UPDeT: Universal Multi-agent Reinforcement Learning via Policy Decoupling with Transformers

Anonymous authors
Paper under double-blind review

Abstract

Recent advances in multi-agent reinforcement learning have been largely limited in training one model from scratch for every new task. The limitation is due to the restricted model architecture related to fixed input and output dimensions. This hinders the experience accumulation and transfer of the learned agent over tasks with diverse levels of difficulty (e.g. 3 vs 3 or 5 vs 6 multi-agent games). In this paper, we make the first attempt to explore a universal multi-agent reinforcement learning pipeline, designing one single architecture to fit tasks with the requirement of different observation and action configurations. Unlike previous RNN-based models, we utilize a transformer-based model to generate a flexible policy by decoupling the policy distribution from the intertwined input observation with an importance weight measured by the merits of the self-attention mechanism. Compared to a standard transformer block, the proposed model, named as Universal Policy Decoupling Transformer (UPDeT), further relaxes the action restriction and makes the multi-agent task's decision process more explainable. UPDeT is general enough to be plugged into any multi-agent reinforcement learning pipeline and equip them with strong generalization abilities that enables the handling of multiple tasks at a time. Extensive experiments on large-scale SMAC multi-agent competitive games demonstrate that the proposed UPDeT-based multi-agent reinforcement learning achieves significant results relative to state-of-the-art approaches, demonstrating advantageous transfer capability in terms of both performance and training speed (10 times faster).

1 Introduction

Reinforcement Learning (RL) provides a framework for decision-making problems in an interactive environment, with applications including robotics control (Hester et al. (2010)), auto-driving (Bojarski et al. (2016)) and video gaming (Mnih et al. (2015)). Cooperative multi-agent reinforcement learning (MARL), which is a long-standing problem in the RL context, involves organizing multiple agents to achieve a goal, and is thus a key tool for addressing many real-world problems, such as mastering multi-player video games (Peng et al. (2017)) and studying population dynamics (Yang et al. (2017)).

A number of methods have been proposed to exploit an action-value function to learn a multi-agent model (Du et al. (2019), Mahajan et al. (2019), Rashid et al. (2018), Zhou et al. (2020), Yang et al. (2020), Sunehag et al. (2017), Hostallero et al. (2019)). However, current methods have poor ability in representation learning and fail to exploit a common structure underlying the tasks. This is because they treat observation from different entities of the environment as an integral part of the whole. Accordingly, they give tacit consent to the assumption that neural networks can automatically decouple the observation to find the best mapping between the whole observation and policy. Thus, they treat all information from other agents or different parts of the environment in the same way. The most commonly used method involves concatenating the observation from each entity to a vector as input (Rashid et al. (2018), Du et al. (2019), Zhou et al. (2020)). In addition, current methods ignore the rich physical meanings behind each action. Multi-agent tasks have close relationship between the observation and output. If the model does not decouple the observation from different agents, individual functions can misguided and impede the centralized value function.

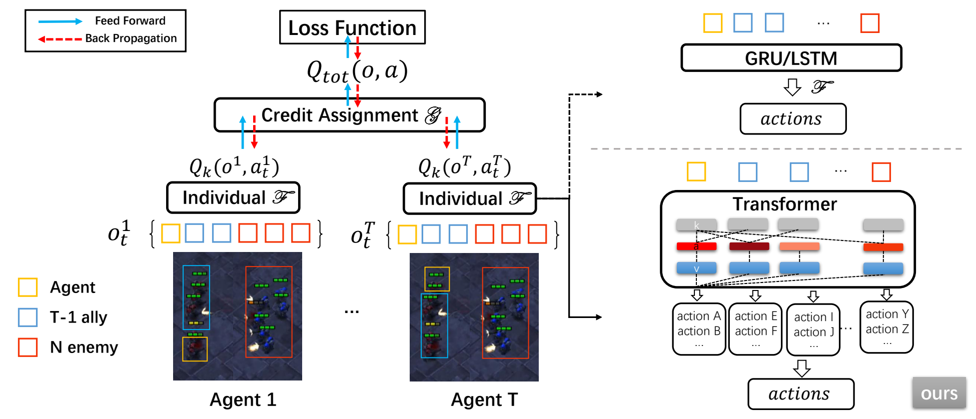


Figure 1: An overview of the MARL framework. Our work replaces the widely used GRU/LSTM-based individual value function with a transformer-based function. Actions are separated into action groups according to observations.

Worse yet, conventional models require the input and the output dimensions to be fixed (Shao et al. (2018),Wang et al. (2020)), makes the zero shot transfer impossible. Thus the application of current methods is limited in real-world applications.

Our solution to these weaknesses is to develop a multi-agent reinforcement learning (MARL) framework with no limitation on input or output dimension. Besides, the model should be general enough that can be applied to any existing MARL methods. More importantly, the model is expected to be explainable and can provide further improvement on both final performance on single-task scenario and transfer capability on multi-task scenarios.

Inspired by self-attention mechanism (Vaswani et al. (2017)), we provide a transformer-based MARL framework, named Universal Policy Decoupling Transformer (**UPDeT**). It includes four advantages: 1) Once trained, it can be universally deployed; 2) robuster representation with policy decoupling strategy; 3) more explainable; 4) general enough to be applied on any MARL model. We design a transformer-based function to handle various observation sizes by treating individual observation, as "observation-entity". We match the related observation-entity with action-groups by separate the action space into several action-groups according to the corresponding observation-entity, allowing us to get matched observation-entity — action-group pairs set. We use a self-attention mechanism to learn the relationship between the matched observation-entity with other observation-entities. Through the use of self-attention map and embedding of each observation-entity, UPDeT can optimize the policy at an action-group level. We call this strategy as **Policy Decoupling**. By combining the transformer and policy decoupling strategy, UPDeT significantly outperforms conventional RNN based model.

In UPDeT, no additional parameters need to be introduced for new tasks. We also prove that only with decoupled policy and matched observation-entity — action-group pairs can UPDeT learn a strong representation with high transfer capability. Finally, our proposed UPDeT can be plugged into any existing methods with almost no changes to the framework architecture required, and can still bring significant improvements on final performance, especially in hard and complex multi-agent tasks.

