}(\mathrm{a}^{i_m}|\mathbf{o})$. Note that the expectation is taken over the newly-updated previous agents' policies, *i.e,* $\boldsymbol{\pi}^{i_{1:m-1}}_{\text{new}}$; this reflects an intuition that, under Theorem (1), the agent i_m reacts to its preceding agents $i_{1:m-1}$. However, one drawback of HAPPO is that agent's policies have to follow the sequential update scheme in the permutation, thus it cannot be run in parallel.

2.4 The Transformer Model

Transformer [30] was originally designed for machine translation tasks (e.g., input English, output French). It maintains an encoder-decoder structure, where the encoder maps an input sequence of tokens to latent representations and then the decoder generates a sequence of desired outputs in an auto-regressive manner wherein at each step of inference, the Transformer takes all previously generated tokens as the input. One of the most essential component in Transformer is the scaled dot-product attention, which captures the interrelationship of input sequences. The attention function is written as $\mathrm{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \mathrm{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}}\right)\mathbf{V}$, where the $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ corresponds to the vector of queries, keys and values, which can be learned during training, and the d_k represent the dimension of $\mathbf{Q}$ and $\mathbf{K}$. Self-attentions refer to cases when $\mathbf{Q}, \mathbf{K}, \mathbf{V}$ share the same set of parameters.

Based on the sequential property described in the Theorem (1) and the principle behind HAPPO [14], it is intuitive to think about a Transformer-based implementation for multi-agent trust-region learning. **By treating a team of agents as a sequence**, the Transformer architecture allows us to model teams of agents with variable numbers and types, while avoiding drawbacks of MAPPO/HAPPO. We will describe in more details how a cooperative MARL problem can be solved by a sequence model.

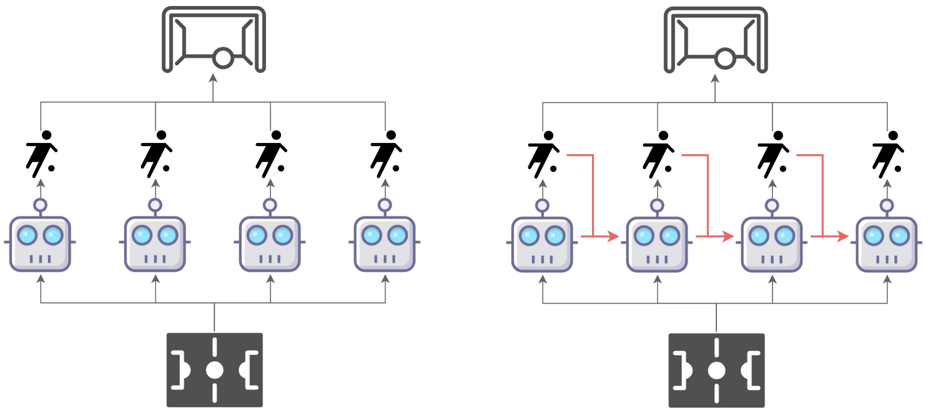


Figure 1: Conventional multi-agent learning paradigm (left) wherein all agents take actions simultaneously *vs.* the multi-agent sequential decision paradigm (right) where agents take actions by following a sequential order, each agent accounts for decisions from preceding agents as red arrows suggest.

3 The Surprising Connection Between MARL and Sequence Models

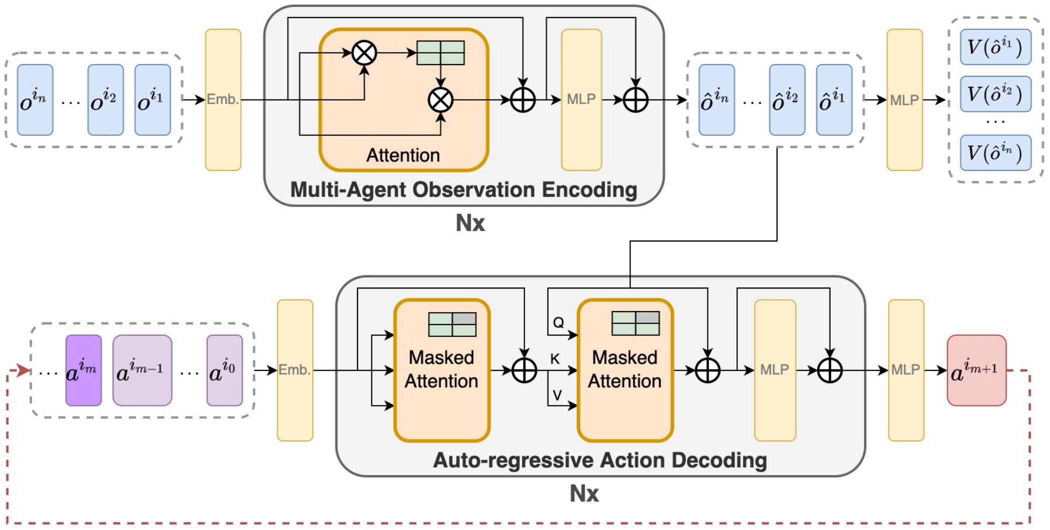
To establish the connection between MARL and sequence models, Theorem (1) provides a new angle of understanding the MARL problem from an SM perspective. If each agent knows its predecessors' actions with an arbitrary decision order, the sum of agents' local advantages $A^{i_j}_{\boldsymbol{\pi}}(\boldsymbol{o}, \boldsymbol{a}^{i_{1:m-1}}, a^{i_m})$ will be exactly equal to the joint advantages $A^{i_{1:n}}_{\boldsymbol{\pi}}(\boldsymbol{o}, \boldsymbol{a}^{i_{1:n}})$. This orderly decision setting across agents simplifies the update of their joint policy, where maximizing each agent's own local advantage is equivalent to maximizing the joint advantage. As such, agents do not need to worry about interference from other agents anymore during the policy update; the local advantage functions have already captured the relationship between agents. This property revealed by Theorem (1) inspires us to propose a multi-agent sequential decision paradigm for MARL problems as show in Figure (1), where we assign agents with an arbitrary decision order (one permutation for each iteration); each agent can access its predecessors' behaviors, based on which it then takes the optimal decision. This sequential paradigm motivates us to leverage a sequential model, e.g., Transformer, to explicitly capture the sequential relationship between agents described in Theorem (1).

Underpinned by Theorem (1), sequence modeling reduces the complexity growth of MARL problems with the number of agents from multiplicative to additive, thus rendering linear complexity. With the help of the Transformer architecture, we can model policies of heterogeneous agents with an unified network but treat each agent discriminatively with different position, and thus ensuring high sample efficiency while avoiding the exponentially-worse outcome that MAPPO is facing. Besides, the Transformer architecture allows the sequential policies to be trained in parallel, similar to machine translation tasks, which significantly improves the training speed and makes it feasible for a large size of agents. Furthermore, in cases that the number and the type of agents are different, an SM can incorporates them into a unified solution through its capability on modeling sequences with flexible sequence length, rather than treat different agent numbers as different tasks. To realize the above idea, we introduce a practical architecture named Multi-Agent Transformer in the next section.