The main contributions of this work are as follows: First, our UPDeT-based MARL framework outperforms RNN-based frameworks on state-of-the-art centralized functions by a large margin in terms of final performance. Second, our model has strong transfer capability and can handle a number of different task at a time. Third, our model accelerates the transfer learning speed (total steps cost) so that it is about 10 times faster compared to RNN-based models in most scenarios.

2 Related Work

Attention mechanisms have become an integral part of models that capture global dependencies. In particular, self-attention (Parikh et al. (2016)) calculates the response at a position in a sequence by

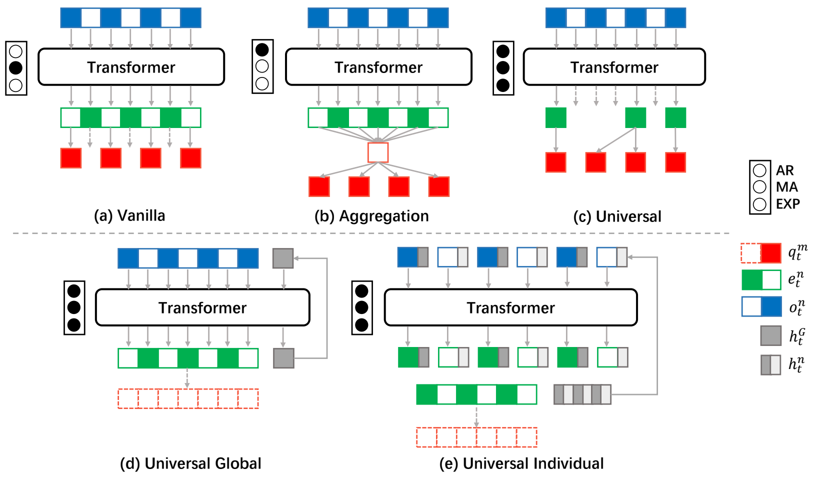


Figure 2: 3 variants on different policy decoupling methods type (upper part) and 2 variants on different temporal unit type (bottom). 'AR' , 'MA' and 'EXP' represent for Action Restriction, Multi-task at a time and EXPlainable respectively. o, e, q and h represents for observation, embedding, q-value and hidden states with n observation entities and m available actions. G represents for global hidden state and t is the current time step. Black circle indicate the variant possess this attribute and variant (d) is our proposed UPDeT with best performance . Details of all 5 variants can be found in Section 3.

attending to all positions within the same sequence. Vaswani et al. (2017) demonstrated that machine translation models could achieve state-of-the-art results by solely using a self-attention model. Parmar et al. (2018) proposed an Image Transformer model to apply self-attention to image generation. Wang et al. (2018) formalized self-attention as a non-local operation to model the spatial-temporal dependencies in video sequences. In spite of this, self-attention mechanisms has not yet been fully explored in multi-agent reinforcement learning.

Another research line is multi-agent reinforcement learning (MARL). Existing work in MARL mainly focus on building a centralized function to guide the training of individual value function (Lowe et al. (2017), Sunehag et al. (2017), Rashid et al. (2018), Mahajan et al. (2019), Hostallero et al. (2019), Yang et al. (2020), Zhou et al. (2020)). Few work has consider to form a better individual function with strong representation and transfer capability. In standard reinforcement learning, the generalization has been fully studied (Taylor & Stone (2009), Ammar et al. (2012), Parisotto et al. (2015), Gupta et al. (2017), Da Silva & Costa (2019)). While multi-agent transfer learning has been proved to be more difficult than single agent scenario (Boutsioukis et al. (2011), Shao et al. (2018), Vinyals et al. (2019)). However, transfer capability of a multi-agent system is of greater significance due to various agents number, observations size and policy distribution.

As far as we know, we are the first to develop a multi-agent framework that can handle multiple task at a time. Besides, we provide a policy decoupling strategy to further improve the model performance and facilitate the multi-agent transfer learning, which is a big step to real world multi-agent applications.

3 Method

We start by introducing the necessary notations and basic task settings for our approach. We then describe a transformer-based individual function and policy decoupling strategy under MARL. Finally, we introduce different temporal unit and assimilate our Universal Policy Decoupling Transformer (UPDeT) into Dec-POMDP.

3.1 Notations and Task Settings

Multi-agent Reinforcement Learning A cooperative multi-agent task is a decentralized partial observable Markov decision process (Oliehoek et al. (2016)) with a tuple $G =$

$\langle S, A, U, P, r, Z, O, n, \gamma \rangle$. Let S denotes the global state of the environment. Denote A as the set of n agents and U as the action space respectively. At each time step t, agent $a \in \mathbf{A} \equiv \{1, ..., n\}$ selects an action $u \in U$, forming a joint action $\mathbf{u} \in \mathbf{U} \equiv U^n$, which in turn causes a transition in the environment according to the state transition function $P(s'|s, \mathbf{u}) : S \times \mathbf{U} \times S \to [0, 1]$. All agents share the same reward function $r(s, \mathbf{u}) : S \times \mathbf{U} \to R$ and $\gamma \in [0, 1)$ is a discount factor. We consider a partially observable scenario in which each agent draws individual observations $z \in Z$ according to the observation function $O(s, a) : S \times A \to Z$. Each agent has an action-observation history that conditions a stochastic policy π^t, forming the joint action value: $Q^\pi(s_t, \mathbf{u}_t) = \mathbb{E}_{s_{t+1:\infty}, \mathbf{u}_{t+1:\infty}} [R_t | s_t, \mathbf{u}_t]$, where $R_t = \sum_{i=0}^{\infty} \gamma^i r_{t+i}$ is the discounted return.

Centralized training with decentralized execution Centralized training with decentralized execution (CTDE) is a commonly used architecture in MARL. Each agent conditions only on its own action-observation history to make decision using learned policy. The centralized value function provides centralized gradient to update individual function based on the output of individual function. Therefore, a stronger individual value function can benefit the centralized training.

3.2 Transformer based Individual Value Function

In this section, we give a mathematical formulation of our transformer-based model UPDeT. We describe the calculation of global Q-function with self-attention mechanism. At first, the observation O is embedded into an semantic embedding to handle various observation space. For example, if an agent a_i observes k other entities $\{o_{i,1}, ..., o_{i,k}\}$ at time step t, all the observation entities are embedded via an embedding layer E:

$$e_i^t = \{E(o_{i,1}^t), ..., E(o_{i,k}^t)\}. \tag{1}$$

Here i is the index of the agent, $i \in \{1, ..., n\}$. Next, the value functions $\{Q_1, ..., Q_n\}$ for the n agents for each step are estimated as follows:

$$q_i^t = Q_i(h_i^{t-1}, e_i^t, \mathbf{u}_t). \tag{2}$$

We introduce h_i^{t-1}, the temporal hidden state at last time step $t-1$ since POMDP policy highly dependent on the historical information. The e_i^t is the observation embedding. The u_i^t is the candidate action, $u_i^t \in U$. θ_i is the parameter defines the Q_i. At last, the global Q-function Q_π is calculated by all individual value functions as follows:

$$Q_\pi(s_t, \mathbf{u}_t) = F(q_1^t, .., q_n^t) \tag{3}$$

F_i is the credit assignment function for defined by ϕ_i for each agent a_i. It is widely used in Rashid et al. (2018) and Sunehag et al. (2017). For example, in VDN, F is a sum function noted as $F(q_1^t, .., q_n^t) = \sum_{i=1}^n q_i^t$.

Implement Q-function with Self-attention Vaswani et al. (2017) adopts three matrices $\mathbf{K}$, $\mathbf{Q}$, $\mathbf{V}$ representing a set of keys, quires and values respectively. The attention is computed as:

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d_k}})\mathbf{V}, \tag{4}$$

where d_k is a scaling factor that equals to the dimension of the key. In our method, we adopt the self-attention to learn the features and the relationships from the observation entity embedding and global temporal information. To learn the independent policy in decentralized multi-agent learning, we define K_i, Q_i and V_i are the key, query and value metrics for each agent a_i. We further consider query, key and value the same matrices $R_i^l = K_i = Q_i = V_i$, where $l \in \{1, ..., L\}$ is the number of layers of the transformer. Thus we formulate our transformer as :

$$\begin{aligned} R_i^1 &= \{h_i^{t-1}, e_i^t\} \\ Q_i^l, K_i^l, V_i^l &= LF_{Q,K,V}(R_i^l) \\ R_i^{l+1} &= \text{Attention}(Q_i^l, K_i^l, V_i^l). \end{aligned} \tag{5}$$

where LF represents for linear functions to compute $\mathbf{K}$, $\mathbf{Q}$, $\mathbf{V}$. At last we project the entity features of the last transformer layer R_i^L to the output space of value function Q_i. We implement the projection using a linear function P:

$$Q_i(h_i^{t-1}, e_i^t, u_i) = P(R_i^L, u_i). \tag{6}$$

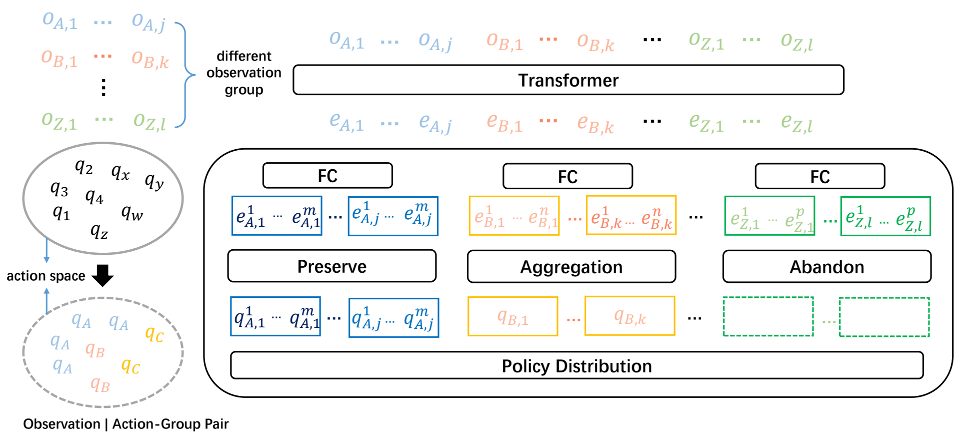


Figure 3: The main pipeline of our proposed UPDeT, where o, e, q represent for observation entity, feature embedding and Q-value of each action respectively. 3 operations are adopt to avoid introducing new parameters in forming policy distribution including 'preserve', 'aggregation' and 'abandon'. Details can be found in Section 3.3 and a more specific real case can be found in Fig. 7.

3.3 Policy Decoupling

A single transformer based individual function with self-attention mechanism is still not able to handle various policy distribution. A flexible mapping function P in Eq. 6 is needed to deal with various input and output dimension and provide strong representation ability. Using correlation between input and output, we design a strategy called policy decoupling as the key part of UPDeT.

The main idea of policy decoupling strategy can be summarized into three points:

- Point ①: No restriction on policy dimension. The output dimension of a standard transformer block has to be equal or less to input dimension. This is unacceptable in some MARL tasks as the action number can be larger than entity number.
- Point ②: Handle multi-task at a time. This requires a fixed model architecture without new parameters introduced on new tasks. Unluckily, satisfying point ① easily breaks the point ②. The difficulty lies in how to reconcile point ① and point ②.
- Point ③: Make the model more explainable. It is better if we can replace the conventional RNN based model with a more explainable structure on policy generation.

Following the above three points, we propose three policy decoupling methods namely Vanilla Transformer, Aggregation Transformer and Universal Policy Decoupling Transformer (UPDeT). The pipelines can be found in Fig. 2. The details of **Vanilla Transformer** and **Aggregation Transformer** is described in experiment section as our baselines. In this section, we mainly discuss the mechanism of our proposed **UPDeT**.