4 The Multi-Agent Transformer

To implement the sequence modeling paradigm for MARL, our solution is *Multi-Agent Transformer* (MAT). The idea of applying the Transformer architecture comes from the fact that the mapping between the input of agents' observation sequence $(o^{i_1}, \ldots, o^{i_n})$ and the output of agents' action sequence $(a^{i_1}, \ldots, a^{i_n})$ are sequence modeling tasks similar to machine translations. As eluded by Theorem (1), the action a^{i_m} depends on all previous agents' decisions $\boldsymbol{a}^{i_{1:m-1}}$. Hence, our MAT in Figure (2) consists of an *encoder*, which learns representations of the joint observations, and a *decoder* which outputs actions for each individual agent in an auto-regressive manner.

The encoder, whose parameters we denote by ϕ, takes a sequence of observations $(o^{i_1}, \ldots, o^{i_n})$ in arbitrary order and passes them through several computational blocks. Each such block consists of a *self-attention* mechanism and a *multi-layer perceptron* (MLP), as well as *residual connections* to prevent gradient vanishing and network degradation with the increase of depth. We denote the output



o^{i_n} ⋯ o^{i_2} o^{i_1}
Emb.
Attention
Multi-Agent Observation Encoding
Nx
MLP
$\hat{o}^{i_n}$ ⋯ $\hat{o}^{i_2}$ $\hat{o}^{i_1}$
MLP
$V(\hat{o}^{i_1})$
$V(\hat{o}^{i_2})$
⋯
$V(\hat{o}^{i_n})$
⋯ a^{i_m} $a^{i_{m-1}}$ ⋯ a^{i_0}
Emb.
Masked Attention
Q
K
V
Masked Attention
MLP
Auto-regressive Action Decoding
Nx
MLP
$a^{i_{m+1}}$

Figure 2: The encoder-decoder architecture of MAT. At each time step, the encoder takes in a sequence of agents' observations and encodes them into a sequence of latent representations, which is then passed into the decoder. The decoder generate each agent's optimal action in a sequential and auto-regressive manner. The masked attention blocks ensures agents can only access its preceding agents' actions during training. We list the full pseudocode of MAT in Appendix A and a video that shows the dynamic data flow of MAT in `https://sites.google.com/view/multi-agent-transformer`.

encoding of the observations as $(\hat{\boldsymbol{o}}^{i_1}, \ldots, \hat{\boldsymbol{o}}^{i_n})$, which encodes not only the information of agents $(i_1, \ldots, i_n)$ but also the high-level interrelationships that represent agents' interactions. In order to learn expressive representations, in the training phase, we make the encoder to approximate the value functions, whose objective is to minimize the empirical Bellman error by

$$L_{\text{Encoder}}(\phi) = \frac{1}{Tn} \sum_{m=1}^{n} \sum_{t=0}^{T-1} \Big[R(\mathbf{o}_t, \mathbf{a}_t) + \gamma V_{\bar{\phi}}(\hat{\mathbf{o}}_{t+1}^{i_m}) - V_{\phi}(\hat{\mathbf{o}}_t^{i_m}) \Big]^2, \tag{4}$$

where $\bar{\phi}$ is the target network's parameter, which is non-differentiable and updated every few epochs.

The decoder, whose parameters we denote by θ, passes the embedded joint action $\boldsymbol{a}^{i_{0:m-1}}, m = \{1, \ldots n\}$ (where a^{i_0} is an arbitrary symbol indicating the start of decoding) to a sequence of decoding blocks. Crucially, every decoding block comes with a *masked self-attention* mechanism, where the masking makes sure that, for every i_j, attention is computed only between the i_r^{th} and the i_j^{th} action heads wherein $r < j$ so that the sequential update scheme can be maintained. This is then followed by a second masked attention function, which computes the attention between the action heads and observation representations. Finally, the block finishes with an MLP and skipping connections. The output to the last decoder block is a sequence of representations of the joint actions, $\{\hat{\boldsymbol{a}}^{i_0:i-1}\}_{i=1}^{m}$. This is fed to an MLP that outputs the probability distribution of i_m's action, namely, the policy $\pi_\theta^{i_m}(\mathrm{a}^{i_m} | \hat{\mathbf{o}}^{i_{1:n}}, \mathbf{a}^{i_{1:m-1}})$. To train the decoder, we minimize the following clipping PPO objective of

$$L_{\text{Decoder}}(\theta) = -\frac{1}{Tn} \sum_{m=1}^{n} \sum_{t=0}^{T-1} \min \Big(\mathrm{r}_t^{i_m}(\theta) \hat{A}_t, \text{clip}(\mathrm{r}_t^{i_m}(\theta), 1 \pm \epsilon) \hat{A}_t \Big),$$
$$\text{where } \mathrm{r}_t^{i_m}(\theta) = \frac{\pi_\theta^{i_m}(\mathrm{a}_t^{i_m} | \hat{\mathbf{o}}_t^{i_{1:n}}, \hat{\mathbf{a}}_t^{i_{1:m-1}})}{\pi_{\theta_{\text{old}}}^{i_m}(\mathrm{a}_t^{i_m} | \hat{\mathbf{o}}_t^{i_{1:n}}, \hat{\mathbf{a}}_t^{i_{1:m-1}})}, \tag{5}$$

and $\hat{A}_t$ is an estimate of the joint advantage function. One can apply *generalized advantage estimation* (GAE) [24] with $\hat{V}_t = \frac{1}{n} \sum_{m=1}^{n} V(\hat{\mathrm{o}}_t^{i_m})$ as a robust estimator for the joint value function. Notably, the action generation process is different between the inference and the training stage. In the inference stage, each action is generated auto-regressively, in the sense that a^{i_m} will be inserted back into the decoder again to generate $\mathrm{a}^{i_{m+1}}$ (starting with a^{i_0} and ending with $\mathrm{a}^{i_{n-1}}$). While during the training stage, the output of all actions, $\mathbf{a}^{i_{1:n}}$ can be computed in parallel simply because $\mathbf{a}^{i_{1:n-1}}$ have already been collected and stored in the replay buffer.

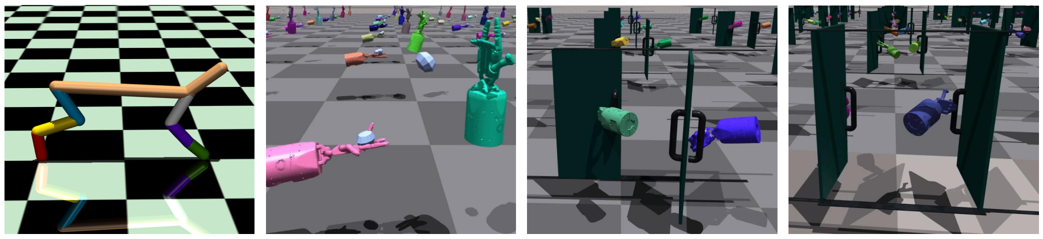


Figure 3: Demonstrations of the Bi-DexHands and the HalfCheetah environments.