With the entity features of the last transformer layer in Eq. 5. The main challenge is to build a strong mapping between the features and the policy distribution. UPDeT first matches the input entity with related output policy part. This corresponding is easy to find in MARL task as interactive action between two agents is quite common. Once we match the corresponding entity features and actions, we largely reduce the burden of model learning representation using self-attention mechanism. Considering there might be more than one interactive actions of the matched entity feature, we separate the action space to several action groups. Each action group consists several actions matched with one entity. The pipeline of this process can be found in the left part of Fig. 3. In the mapping function, to satisfy point ① and point ②, we adopt two strategies. First, if the action-group of one entity feature contains more than one action, a shared fully connected layer is added to map the output to action number dimension. Second, if one entity feature has no corresponding action, we abandon it. There is no worry for losing the information carried by this kind of entity feature as transformer has aggregated the necessary information to each output. The pipeline of UPDeT can be found in right part of Fig. 3. With UPDeT, there is no action restriction and no new parameter introduced in new scenarios. A single model can be trained in multiple tasks and deployed universally. In addition, matching the corresponding entity feature and action-group satisfies point ③ as the policy is explainable using attention heatmap which we will discuss in Section 4.4.

3.4 Temporal Unit Structure

Yet, a transformer based individual value function with policy decoupling strategy can not handle partial observation decision process without trajectory or history information. In Dec-POMDP (Oliehoek et al. (2016)), each agent a choose an action according to $\pi^a(u^a|\tau^a)$, where u, τ represents for action and action-observation history respectively. In GRU and LSTM, we adopt a hidden state to hold the information of action-observation history. However, the combination of a transformer block and a hidden state has not been fully studied. In this section, we provide 2 approaches to handle the hidden state in UPDeT:

1) **Global** temporal unit treat hidden state as additional input of transformer block. The process is formulated similar to Eq. 5:

$$\begin{aligned} R^1 &= \{h_G^{t-1}, e_1^t\} \\ R^l &= \text{Attention}(Q^{l-1}, K^{l-1}, V^{l-1}) \\ \{h_G^t, e_L^t\} &= R^L \end{aligned} \tag{7}$$

Here we ignore the subscript i and instead use G to represent for 'global'. The global temporal unit is simple but efficient, and provide us with a robust performance in most scenarios;

2) **Individual** temporal unit treats hidden state as inner part of each entity. In other word, each input maintain its own hidden state. Each output projects a new hidden state for next time step. Individual temporal state shows a more precise control approach on history information as it splits the global hidden state into individual parts. We use j to represent for the number of entity. The process is formulated as

$$\begin{aligned} R^1 &= \{h_1^{t-1}...h_j^{t-1}, e_1^t\} \\ R^l &= \text{Attention}(Q^{l-1}, K^{l-1}, V^{l-1}) \\ \{h_1^t...h_j^t, e_L^t\} &= R^L \end{aligned} \tag{8}$$

However, this method brings extra burden on learning hidden state independently for each entity. In experiment Section 4.1.2, we test both 2 variants and make further discussion on them.

3.5 Optimization

We use standard squared $TD\ error$ in DQNs (Mnih et al. (2015)) to optimize our whole framework:

$$L(\theta) = \sum_{i=1}^{b} \left[\left(y_i^{DQN} - Q(s, u; \theta) \right)^2 \right] \tag{9}$$

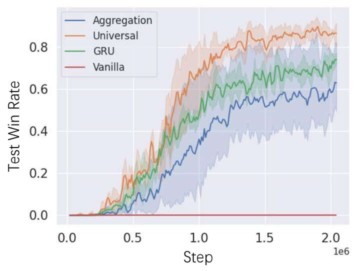
Here b represents for batch size. In partial observable settings, agents can benefit from conditioning on action-observation history. Hausknecht & Stone (2015) propose Deep Recurrent Q-networks (DRQN) to do sequential decision process. We replace the widely used GRU (Chung et al. (2014))/LSTM (Hochreiter & Schmidhuber (1997)) unit in DRQN with transformer based temporal unit and then train the whole model.

4 StarCraft II Experiment

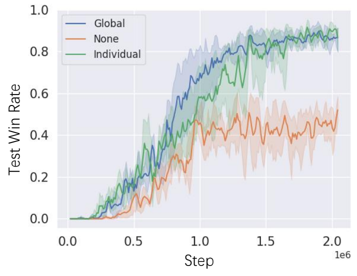
In this section, we evaluate UPDeT and its variants with different policy decoupling methods in challenging micromanagement games in StarCraft II. We compare UPDeT with RNN based model on single scenario and test transfer capability on multiple scenarios transfer tasks. The experiment results show that UPDeT achieve significant improvement compared to RNN based model.

4.1 Single Scenario

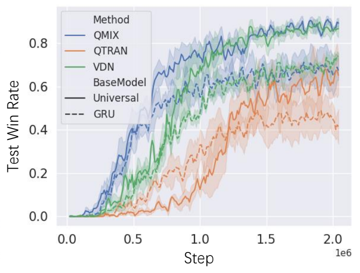
In single scenario experiments, we evaluate model performance on different scenarios from SMAC (Samvelyan et al. (2019)). Specifically, the considered scenarios contain 3 Marines vs 3 Marines



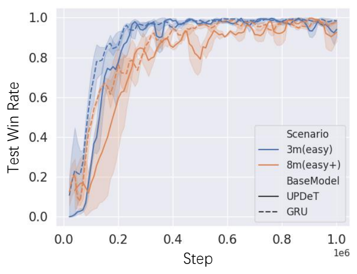
(a) Policy variants



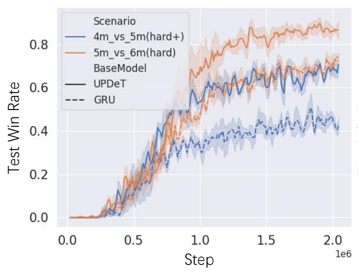
(b) Temporal variants



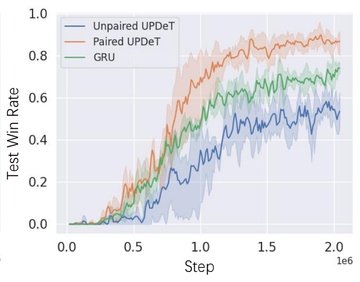
(c) MARL methods



(d) Easy scenarios



(e) Hard scenarios



(f) Mismatch experiment

Figure 4: Experiment result with different task settings. Details can be found in Section 4.1.2.