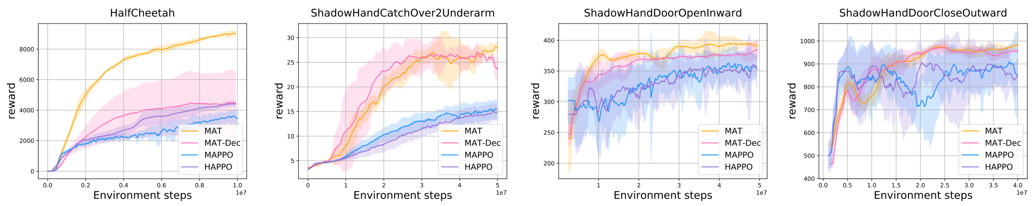


Figure 4: Performance comparisons on the Multi-Agent MuJoCo and the Bi-DexHands benchmarks.

The attention mechanism, which lies in the heart of MAT, encodes observations and actions with a weight matrix calculated by multiplying the embedded queries, $(q^{i_1}, \dots, q^{i_n})$, and keys, $(k^{i_1}, \dots, k^{i_n})$, where each of the weight $w(q^{i_r}, k^{i_j}) = \langle q^{i_r}, k^{i_j} \rangle$. The embedded values $(v^{i_1}, \dots, v^{i_n})$ are multiplied with the weight matrix to output representations. While the unmasked attention in the encoder uses a full weight matrix to extract the interrelationship between agents, i.e., $\hat{\mathbf{o}}^{i_{1:n}}$, the masked attentions in the decoder capture $\mathbf{a}^{i_{1:m}}$ with triangular matrices where $w(q^{i_r}, k^{i_j}) = 0$ for $r < j$ (see a visual illustration in Appendix A). With the properly masked attention mechanism, the decoder can safely output the policy $\pi_{\theta}^{i_{m+1}}(\mathbf{a}^{i_{m+1}} | \hat{\mathbf{o}}^{i_{1:n}}, \mathbf{a}^{i_{1:m}})$, which finishes the implementation of Theorem (1).

The monotonic improvement guarantee. An MAT agent i_m optimises a trust-region objective that is conditioned on new decisions of agents $i_{1:m-1}$ by means of conditioning its policy ratio on them (see Equation (5)). As such, it increases the joint return monotonically like if it followed the sequential update scheme of HAPPO [14, Theorem 2]. However, as oppose to that method, the MAT model does not require i_m to wait until its predecessors make their updates, nor it uses their updated action distribution for importance sampling calculations. In fact, as actions of all agents are outputs of MAT, their clipping objectives can be computed in parallel (during training), thus dominating HAPPO on the time complexity. Lastly, to assure that the limiting joint policy is such that none of the agents is incentivized to change its policy (Nash equilibrium), MAT requires permutating the sequential order of updates at every iteration, which is inline with the discovery in HAPPO [14, Theorem 3].

5 Experiments and Results

MAT provides a new solution paradigm for cooperative MARL problems. The key insights of MAT are the sequential update scheme, which is inspired by Theorem (1), as well as the encoder-decoder architecture, which provides a highly-efficient implementation for a sequence modeling perspective. Importantly, MAT inherits the monotonic improvement guarantee, and agents' policies can be learned in parallel during training. We firmly believe MAT will become a game changer for MARL studies.

To evaluate if MAT meets our expectations, we test MAT on the StarCraftII Multi-Agent Challenge (SMAC) benchmark [23] where MAPPO with parameter sharing [33] has shown superior performance, and the Multi-Agent MuJoCo benchmark [7] where HAPPO [14] shows the current state-of-the-art performance. SMAC and MuJoCo environments are common benchmarks in the MARL field. On top of them, we also test MAT on the Bimanual Dexterous Hands Manipulation (Bi-DexHands) [6] which provides a list of challenging bimanual manipulation tasks (see Figure (3)), and the Google Research Football [15] benchmark with a series of cooperation scenarios in football game.

Table 1: Performance evaluations of win rate and standard deviation on the SMAC benchmark.

Task	Difficulty	MAT	MAT-Dec	MAPPO	HAPPO	QMIX	Steps
1c3s5z	Easy	**100.0**(2.4)	**100.0**(0.4)	**100.0**(2.2)	97.5(1.8)	96.9(1.5)	2e6
MMM	Easy	**100.0**(2.2)	98.1(2.1)	95.6(4.5)	81.2(22.9)	91.2(3.2)	2e6
3s5z	Hard	**100.0**(1.9)	**100.0**(3.3)	72.5(26.5)	90.0(3.5)	84.3(5.4)	3e6
8m vs 9m	Hard	**100.0**(3.1)	95.0(4.6)	93.8(3.5)	86.2(4.4)	92.6(4.0)	5e6
10m vs 11m	Hard	**100.0**(1.4)	**100.0**(2.0)	96.3(5.8)	77.5(9.7)	95.8(6.1)	5e6
25m	Hard	**100.0**(1.3)	86.9(5.6)	**100.0**(2.7)	0.6(0.8)	90.2(9.8)	2e6
27m vs 30m	Super Hard	**100.0**(0.7)	95.3(2.2)	93.1(3.2)	0.0(0.0)	39.2(8.8)	1e7
MMM2	Super Hard	**93.8**(2.6)	91.2(5.3)	81.8(10.1)	0.3(0.4)	88.3(2.4)	1e7
6h vs 8z	Super Hard	**98.8**(1.3)	93.8(4.7)	88.4(5.7)	0.0(0.0)	9.7(3.1)	1e7

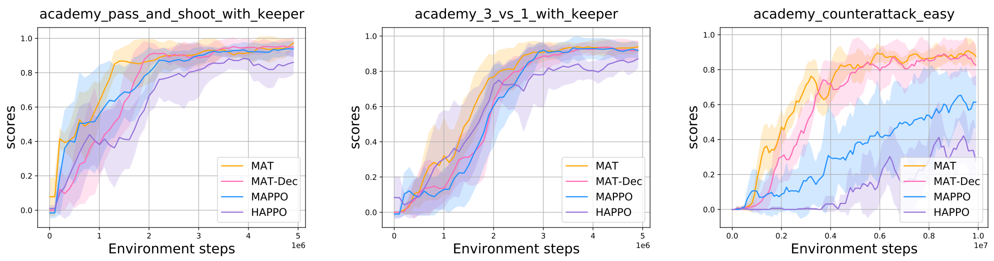


Figure 5: Performance comparison on the Google Research Football tasks with 2-4 agents from left to right respectively, telling the same conclusion that MAT outperforms MAPPO and HAPPO.

Furthermore, since Transformer-based models often demonstrate strong generalization performance on few-short tasks [3, 8], we believe MAT can possess strong generalization ability on unseen MARL tasks as well. To validate such an assumption, we design zero-shot and few-shot experiments on SMAC and multi-agent MuJoCo tasks. For SMAC tasks, we pre-train agents on eight tasks involving five types of units (*3m*, *8m vs 9m*, *10m vs 11m*, *25m*, *3s vs 3z*, *2s3z*, *3s5z*, *MMM*) with 10M examples in total and then apply them on six separate and much harder tasks (*5m vs 6m*, *8m*, *27m vs 30m*, *2s vs 1sc*, *1c3s5z*, *MMM2*) including seven types of units. This setting is designed to evaluate the generalization ability of MAT when training on simple tasks but transferring to more diverse and complex downstream tasks. In terms of multi-agent MuJoCo, we reuse the models trained on the complete HalfCheetah robot as the pre-trained agent and then directly apply it to six new tasks, each with a different leg being disfunctioned (see Figure (3a)). We investigate the generaliszation capability of pre-trained models on each downstream task with 0% (zero-shot), 1%, 5%, 10% few-short new examples, respectively. Note that common MARL baselines such as HAPPO assume fixed number of agents during training, thus it cannot directly handle the cases with changing number of agents.

We apply the same hyper-parameters of baseline algorithms from their original paper to ensure their best performance, and adopt the same hyper-parameter tuning process for our methods with details in Appendix B. To ensure fair comparisons to CTDE methods, we also introduce a CTDE-variant of MAT called **MAT-Dec**, which essentially adopts a fully decentralized actor for each individual agent (rather than using the decoder proposed in MAT) while keeping the encoder fixed. The critic's loss for MAT-Dec is $L(\phi) = \frac{1}{T}\sum_{t=0}^{T-1}\left[R(\mathbf{o}_t, \mathbf{a}_t) + \gamma\frac{1}{n}\sum_{m=1}^{n} V_{\bar{\phi}}(\hat{\mathbf{o}}_{t+1}^{i_m}) - \frac{1}{n}\sum_{m=1}^{n} V_{\phi}(\hat{\mathbf{o}}_{t}^{i_m})\right]^2$, and we apply the local advantage estimation $A_t(\hat{\mathrm{o}}_t^{i_m}, a^{i_m})$ to guide the subsequent policy update.

5.1 Performance on Cooperative MARL Benchmarks

According to Table (1) and Figure (4), MAPPO significantly outperforms HAPPO in SMAC with higher sample efficiency. This verified the homogeneity of SMAC agents and the heterogeneity of multi-agent MuJoCo agents, that are also discovered by Kuba et al. [14]. Take the SMAC task *25m* as an example, all the marines are equivalent and interchangeable so that agents can learn from their teammate's experience. Sharing parameters in this settings means leveraging 25 times more examples to train each agents comparing with separated network of HAPPO, and thus enjoying higher learning efficiency. On the other hand, with the heterogeneous settings of multi-agent MuJoCo, training a "foot" agent with experience from a "thigh" agent can surely harm its performance since they

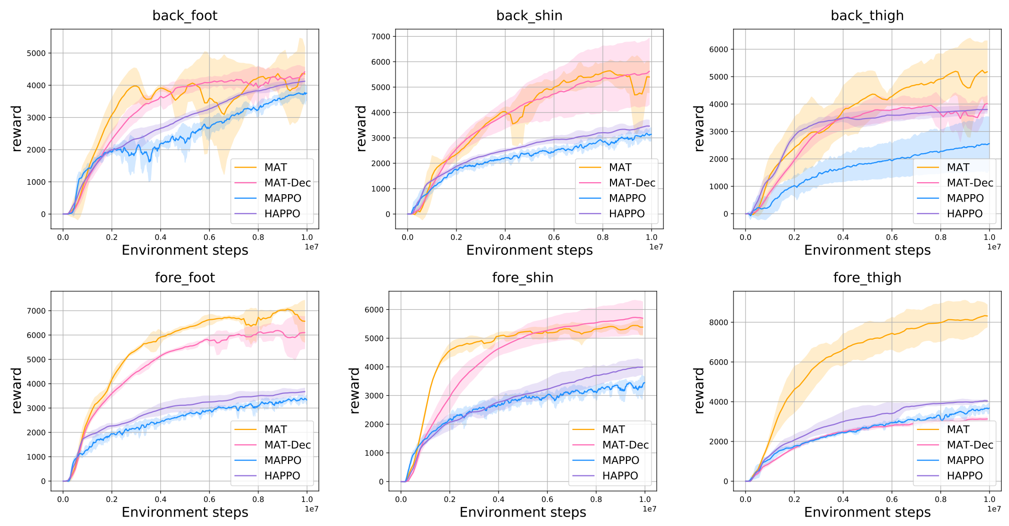


Figure 6: Performance on the HalfCheetah task with different disabled joints shown in Figure (3a).

Table 2: Median evaluation win rate and the standard deviation on the SMAC benchmark for pre-trained models with different number of online examples.

Methods	MAT				MAPPO				MAT-from scratch			
#examples	0%	1%	5%	10%	0%	1%	5%	10%	0%	1%	5%	10%
5m vs 6m	0.0$_{(0.0)}$	0.0$_{(0.0)}$	**5.8**$_{(3.1)}$	18.8$_{(7.1)}$	0.0$_{(0.0)}$	0.0$_{(0.0)}$	4.3$_{(3.8)}$	**21.9**$_{(12.2)}$	0.0$_{(0.0)}$	0.0$_{(0.0)}$	1.9$_{(1.3)}$	3.8$_{(2.1)}$
8m	100$_{(0.0)}$	100$_{(1.2)}$	100$_{(0.3)}$	100$_{(2.1)}$	100$_{(0.0)}$	100$_{(1.4)}$	100$_{(0.3)}$	100$_{(1.4)}$	0.0$_{(0.0)}$	10.6$_{(23.8)}$	92.5$_{(3.7)}$	100$_{(1.4)}$
27m vs 30m	0.0$_{(0.0)}$	6.3$_{(2.4)}$	**53.8**$_{(16.4)}$	**71.2**$_{(8.2)}$	**9.4**$_{(3.6)}$	**15**$_{(5.9)}$	26.2$_{(7.8)}$	26.8$_{(9.7)}$	0.0$_{(0.0)}$	0.0$_{(0.0)}$	0.0$_{(0.3)}$	0.3$_{(15.6)}$
2s vs 1sc	0.0$_{(0.0)}$	15.6$_{(13.8)}$	100$_{(9.7)}$	100$_{(0.0)}$	0.0$_{(0.0)}$	**43.1**$_{(17.6)}$	100$_{(1.1)}$	100$_{(1.8)}$	0.0$_{(0.0)}$	19.3$_{(33.3)}$	96.3$_{(6.2)}$	100$_{(0.3)}$
1c3s5z	**3.1**$_{(1.8)}$	**5.6**$_{(5.0)}$	**82.5**$_{(5.5)}$	**100**$_{(2.7)}$	**3.1**$_{(1.8)}$	4.3$_{(4.9)}$	73.8$_{(13.0)}$	97.5$_{(2.1)}$	0.0$_{(0.0)}$	7.5$_{(4.8)}$	87.5$_{(3.9)}$	100$_{(1.4)}$
MMM2	0.0$_{(3.6)}$	0.0$_{(1.8)}$	**33.8**$_{(13.7)}$	**62.5**$_{(12.1)}$	0.0$_{(0.0)}$	0.0$_{(1.4)}$	13.8$_{(7.0)}$	36.2$_{(9.6)}$	0.0$_{(0.0)}$	0.0$_{(0.0)}$	0.0$_{(0.0)}$	0.0$_{(0.7)}$

Table 3: Average evaluation score and standard deviation on Multi-Agent MuJoCo for pre-trained models with different number of online examples.