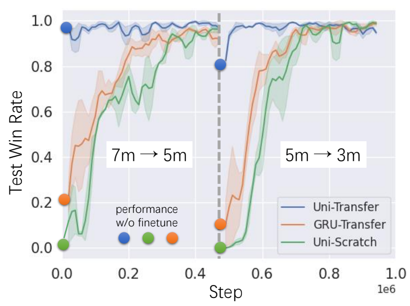
(3m, Easy), 8 Marines vs 8 Marines (8m, Easy), 4 Marines vs 5 Marines (4m_vs_5m, Hard+) and 5 Marines vs 6 Marines (5m_vs_6m, Hard). In all these games, only the units from self side are treated as agents. Dead enemy units will be masked out from the action space to ensure the executed action is valid. More detailed settings can be acquired from the SMAC environment (Samvelyan et al. (2019)).

4.1.1 Based Methods and Training Details

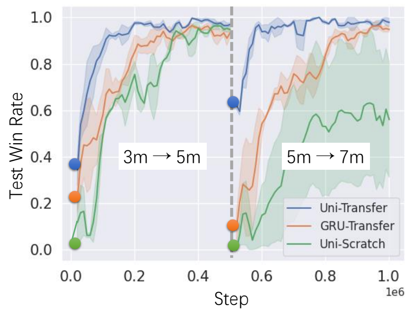
The considered MARL methods for evaluation include VDN (Sunehag et al. (2017)), QMIX (Rashid et al. (2018)) and QTRAN (Hostallero et al. (2019)). All three SOTA methods' original implementation can be found in `https://github.com/oxwhirl/pymarl`. We choose these methods because of their robust performance across different multi-agent tasks. Other methods including COMA (Foerster et al. (2017)) and IQL (Tan (1993)) do not have a stable performance in all tasks, which has been proved in many recent work (Rashid et al. (2018), Mahajan et al. (2019), Zhou et al. (2020)). Therefore, we combined UPDeT with VDN, QMIX and QTRAN to prove that our model can further improve the model performance significantly compared to GRU based model. The entire training procedure is repeated for 8 times to plot the winning rate curve with standard deviation. The results are reported in Fig. 4, where the test win rate vs. the training steps on all the battle scenarios are given.

4.1.2 Result

The result of model performances with different policy decoupling methods can be found in Fig. 4a. **Vanilla Transformer** is our baseline of all transformer based models. Vanilla transformer only satisfy the point ②. Each output embedding can be either projected to an action or abandoned. Vanilla transformer fail to converge in the experiment. **Aggregation Transformer** is a variant of vanilla transformer, whose embeddings are aggregated into a global embedding and then projected to a policy distribution. Aggregation transformer only satisfies the point ①. The performance of the aggregation transformer is worse than GRU based model. The result proves that only with policy decoupling strategy can transformer based model outperform conventional RNN based model. Next, we adopt UPDeT to find the best architecture of temporal unit in Fig. 4b. The result shows that without hidden state, the performance is decreased heavily. Temporal unit with global hidden state is more efficient on converge speed than individual hidden state. However, the final performances are almost same. To test the generalization of our model, we combine the UPDeT with VDN / QMIX / QTRAN respectively and compare the final performance with RNN based methods in Fig. 4c.



(a) Transfer from 7 marines to 3 marines



(b) Transfer from 3 marines to 7 marines

Figure 5: Experiment result on transfer learning with UPDeT (Uni-Transfer) and GRU unit (GRU-Transfer), along with UPDeT training from scratch (Uni-Scratch). At 0 time step and 500k time step, we load the model from source scenario and finetune on target scenarios. Circle point indicates the model performance on new scenarios without finetune.

We evaluate model performance on 5m_vs_6m (Hard) scenarios. Combined with UPDeT, all three MARL methods get significant improvement compared to GRU based model with a large margin. The result proves that our model can be injected into any existing state of the art methods on MARL to get higher performance. Further more, we combine UPDeT with VDN and evaluate the model performance on different scenarios from Easy to Hard+ in Fig. 4d, Fig. 4e. The result shows that the UPDeT has stable performance on easy scenarios and outperform GRU based model significantly on hard scenarios. In 4m_vs_5m(Hard+) scenario, UPDeT gains about 80% performance improvement compared to GRU based model. Finally, we do ablation study on UPDeT with paired and unpaired observation-entity—action-group. The result can be seen in Fig. 4f. We disrupt the original correspondence between 'attack' action and enemy unit. The final performance is heavily decreased compared to original one and even worse than GRU based model. To this end, we conclude that only with policy decoupling and paired observation-entity—action-group strategy can UPDeT learn a strong policy.