Methods	MAT				MAPPO				MAT-from scratch			
#examples	0%	1%	5%	10%	0%	1%	5%	10%	0%	1%	5%	10%
back foot	2100$_{(89)}$	2837$_{(95)}$	**4691**$_{(235)}$	**5646**$_{(79)}$	**2936**$_{(301)}$	**3017**$_{(135)}$	3221$_{(119)}$	3304$_{(129)}$	-0.44$_{(0.4)}$	-5.18$_{(11)}$	670$_{(1098)}$	1635$_{(1184)}$
back shin	**4005**$_{(316)}$	**4143**$_{(230)}$	**6077**$_{(209)}$	**7176**$_{(74)}$	2406$_{(32)}$	2542$_{(108)}$	2796$_{(137)}$	2955$_{(127)}$	-0.31$_{(0.1)}$	-3.95$_{(17)}$	743$_{(537)}$	1252$_{(1123)}$
back thigh	**5361**$_{(45)}$	**5641**$_{(150)}$	**7101**$_{(119)}$	**7460**$_{(61)}$	3043$_{(79)}$	3060$_{(143)}$	3217$_{(33)}$	3353$_{(71)}$	-0.54$_{(0.3)}$	-4.87$_{(7.7)}$	930$_{(589)}$	2067$_{(861)}$
fore foot	**1313**$_{(512)}$	**1955**$_{(232)}$	**4856**$_{(146)}$	**6054**$_{(172)}$	623$_{(44)}$	970$_{(185)}$	2025$_{(371)}$	2480$_{(239)}$	-0.37$_{(0.2)}$	-2.25$_{(7.9)}$	1821$_{(157)}$	2877$_{(106)}$
fore shin	**2435**$_{(13)}$	**2617**$_{(71)}$	**3851**$_{(57)}$	**4373**$_{(83)}$	1715$_{(55)}$	2457$_{(125)}$	3096$_{(59)}$	3310$_{(54)}$	-0.15$_{(0.06)}$	-0.96$_{(6.0)}$	1461$_{(101)}$	3003$_{(316)}$
fore thigh	**5631**$_{(321)}$	**6448**$_{(417)}$	**7952**$_{(109)}$	**8347**$_{(81)}$	3087$_{(110)}$	3171$_{(83)}$	3340$_{(52)}$	3519$_{(59)}$	-0.29$_{(0.3)}$	0.82$_{(14)}$	1021$_{(177)}$	2600$_{(215)}$

represent different functions on the Cheetah. However, MAT outperforms MAPPO and HAPPO in almost all tasks in Table (1) and Figure (4), indicating its modeling capability on both homogeneous and heterogeneous-agent tasks. MAT also enjoys the superior performance over MAT-Dec, which emphasize the importance of the decoder architecture in the MAT design. On the Bi-DexHands tasks, MAT outperforms MAPPO and HAPPO methods by a large margin. We save the Google Football results to Figure (5), where the conclusion stays the same.

5.2 MAT as Excellent Few-short Learners

We summarize the zero-shot and few-shot results of each algorithm in Table (2) and (3), where the bold number indicates the best performance. We also provide the performance of MAT if it was given the same amount of data but is trained from scratch as the control group. As both tables suggest, bold numbers are mainly located in the area of MAT, which justifies MAT's strong generalization performance as a few-short learner. Surprisingly, we find that the few-shot MAT with only 10% data show even higher rewards than its counterpart that is purely trained on HalfCheetah with the same disabled joints (*back foot*, *back shin* and *back thigh*) and 100% full amount data, we believe it is because the pre-train process offers initial weights that are not only closer to optimum but also less likely to stuck in bad local optima than random initialization.

6 Conclusion

In the past five years, large sequence models have achieved remarkable successes on solving visual language tasks. In this paper, we take the initial effort to build the connection between multi-agent reinforcement learning (MARL) problems and generic sequence models (SM), with the ambition that MARL researchers can hereafter benefit from the prosperous development on the sequence modeling side. Specifically, we contribute by unifying a general solution to cooperative MARL problems into a Transformer like encoder-decoder model. The proposed Multi-Agent Transformer (MAT) leverages the multi-agent advantage decomposition theorem, which essentially transforms the joint policy optimization process into a sequential decision making process that can be simply implemented by an auto-regressive model. We have demonstrated MAT's strong empirical performance on three challenging benchmarks against current state-of-the-art MARL solutions including MAPPO and HAPPO. Based on the established connection between MARL and SMs, in the future, we plan to bring multi-agent learning tasks into large multi-modal SMs, chasing for more generally intelligent models as the most recent success of GATO has already demonstrated [22].

References

[1] Jean-Baptiste Alayrac, Jeff Donahue, Pauline Luc, Antoine Miech, Iain Barr, Yana Hasson, Karel Lenc, Arthur Mensch, Katie Millican, Malcolm Reynolds, et al. Flamingo: a visual language model for few-shot learning. *arXiv preprint arXiv:2204.14198*, 2022.

[2] Rishi Bommasani, Drew A Hudson, Ehsan Adeli, Russ Altman, Simran Arora, Sydney von Arx, Michael S Bernstein, Jeannette Bohg, Antoine Bosselut, Emma Brunskill, et al. On the opportunities and risks of foundation models. *arXiv preprint arXiv:2108.07258*, 2021.

[3] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *NeurIPS*, 2020.

[4] Yu-Han Chang, Tracey Ho, and Leslie Kaelbling. All learning is local: Multi-agent learning in global reward games. *Advances in neural information processing systems*, 16, 2003.

[5] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling. *arXiv preprint arXiv:2106.01345*, 2021.

[6] Yuanpei Chen, Shengjie Wang, and Yaodong Yang. Project webpage. *https://github.com/PKU-MARL/DexterousHands*.

[7] Christian Schröder de Witt, Bei Peng, Pierre-Alexandre Kamienny, Philip H. S. Torr, Wendelin Böhmer, and Shimon Whiteson. Deep multi-agent reinforcement learning for decentralized continuous cooperative control. *CoRR*, abs/2003.06709, 2020.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT (1)*, 2019.

[9] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020.

[10] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

[11] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. *arXiv preprint arXiv:2111.06377*, 2021.

[12] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. *Advances in Neural Information Processing Systems*, 34, 2021.

[13] Jakub Kuba, Muning Wen, Linghui Meng, Haifeng Zhang, David Mguni, Jun Wang, Yaodong Yang, et al. Settling the variance of multi-agent policy gradients. *Advances in Neural Information Processing Systems*, 34, 2021.

[14] Jakub Grudzien Kuba, Ruiqing Chen, Munning Wen, Ying Wen, Fanglei Sun, Jun Wang, and Yaodong Yang. Trust region policy optimisation in multi-agent reinforcement learning. *ICLR*, 2022.

[15] Karol Kurach, Anton Raichuk, Piotr Stańczyk, Michał Zając, Olivier Bachem, Lasse Espeholt, Carlos Riquelme, Damien Vincent, Marcin Michalski, Olivier Bousquet, et al. Google research football: A novel reinforcement learning environment. *arXiv preprint arXiv:1907.11180*, 2019.

[16] Xueguang Lyu, Yuchen Xiao, Brett Daley, and Christopher Amato. Contrasting centralized and decentralized critics in multi-agent reinforcement learning. *arXiv preprint arXiv:2102.04402*, 2021.

[17] Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. *Advances in Neural Information Processing Systems*, 32, 2019.