4.2 Multiple Scenarios

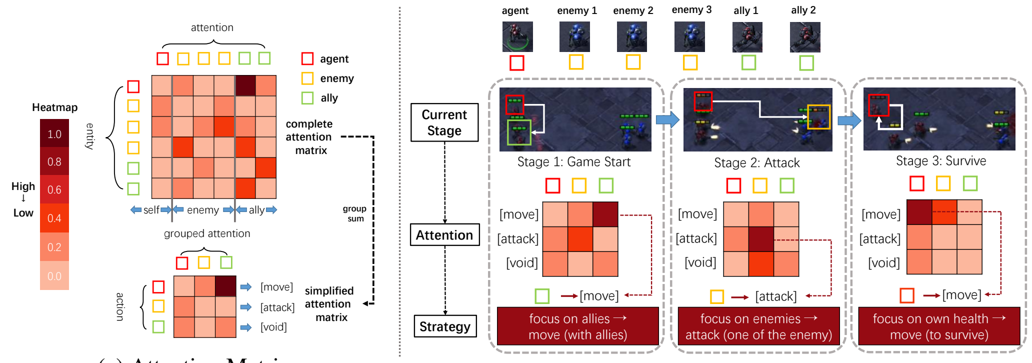
In this section, we discuss the transfer capability of UPDeT compared to RNN based model. We evaluate the model performance in a curriculum style. The model is trained in 3m (3 marine vs 3 marine) scenario firstly. Then we used the pretrained 3m model to continually train on 5m (5 marine vs 5 marine) scenario and 7m (7 marines vs 7 marines) scenario. We also do a reverse experiment from 7m to 3m. During transfer learning, the model architecture of UPDeT keeps fixed. Considering RNN based model can not handle various input and output dimension, we modify the architecture of the source model of RNN when training on target scenario. We preserve the parameters of GRU cell and initialize fully connected layer with proper input and output dimension to fit the new scenario. The final result can be seen in Fig. 5a, Fig. 5b. Our proposed UPDeT show significant superiority against GRU based model. Statistically, the total timestep cost to converge with UPDeT is at least 10 times less than GRU based model and 100 times less than training from scratch. Besides, the model shows strong generalization ability without finetune, indicating UPDeT learns a robust policy with meta-level skill.

4.3 Extensive experiment on large-scale MAS

To evaluate the model performance in large-scale scenarios, we test our proposed UPDeT in 10m_vs_11m and 20m_vs_21m scenarios from SMAC and 64_vs_64 battle game in MAgent Environment (Zheng et al. (2017)). The final results can be found in Appendix E.

4.4 Attention based Strategy: An analysis

The significant performance improvement using UPDeT on SMAC multi-agent challenge can be credited to self-attention mechanism brought by both transformer blocks and policy decoupling strategy in UPDeT. In this section, we mainly discuss how the attention mechanism helps to learn



(a) Attention Matrix

(b) Strategy with Attention

Figure 6: An analysis on attention based strategy of UPDeT. Part (a) visualizes a typical attention matrix. To better explain how the attention mechanism works, we simplified the raw attention matrix and sum the attention weights by groups. Different groups correspond to different actions. Part (b) utilizes the simplified attention matrix to describe the relationship between attention and final strategy in a 3 marines vs 3 marines game. Discussion can be found in Section 4.4.

a much more robust and explainable strategy. Here we take 3 marines vs 3 marines (therefore the size of raw attention matrix is 6x6) as an instance to see how the attention mechanism works. As mentioned in the caption of Fig. 6, we simplify the raw complete attention matrix to a grouped attention matrix. Fig. 6b shows the three different stages in one episode including *Game Start*, *Attack* and *Survive* with corresponding attention matrix and strategies. In *Game Start* stage, the highest attention is in line 1 col 3 of the matrix, indicating that the agent pays more attention to its allies rather than enemies. This phenomenon can be interpreted as follows: In the startup stage of one game, all the allies are spawned at the left side of the map and are encouraged to find and attack the enemies in the right side with its allies (which is learned as a meta strategy in all different tasks. Although there is no enemy inside the ally's attack range or even view range, all allies will move to the right side to get more potential rewards). In *Attack* stage, the highest attention is in line 2 col 2 of the matrix, which means the enemy is now in the agent's attack range. Therefore, the agent will attack the enemy to get more rewards. Surprisingly, the agent chooses to attack the enemy with lowest health value. This indicates that a long term plan can be learned based on attention mechanism because killing the weakest enemy firstly can decrease the punishment from the future enemy attack. In *Survive* stage, the agent's health value is low and need to avoid being attacked. The highest attention lies in line 1 col 1, clearly showing that the most important thing under current circumstance is keeping alive. Once alive, there is still chance to return to the front line and get more reward when enemies are attacking the allies instead of the agent itself.

In conclusion, the self-attention mechanism and policy decoupling strategy of UPDeT provides a strong and clear relation between attention weights and final strategies. This relation can help us better understand the policy generation based on attentions among different entities. An interesting idea is: if we can find a strong mapping between attention matrix and final policy, the character of the agent can be modified in an unsupervised manner. If we want the agent to be brave, the simplest way is to manually lower the attention weight of the agent itself (e.g. lower the value of line 1 col 1 in the attention matrix in Fig. 6a) and vice versa. In general, all these advantages are credited to UPDeT which equips the MARL framework a more explainable structure on policy generation.

5 Conclusion

In this paper, we propose UPDeT, a universal policy decoupling transformer model extending MARL to a much broader scenario. UPDeT is general enough to be plugged into any existing MARL method. Moreover, our experiment results show that, combined with UPDeT, existing state of the art MARL methods can further get significant improvement with same training pipeline. On transfer learning tasks, our model is 100 times faster than training from scratch and 10 times faster than training from RNN based model. In the future, we aim to conduct a centralized function based on UPDeT and apply the self-attention mechanism into whole pipeline of MARL framework to make further improvement.

References

Haitham B Ammar, Karl Tuyls, Matthew E Taylor, Kurt Driessens, and Gerhard Weiss. Reinforcement learning transfer via sparse coding. In *Proceedings of the 11th international conference on autonomous agents and multiagent systems*, volume 1, pp. 383–390. International Foundation for Autonomous Agents and Multiagent Systems ..., 2012.

Mariusz Bojarski, Davide Del Testa, Daniel Dworakowski, Bernhard Firner, Beat Flepp, Prasoon Goyal, Lawrence D Jackel, Mathew Monfort, Urs Muller, Jiakai Zhang, et al. End to end learning for self-driving cars. *arXiv preprint arXiv:1604.07316*, 2016.

Georgios Boutsioukis, Ioannis Partalas, and Ioannis Vlahavas. Transfer learning in multi-agent reinforcement learning domains. In *European Workshop on Reinforcement Learning*, pp. 249–260. Springer, 2011.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

Felipe Leno Da Silva and Anna Helena Reali Costa. A survey on transfer learning for multiagent reinforcement learning systems. *Journal of Artificial Intelligence Research*, 64:645–703, 2019.

Yali Du, Lei Han, Meng Fang, Ji Liu, Tianhong Dai, and Dacheng Tao. Liir: Learning individual intrinsic reward in multi-agent reinforcement learning. In *Advances in Neural Information Processing Systems*, pp. 4403–4414, 2019.

Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. *arXiv preprint arXiv:1705.08926*, 2017.

Abhishek Gupta, Coline Devin, YuXuan Liu, Pieter Abbeel, and Sergey Levine. Learning invariant feature spaces to transfer skills with reinforcement learning. *arXiv preprint arXiv:1703.02949*, 2017.

Matthew Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *arXiv preprint arXiv:1507.06527*, 2015.

Todd Hester, Michael Quinlan, and Peter Stone. Generalized model learning for reinforcement learning on a humanoid robot. In *2010 IEEE International Conference on Robotics and Automation*, pp. 2369–2374. IEEE, 2010.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Wan Ju Kang David Earl Hostallero, Kyunghwan Son, Daewoo Kim, and Yung Yi Qtran. Learning to factorize with transformation for cooperative multi-agent reinforcement learning. In *Proceedings of the 31st International Conference on Machine Learning, Proceedings of Machine Learning Research. PMLR*, 2019.

Ryan Lowe, Yi I Wu, Aviv Tamar, Jean Harb, OpenAI Pieter Abbeel, and Igor Mordatch. Multi-agent actor-critic for mixed cooperative-competitive environments. In *Advances in neural information processing systems*, pp. 6379–6390, 2017.

Anuj Mahajan, Tabish Rashid, Mikayel Samvelyan, and Shimon Whiteson. Maven: Multi-agent variational exploration. In *Advances in Neural Information Processing Systems*, pp. 7613–7624, 2019.

Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *nature*, 518(7540):529–533, 2015.

Frans A Oliehoek, Christopher Amato, et al. *A concise introduction to decentralized POMDPs*, volume 1. Springer, 2016.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

Emilio Parisotto, Jimmy Lei Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. *arXiv preprint arXiv:1511.06342*, 2015.

Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Łukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. *arXiv preprint arXiv:1802.05751*, 2018.

Peng Peng, Ying Wen, Yaodong Yang, Quan Yuan, Zhenkun Tang, Haitao Long, and Jun Wang. Multiagent bidirectionally-coordinated nets: Emergence of human-level coordination in learning to play starcraft combat games. *arXiv preprint arXiv:1703.10069*, 2017.

Tabish Rashid, Mikayel Samvelyan, Christian Schroeder De Witt, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. Qmix: Monotonic value function factorisation for deep multi-agent reinforcement learning. *arXiv preprint arXiv:1803.11485*, 2018.

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. *arXiv preprint arXiv:1902.04043*, 2019.

Kun Shao, Yuanheng Zhu, and Dongbin Zhao. Starcraft micromanagement with reinforcement learning and curriculum transfer learning. *IEEE Transactions on Emerging Topics in Computational Intelligence*, 3(1):73–84, 2018.

Peter Sunehag, Guy Lever, Audrunas Gruslys, Wojciech Marian Czarnecki, Vinicius Zambaldi, Max Jaderberg, Marc Lanctot, Nicolas Sonnerat, Joel Z Leibo, Karl Tuyls, et al. Value-decomposition networks for cooperative multi-agent learning. *arXiv preprint arXiv:1706.05296*, 2017.

Ming Tan. Multi-agent reinforcement learning: Independent vs. cooperative agents. In *Proceedings of the tenth international conference on machine learning*, pp. 330–337, 1993.

Matthew E Taylor and Peter Stone. Transfer learning for reinforcement learning domains: A survey. *Journal of Machine Learning Research*, 10(7), 2009.

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*, pp. 5998–6008, 2017.

Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019.

Weixun Wang, Tianpei Yang, Yong Liu, Jianye Hao, Xiaotian Hao, Yujing Hu, Yingfeng Chen, Changjie Fan, and Yang Gao. From few to more: Large-scale dynamic multiagent curriculum learning. In *AAAI*, pp. 7293–7300, 2020.

Xiaolong Wang, Ross Girshick, Abhinav Gupta, and Kaiming He. Non-local neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7794–7803, 2018.

Yaodong Yang, Lantao Yu, Yiwei Bai, Jun Wang, Weinan Zhang, Ying Wen, and Yong Yu. A study of ai population dynamics with million-agent reinforcement learning. *arXiv preprint arXiv:1709.04511*, 2017.

Yaodong Yang, Rui Luo, Minne Li, Ming Zhou, Weinan Zhang, and Jun Wang. Mean field multi-agent reinforcement learning. *arXiv preprint arXiv:1802.05438*, 2018.

Yaodong Yang, Ying Wen, Lihuan Chen, Jun Wang, Kun Shao, David Mguni, and Weinan Zhang. Multi-agent determinantal q-learning. *arXiv preprint arXiv:2006.01482*, 2020.

Lianmin Zheng, Jiacheng Yang, Han Cai, Weinan Zhang, Jun Wang, and Yong Yu. Magent: A many-agent reinforcement learning platform for artificial collective intelligence. *arXiv preprint arXiv:1712.00600*, 2017.

Meng Zhou, Ziyu Liu, Pengwei Sui, Yixuan Li, and Yuk Ying Chung. Learning implicit credit assignment for multi-agent actor-critic. *arXiv preprint arXiv:2007.02529*, 2020.

A Details of SMAC environment

The action space contains 4 move directions, k attack actions where k is the fixed maximum number of the enemy units in a map, stop and none-operation. At each time step, the agents receive a joint team reward which is defined by the total damage of the agents and the total damage from the enemy side. Each agent is described by several attributes including the health point HP, weapon cooling down (CD), unit type, last action and the relative distance of the observed units. The enemy unit is described in the same way except that CD is excluded. The partial observation of an agent is composed by the attributes of the units, including both the agents and the enemy units, shown up within its view range that is a circle with a certain radius.