[18] Prakash M Nadkarni, Lucila Ohno-Machado, and Wendy W Chapman. Natural language processing: an introduction. *Journal of the American Medical Informatics Association*, 18(5):544–551, 2011.

[19] Joni Pajarinen and Jaakko Peltonen. Periodic finite state controllers for efficient pomdp and dec-pomdp planning. *Advances in neural information processing systems*, 24, 2011.

[20] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In *International Conference on Machine Learning*, pages 8821–8831. PMLR, 2021.

[21] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 4295–4304. PMLR, 2018.

[22] Scott Reed, Konrad Zolna, Emilio Parisotto, Sergio Gomez Colmenarejo, Alexander Novikov, Gabriel Barth-Maron, Mai Gimenez, Yury Sulsky, Jackie Kay, Jost Tobias Springenberg, Tom Eccles, Jake Bruce, Ali Razavi, Ashley Edwards, Nicolas Heess, Yutian Chen, Raia Hadsell, Oriol Vinyals, Mahyar Bordbar, and Nando de Freitas. A generalist agent. *arXiv preprint arXiv:2205.06175*, 2022.

[23] Mikayel Samvelyan, Tabish Rashid, Christian Schroeder de Witt, Gregory Farquhar, Nantas Nardelli, Tim GJ Rudner, Chia-Man Hung, Philip HS Torr, Jakob Foerster, and Shimon Whiteson. The starcraft multi-agent challenge. 2019.

[24] John Schulman, Philipp Moritz, Sergey Levine, Michael Jordan, and Pieter Abbeel. High-dimensional continuous control using generalized advantage estimation. *arXiv preprint arXiv:1506.02438*, 2015.

[25] John Schulman, F. Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *ArXiv*, abs/1707.06347, 2017.

[26] Kyunghwan Son, Daewoo Kim, Wan Ju Kang, David Earl Hostallero, and Yung Yi. Qtran: Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *International Conference on Machine Learning*, pages 5887–5896. PMLR, 2019.

[27] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction*. MIT press, 2018.

[28] Hugo Touvron, Matthieu Cord, Alexandre Sablayrolles, Gabriel Synnaeve, and Hervé Jégou. Going deeper with image transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 32–42, 2021.

[29] Zhengzhong Tu, Hossein Talebi, Han Zhang, Feng Yang, Peyman Milanfar, Alan Bovik, and Yinxiao Li. Maxvit: Multi-axis vision transformer. *arXiv preprint arXiv:2204.01697*, 2022.

[30] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[31] Wenxiao Wang, Lu Yao, Long Chen, Binbin Lin, Deng Cai, Xiaofei He, and Wei Liu. Crossformer: A versatile vision transformer hinging on cross-scale attention. *arXiv preprint arXiv:2108.00154*, 2021.

[32] Yaodong Yang and Jun Wang. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv preprint arXiv:2011.00583*, 2020.

[33] Chao Yu, A. Velu, Eugene Vinitsky, Yu Wang, A. Bayen, and Yi Wu. The surprising effectiveness of mappo in cooperative, multi-agent games. *ArXiv*, abs/2103.01955, 2021.

Appendices

A Algorithm Details

A.1 Pseudo Code of Multi-Agent Transformer.

Algorithm 1 Multi-Agent Transformer

```
1: Input: Stepsize $\alpha$, batch size $B$, number of agents $n$, episodes $K$, steps per episode $T$.
2: Initialize: Encoder $\{\phi_0\}$, Decoder $\{\theta_0\}$, Replay buffer $\mathcal{B}$.
3: for $k = 0, 1, \ldots, K-1$ do
4:     for $t = 0, 1, \ldots, T-1$ do
5:         Collect a sequence of observations $o_t^{i_1}, \ldots, o_t^{i_n}$ from environments.
           {// The Inference Phase}
6:         Generate representation sequence $\hat{o}_t^{i_1}, \ldots, \hat{o}_t^{i_n}$ by feeding observations to the encoder.
7:         Input $\hat{o}_t^{i_1}, \ldots, \hat{o}_t^{i_n}$ to the decoder.
8:         for $m = 0, 1, \ldots, n-1$ do
9:             Input $a_t^{i_0}, \ldots, a_t^{i_m}$ and infer $a_t^{i_{m+1}}$ with the auto-regressive decoder.
10:        end for
11:        Execute joint actions $a_t^{i_0}, \ldots, a_t^{i_n}$ in environments and collect the reward $R(\boldsymbol{o}_t, \boldsymbol{a}_t)$.
12:        Insert $(\boldsymbol{o}_t, \boldsymbol{a}_t, R(\boldsymbol{o}_t, \boldsymbol{a}_t))$ in to $\mathcal{B}$.
13:    end for
       {// The Training Phase}
14:    Sample a random minibatch of $B$ steps from $\mathcal{B}$.
15:    Generate $V_\phi(\hat{o}^{i_1}), \ldots, V_\phi(\hat{o}^{i_n})$ with the output layer of the encoder.
16:    Calculate $L_{\text{Encoder}}(\phi)$ with Equation (4).
17:    Compute the joint advantage function $\hat{A}$ based on $V_\phi(\hat{o}^{i_1}), \ldots, V_\phi(\hat{o}^{i_n})$ with GAE.
18:    Input $\hat{o}^{i_1}, \ldots, \hat{o}^{i_n}$ and $a^{i_0}, \ldots, a^{i_{n-1}}$, generate $\pi_\theta^{i_1}, \ldots, \pi_\theta^{i_n}$ at once with the decoder.
19:    Calculate $L_{\text{Decoder}}(\theta)$ with Equation (5).
20:    Update the encoder and decoder by minimising $L_{\text{Encoder}}(\phi) + L_{\text{Decoder}}(\theta)$ with gradient descent.
21: end for
```

A.2 Dynamic Process and Source Code of MAT

Please refer to `https://sites.google.com/view/multi-agent-transformer`

B Hyper-parameter Settings for Experiments

During experiments, the implementations of baseline methods are consistent with their official repositories, all hyper-parameters left unchanged at the origin best-performing status. The hyper-parameters adopted for different algorithms and tasks are listed in Table 4-12. In particular, the *ppo epochs* and *ppo clip* across different SMAC scenarios are unified to 10 and 0.05 respectively for pre-training and fine-tuning in few-shot experiments.

Table 4: Common hyper-parameters used for MAT, MAT-Dec, MAPPO and HAPPO in the SMAC domain.

hyper-parameters	value	hyper-parameters	value	hyper-parameters	value
critic lr	5e-4	actor lr	5e-4	stacked frames	1
gain	0.01	optim eps	1e-5	batch size	3200
training threads	16	num mini-batch	1	rollout threads	32
entropy coef	0.01	max grad norm	10	episode length	100
optimizer	Adam	hidden layer dim	64	use huber loss	True

Table 5: Different hyper-parameters used for MAT and MAT-Dec in the SMAC domain.

Maps	ppo epochs	ppo clip	num blocks	num heads	steps	gamma
1c3s5z	10	0.2	1	1	2e6	0.99
MMM	15	0.2	1	1	2e6	0.99
3s5z	10	0.2	1	1	3e6	0.99
8m vs 9m	10	0.05	1	1	5e6	0.99
10m vs 11m	10	0.05	1	1	5e6	0.99
25m	15	0.05	1	1	2e6	0.99
27m vs 30m	5	0.2	1	1	1e7	0.99
MMM2	5	0.05	1	1	1e7	0.99
6h vs 8z	15	0.05	1	1	1e7	0.99

Table 6: Different hyper-parameters used for MAPPO and HAPPO in the SMAC domain.

Maps	ppo epochs	ppo clip	hidden leyer	steps	$\text{gamma}_{\text{MAPPO}}$	$\text{gamma}_{\text{HAPPO}}$
1c3s5z	15	0.2	2	2e6	0.99	0.95
MMM	15	0.2	2	2e6	0.99	0.95
3s5z	5	0.2	2	3e6	0.99	0.95
8m vs 9m	15	0.05	2	5e6	0.99	0.95
10m vs 11m	10	0.2	2	5e6	0.99	0.95
25m	10	0.2	2	2e6	0.99	0.95
27m vs 30m	5	0.2	2	1e7	0.99	0.95
MMM2	5	0.2	2	1e7	0.99	0.95
6h vs 8z	5	0.2	2	1e7	0.99	0.95

Table 7: Common hyper-parameters used for all methods in the multi-agent MuJoCo domain.

hyper-parameters	value	hyper-parameters	value	hyper-parameters	value
gamma	0.99	steps	1e7	stacked frames	1
gain	0.01	optim eps	1e-5	batch size	4000
training threads	16	num mini-batch	40	rollout threads	40
entropy coef	0.001	max grad norm	0.5	episode length	100
optimizer	Adam	hidden layer dim	64	use huber loss	True

Table 8: Different hyper-parameter used in the mulit-agent MuJoCo domain.

Maps	MAT	MAT-Dec	MAPPO	HAPPO
critic lr	5e-5	5e-5	5e-3	5e-3
actor lr	5e-5	5e-5	5e-6	5e-6
ppo epochs	10	10	5	5
ppo clip	0.05	0.05	0.2	0.2
num hidden layer	/	/	2	2
num blocks	1	1	/	/
num head	1	1	/	/

Table 9: Common hyper-parameters used for all methods in the Bi-DexHands domain.

hyper-parameters	value	hyper-parameters	value	hyper-parameters	value
gamma	0.96	steps	5e7	stacked frames	1
gain	0.01	optim eps	1e-5	ppo epochs	5
ppo clip	0.2	num mini-batch	1	rollout threads	80
batch size	6000	episode length	75	optimizer	Adam
entropy coef	0.001	max grad norm	0.5	training threads	16

Table 10: Different hyper-parameter used in the Bi-DexHands domain.

Maps	MAT	MAT-Dec	MAPPO	HAPPO
critic lr	5e-5	5e-5	5e-4	5e-4
actor lr	5e-5	5e-5	5e-4	5e-4
hidden dim	64	64	512	512
num hidden layer	/	/	1	1
num blocks	1	1	/	/
num head	1	1	/	/

Table 11: Common hyper-parameters used for all methods in the Google Research Football domain.

hyper-parameters	value	hyper-parameters	value	hyper-parameters	value
critic lr	5e-4	actor lr	5e-4	gamma	0.99
gain	0.01	optim eps	1e-5	batch size	4000
training threads	16	num mini-batch	1	rollout threads	20
entropy coef	0.01	max grad norm	0.5	episode length	200
optimizer	Adam	hidden layer dim	64	stacked frames	1

Table 12: Different hyper-parameter used in the Google Research Football domain.

Maps	MAT	MAT-Dec	MAPPO	HAPPO
ppo epochs	10	10	5	5
ppo clip	0.05	0.05	0.2	0.2
num hidden layer	/	/	2	2
num blocks	1	1	/	/
num head	1	1	/	/

C Details of Experimental Results

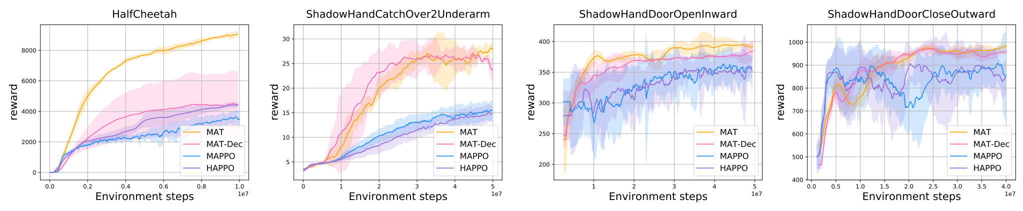


Figure 7: Performance comparisons on the Multi-Agent MuJoCo and the Bi-DexHands benchmarks, showing MAT's advantages in robot control.

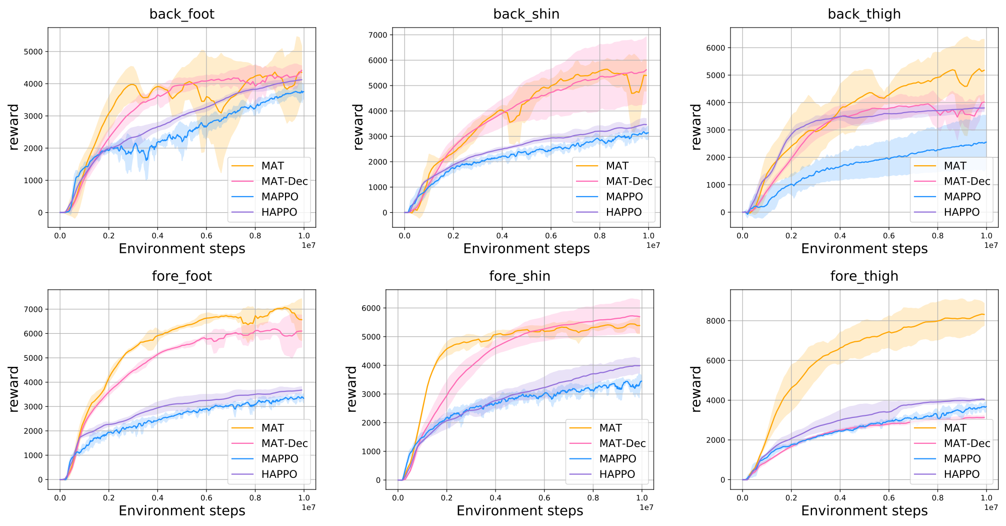


Figure 8: Performance comparison on the HalfCheetah tasks with different disabled joints, where MAT significantly outperformed baseline methods. Together with performance on the complete HalfCheetah in Figure (7), it emphasises MAT's capability for heterogeneous-agent tasks (agents are not interchangeable).