B Details of Model

The transformer block in all different experiment consists of 3 heads and 3 layer of self-attention layers. The other important training hyper parameters are as follows:

Hyper Parameters List	
Name	Value
batch size	32
test interval	2000
gamma	0.99
buffer size	5000
token dimension (UPDeT)	32
channel dimension (UPDeT)	32
epsilon start	1.0
epsilon end	0.05
rnn hidden dimension	64
target net update interval	200
mixing embeddding dimension (QMIX)	32
hypernet layers (QMIX)	2
hypernet embedding (QMIX)	64
mixing embeddding dimension (QTRAN)	32
opt loss (QTRAN)	1
nopt min loss (QTRAN)	0.1

No addtional hyper parameters for VDN.

C SOTA MARL value based Framework

A brief summary of three SOTA methods:

- VDN (Sunehag et al. (2017)): the method learns an individual Q-value function and representing Q_{tot} as a sum of individual Q-value functions that condition only on individual observations and actions.
- QMIX (Rashid et al. (2018)): the method learns decentralized Q-function for each agent with the assumption that the centralized Q-value is monotonically increasing with the individual Q-values.
- QTRAN (Hostallero et al. (2019)): the method formulates multi-agent learning as an optimisation problem with linear constraints and relaxing it with L2 penalties for tractability.

D UPDeT on SMAC: A real case

We take 3 marines vs 3 marines challenge from SMAC with UPDeT as an example. Details can be found in Fig. 7. The observation has been separated to 3 groups: own agent, 2 other ally agents and 3 enemies. The policy output including basic action corresponding to own agent observation and attack action one to each corresponding enemy observation. The hidden state is added after

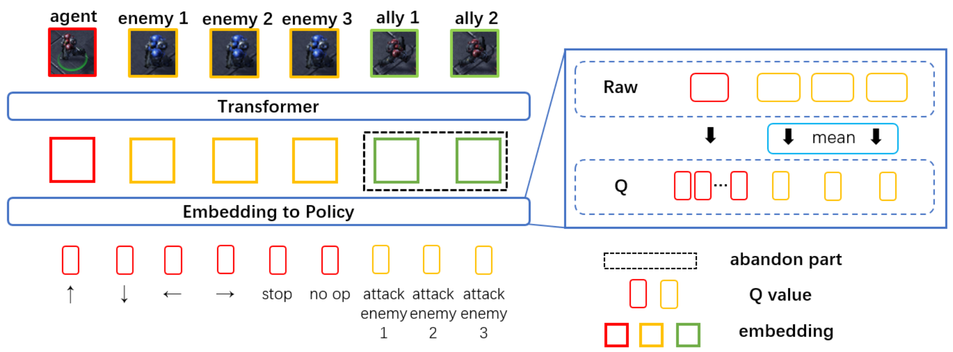
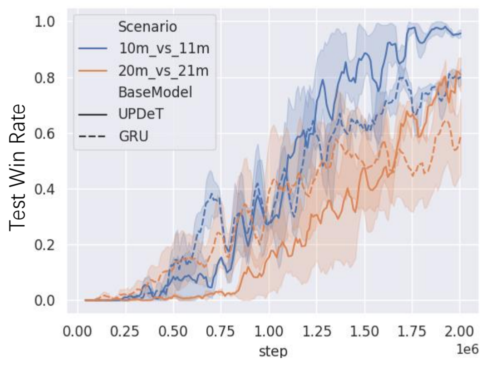
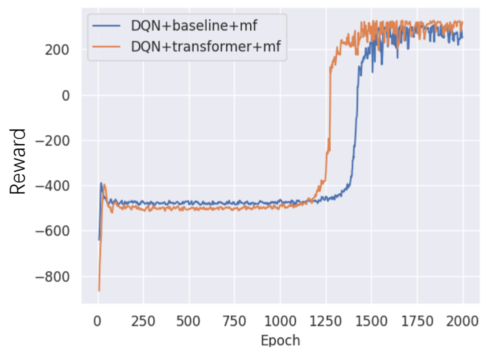


Figure 7: Real case on 3 Marines vs 3 Marines Challenge from SMAC.



(a) Large Scale Scenarios



(b) MAgent Battle: 64 vs 64

Figure 8: Experiment result on large scale MAS including SMAC and MAgent.

embedding layer. The output of other agents is abandoned as there is no corresponding action. Once an agent or enemy died, we mask corresponding unavailable action in action select stage to pick the available actions.

E RESULTS OF EXTENSIVE EXPERIMENT ON LARGE SCALE

We further test the robustness of UPDeT in large-scale multi-agent system. We enlarge the game size in SMAC (Samvelyan et al. (2019)) to incorporate more agents and enemies to the battle field. We take 10 marines vs 11 marines game and 20 marines vs 21 marines game to compare the performance between UPDeT and GRU based approaches. In 20 marines vs 21 marines game, to accelerate the training and satisfy the hardware limitation, we decrease the batch size of both GRU baseline and UPDeT from 32 to 24 in training stage. The final result can be found in Fig. 8a. The improvement is still significant on both sample efficiency and final performance. Besides, it is worth mentioning that the model size of UPDeT keeps fixed while the GRU based model becomes larger in large-scale senarios. In 20 marines vs 21 marines game, the model size of GRU is almost double to the UPDeT. This indicates that the UPDeT ensures the lightness of the model and can still perform well in different scenarios.

We also test the model performance in MAgent Enviroment (Zheng et al. (2017)). The setting of MAgent is quite different from SMAC. First, the observation size and available action number is not related to the agent number. Second, the 64_vs_64 battle game we tested is a two-player zero-sum game which is another hot research area combined both MARL and GT (Game Theory) and the most successful try in this area is adopting mean-field approximation of GT in MARL to accelerate the self-play training (Yang et al. (2018)). Third, as for model architecture, there is no need to use recurrent network like GRU in MAgent and the large size of observation requires a CNN to embed. However, treating UPDeT as a pure encoder without recurrent architecture, we can still conduct experiments on MAgent and the final results can be found in Fig. 8b. The UPDeT is better than DQN baseline but the improvement is not so significant as in SMAC.