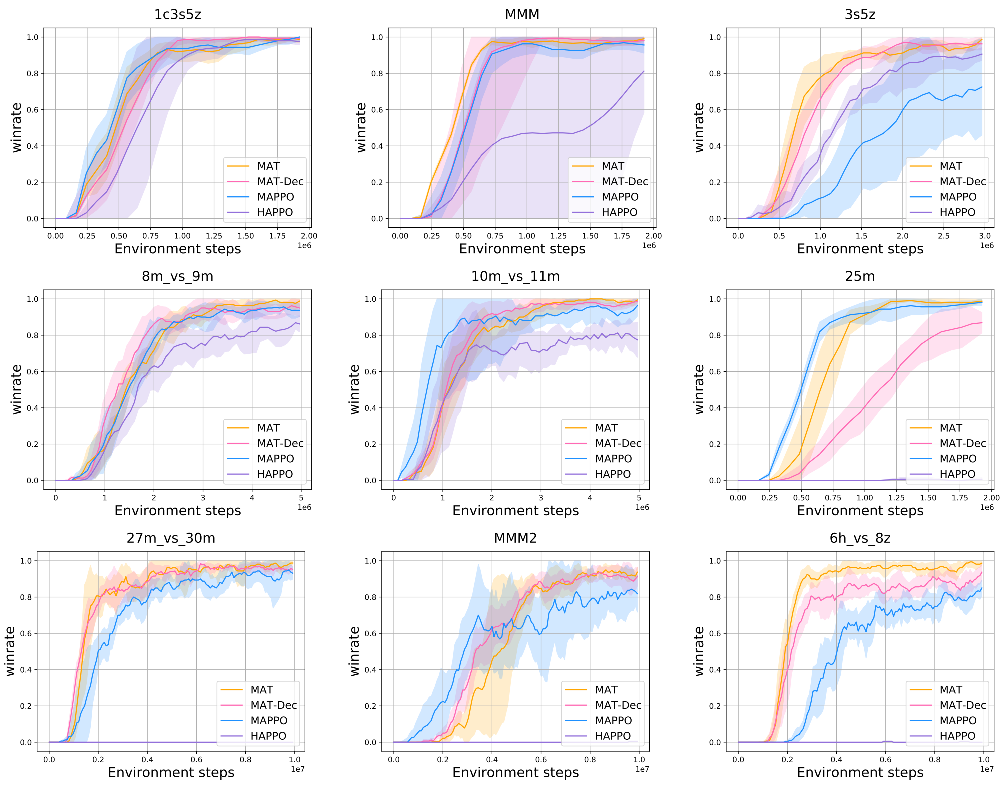


Figure 9: Performance comparison on SMAC tasks. MAT consistently outperforms its rivals, indicating its modeling capability for homogeneous-agent tasks (agents are interchangeable).

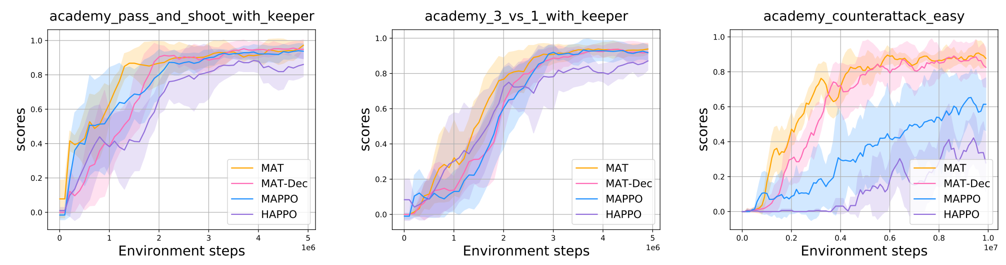


Figure 10: Performance comparison on the Google Research Football tasks with 2-4 agents from left to right respectively, telling the same conclusion that MAT outperforms MAPPO and HAPPO.

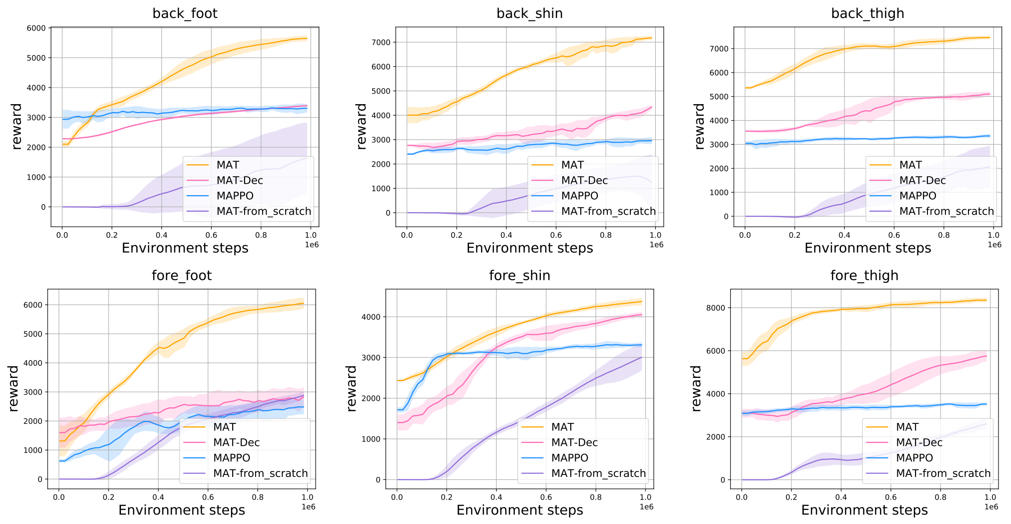


Figure 11: Few-shot performance comparison with pre-trained models on multi-agent MuJoCo tasks. MAT exhibits powerful generalisation capability when parts of the robot fail.

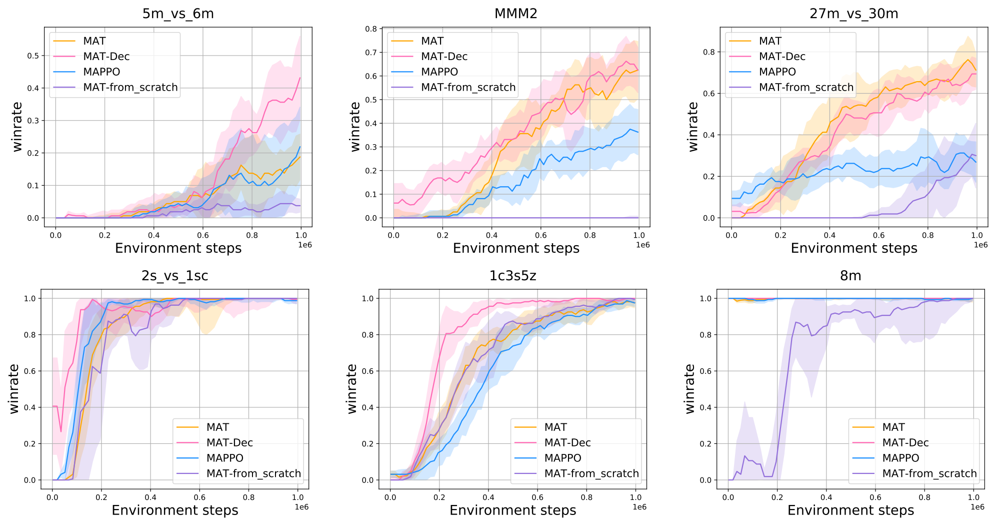


Figure 12: Few-shot performance comparison with pre-trained models on SMAC tasks. Sequence-modeling-based methods, MAT and MAT-Dec, enjoy superior performance over MAPPO, justifying their strong generalisation capability as few-shot learners.